University of Pune

MCA - II

(Under Science Faculty)

Lab Course – 305

Lab Manual

Name: _____

Roll No. _____Academic Year: 20_____ - 20_____

College Name: _____

Preface:

In pursuance of the decision to implement credit system at the post graduate level and ensure continuous assessment, the UOP has decided to implement the credit & semester system (CSS) in all its affiliated college and recognized institutions where post graduate courses are conducted.
Assessment shall consist of continuous assessment (CA and END of Semester Examination (ESE)). Each shall have an equal weight age of 50%.

Assessment and grade point average

The system of evaluation is as follows: Each CA and ESE will be evaluated in terms of marks. The marks for CA and ESE will be added together and then converted into a grade and later a grade point average.
Results will be declared for each semester.

After the gain of minimum number of credits after completion of a PG program, a student will get a grade sheet with total grades earned and a grade point average.
Marks /Grade/Grade point

Marks	Grade	Grade Point
100 to 75	O : Outstanding	06
74 to 65	A : Very good	05
64 to 55	B : Good	04
54 to 50	C : Average	03
49 to 45	D : Satisfactory	02
44 to 40	E : Pass	01
39 to 0	F : Fail	00

Final Grade Points	
Grade Points	Final Grade
5.00 – 6.00	O
4.50 – 4.99	A
3.50 – 4.49	B
2.50 – 3.49	C
1.50 – 2.49	D
0.50 – 1.49	E
0.00 – 0.49	F

Practical Evaluation Format

The internal continuous assessment will be 50M and end semester lab examination will be of 50M.

The outline of distribution of Practical Marks for various aspects /mechanisms towards Continuous Assessment is as follows:

Sr. No.	Distribution Marks	Marks
1.	Journal After finishing everypractical instructor should give the marks to respective assignment and evaluate those marks at the end of the Semester out of 15 (OS and Core Java)	30
2.	Internal Practical Evaluation Respective subject teachers should evaluate Operating System and Core Java assignments for 10M each by conducting mock tests. Oral	10 10
	Total	50M

The outline of distribution of Practical Marks for towards End Semester lab examination is as follows:

Sr. No.	Distribution Marks	Marks
1	Core Java Programs	20
2	Operating System program	20
3	Lab book	5
4	Viva	5
		50M

Introduction

1. About the work book

This workbook is intended to be used by MCA-II (Computer Science) students for the laboratory course CA 305. In MCA, hands-on laboratory experience is critical to the understanding of theoretical concepts studied in the theory courses. This workbook provides numerous computing problems covering all difficulty levels.
The objectives of this book are

- Defining clearly the scope of the course
- Bringing uniformity in the way the course is conducted across different colleges
- Continuous assessment of the course
- Bring in variation and variety in the experiments carried out by different students in a batch
- Catering to the need of slow paced as well as fast paced learners

2. How to use this workbook

This workbook is mandatory for the completion of the laboratory course. It is a measure of the performance of the student in the laboratory for the entire duration of the course.

2.1. Instructions to the students

- Students are expected to carry this book every time they come to the lab for practicals
- Student should maintain separate journal for the source code
- Student should read the topics mentioned in Reading section of this book before coming for practical
- Students should solve only those exercises which are selected by instructor as a part of journal activity. However, students are free to solve additional exercises to do more practice for their practical examination

Exercise	Difficult Level	Rule
Set A	Easy	All programs are compulsory
Set B	Medium	At least 1 program is compulsory
Set C	Hard	Not compulsory

- Student will be assessed for each exercise on a scale of 5

Not Done	0
Incomplete	1
Late Complete	2
Needs Improvement	3
Complete	4
Well Done	5

2.2. Instruction to the Instructors

- Explain the assignment and related concepts in around ten minutes using white board if required or by demonstrating the software
- After a student completes a specific set, the instructor has to verify the outputs and sign in the provided space after the activity
- Ensure that the students use good programming practices
- You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box
- The value should also be entered on assignment completion page

2.3. Instructions to the Lab administrator

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client side are as given below

1) Server Side (Operating System)
    a. * Fedora Core Linux /Red Hat Linux
    b. * Microsoft Windows Server 2003
    c. Apache Tomcat Server
    d. Servers Side (software's to be installed)
    In Linux – C, jdk1.5,onwards,postgresql/Mysql
    In WinXP -- MSOffice

2) Client Side (Operating System)
    a. * Red Hat Linux and Fedora Core
    b. * Microsoft Windows XP
    c. Client Side (software's to be installed)
    In Linux – C, jdk1.5,onwards,postgresql/Mysql
    In WinXP -- MSOffice

Compiled From : T.Y.B.Sc (computer science Lab book)

Prepared and Compiled by :
Ms. Manisha Bharambe, MES Abasaheb Garware College, Pune
Ms. Kiran Tiwari, MES Abasaheb Garware College, Pune
Ms. Saee Joshi, MES Abasaheb Garware College, Pune
Ms. Mahek Shaikh, MES Abasaheb Garware College, Pune

Reviewed by:
Ms. Chitra Nagarkar, MES Abasaheb Garware College, Pune

Ms. Manisha Bharambe, MES Abasaheb Garware College, Pune

SECTION I - Assignment Evaluation(O.S)

Sr. No	Assignment Name	Teachers Sign	Assignment Marks	Viva Marks
1	CPU Scheduling			
2	Deadlock			
3	Paging			
4	File allocation			
5	Disk Scheduling			

SECTION II- Assignment Evaluation(Core Java)

Sr. No	Assignment Name	Teachers Sign	Assignment Marks	Viva Marks
1	Java Tools			
2	Array of Objects, Packages			
3	Constructor Inheritance and Interfaces			
4	Exception Handling and Assertions			
5	I/O and File Handling			
6	GUI using Swing			
7	Event Handling			
8	Applets			
9	Threading			

Name Of Batch Incharge:

Signature Of Batch Incharge: Head of Department

Date:

SECTION I
OPERATING SYSTEM

SESSION 1

CPU Scheduling

Ready reference
• Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

CPU scheduling decisions may take place when a process:

1. Switches from running to waiting state.
2. Switches from running to ready state.
3. Switches from waiting to ready.
4. Terminates.

Scheduling under 1 and 4 is non preemptive.
All other scheduling is preemptive

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – Number of processes that complete their execution per time unit.
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Optimization Criteria

- Max throughput
- Max CPU Utilization
- Min turnaround time
- Min waiting time
- Min response time

There are 5 CPU Scheduling Algorithms:

1. FCFS

- Since context switches only occur upon process termination, and no reorganization of the process queue is required, scheduling overhead is minimal.
- Throughput can be low, since long processes can hog the CPU.

- Turnaround time, waiting time and response time can be low for the same reasons above.
- No prioritization occurs, thus this system has trouble meeting process deadlines.
- The lack of prioritization does permit every process to eventually complete, hence no starvation.

2. SJF

With this strategy the scheduler arranges processes with the least estimated processing time remaining to be next in the queue. This requires advanced knowledge or estimations about the time required for a process to complete.

- If a shorter process arrives during another process' execution, the currently running process may be interrupted, dividing that process into two separate computing blocks. This creates excess overhead through additional context switching. The scheduler must also place each incoming process into a specific place in the queue, creating additional overhead.
- This algorithm is designed for maximum throughput in most scenarios.
- Waiting time and response time increase as the process' computational requirements increase. Since turnaround time is based on waiting time plus processing time, longer processes are significantly affected by this. Overall waiting time is smaller than FIFO, however since no process has to wait for the termination of the longest process.
- No particular attention is given to deadlines; the programmer can only attempt to make processes with deadlines as short as possible.
- Starvation is possible, especially in a busy system with many small processes being run.

3. Priority

- The O/S assigns a fixed priority rank to every process, and the scheduler arranges the processes in the ready queue in order of their priority. Lower priority processes get interrupted by incoming higher priority processes.
- Overhead is not minimal, nor is it significant.
- Waiting time and response time depend on the priority of the process. Higher priority

processes have smaller waiting and response times.

- Deadlines can be met by giving processes with deadlines a higher priority.
- Starvation of lower priority processes is possible with large amounts of high priority processes queuing for CPU time.

4. Round Robin

The scheduler assigns a fixed time unit per process, and cycles through them.

- RR scheduling involves extensive overhead, especially with a small time unit.
- Balanced throughput between FCFS and SJF, shorter jobs are completed faster than in FCFS and longer processes are completed faster than in SJF.
- Fastest average response time, waiting time is dependent on number of processes, and not average process length.
- Because of high waiting times, deadlines are rarely met in a pure RR system.
- Starvation can never occur, since no priority is given. Order of time unit allocation is based upon process arrival time, similar to FCFS.

5. Round Robin with Multilevel Feedback Queues

This is used for situations in which processes are easily classified into different groups. For example, a common division is made between foreground (interactive) processes and background (batch) processes. These two types of processes have different response-time requirements and so may have different scheduling needs.

SET A

1. Write the simulation program using FCFS. The arrival time and first CPU bursts of different jobs should be input to the system. Assume the fixed I/O waiting time. The output should give the Gantt chart, Turnaround Time and Waiting time for each process and average times.
2. Write the simulation program using SJF (non-preemptive). The arrival time and first CPU bursts of different jobs should be input to the system. The Assume the fixed I/O waiting time. The output should give the Gantt chart, Turnaround Time and Waiting time for each process and average times.

SET B

1. Write the simulation program for preemptive scheduling algorithm using SJF/Priority. The

arrival time and first CPU bursts of different jobs should be input to the system. Assume the fixed I/O waiting time (2 units). The next CPU burst should be generated using random function. The output should give the Gantt chart, Turnaround Time and Waiting time for each process and average times.

2. Write the simulation program for preemptive scheduling algorithm using Round Robin with time quantum of 2 units. The arrival time and first CPU bursts of different jobs should be input to the system. Assume the fixed I/O waiting time (2 units). The next CPU burst should be generated using random function.

The output should give the Gantt chart, Turnaround Time and Waiting time for each process and average times.

SET C

Write the simulation program using Round Robin with multilevel feedback queues. The arrival time and first CPU bursts of different jobs should be input to the system. Assume the fixed I/O waiting time (2 units). The next CPU burst should be generated using random function. The output should give the Gantt chart, Turnaround Time and Waiting time for each process and average times.

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 2

Deadlock- Banker's Algorithm

Ready Reference

Definition:

Deadlock is a situation where each process in a set of processes wait for the resources held by another process. Thus, a set of processes is in a deadlock state when every process in the set is waiting for an event that can be caused by another process in the set.

Necessary conditions to occur a deadlock:

A deadlock situation can arise if the following four conditions hold simultaneously in the system:

- **Mutual Exclusion**: At least one resource must be held in a non sharable mode. Only one process at a time can use the resource.
- **Hold and Wait**: A resource must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.
- **No Preemption**: Resources cannot be preempted; that is, a resource cannot be released only voluntarily by the process holding it, after that process has completed its task.
- **Circular Wait**: A set {P0,P1,...Pn} of waiting processes must exist such that P0 is waiting for a resource that is held by P1,P1 is waiting for a resource that is held by P2,...,Pn-1 is waiting for a resource that is held by Pn and Pn is waiting for a resource that is held by P0.

There are three ways to handle a deadlock:

- **Deadlock Prevention**: It is a method to ensure that at least one of the necessary conditions cannot hold.
- **Deadlock Avoidance**: This method requires that the operating system be given in advance additional information concerning which resources a process will request and use during its lifetime. With this additional information, we can decide for each request whether or not a process should wait. To decide whether the current request can be satisfied or must be delayed, the system must consider the resources currently to each process, and the future requests and releases of each process.
- **Deadlock Detection and Recovery**: If a system does not employ deadlock prevention or deadlock avoidance algorithm, then a deadlock situation may occur. In this situation, the system can provide an algorithm that examines the state of the system to determine whether a deadlock has occurred and an algorithm to recover from the deadlock.

Explanation of avoidance algorithm, safe state, safe sequence.

Deadlock Avoidance:

If detail information about the processes and resources is available then it is possible to avoid deadlock, e.g. which process will require which resources, possibly in what sequence, etc. This information may help to decide the sequence in which the processes can be executed to avoid deadlock. Each request can be analyzed on the basis of number of resources currently available, currently allocated and future requests which may come from other processes. From this information system can decide whether a process should wait or not.

The Deadlock avoidance algorithm dynamically examines the resource-allocation state to ensure that a circular wait can never exist. The resource- allocation state is defined by the number of available and allocated resources and the maximum demands of the processes.
If a safe sequence in which processes can be executed is available, then only we can call the system to be in safe state. A safe sequence is the sequence of process such that, if processes are executed in safe sequence then each process in the sequence will be executed serially, with current available resources. Then release, all the resources held by it making available for next process in the sequence so on. A system without safe sequence is unsafe. Unsafe system may or may not have a deadlock.

Banker's Algorithm is one of the Deadlock avoidance algorithms.
For these we require following data structures:
Allocation: An n*m matrix defines the number of resources of each type currently allocated to each process. If Allocation[i][j]=k,then process Pi is currently allocated k instances of resource type Rj.
Max: An n*m matrix defines the maximum demand of each process. If Max[i][j]=k, then process Pi may request at most k instances of resource type Rj.
Available: a vector of length m indicates the number of available resources of each type. If Available[j]=k, then there are k instances of resource type Rj available.
Need: An n*m matrix indicates the remaining resource need of each process.
If Need[i][j]=k, then process Pi may need k more instances of resource type Rj to complete its task. Note that: Need[i][j]=Max[i][j]-Allocation[i][j].

Safety Algorithm:-
This is the algorithm used for finding out whether or not a system is in a safe state or not.
The algorithm is as follows:
Let n be the number of processes in the system. Let m be the number of resources

1) Let Work and Finish be the vectors of length m and n respectively. Initialize Work = Available and Finish[i] =false for i= 1, 2, 3,..., n.

2) Find an i such that

a) Finish[i]=false

b) Need i <= Work

If no such i exists, go to step 4.

III. Work = Work + Allocation i Finish[i]=true

Go to step 2.

JJJ. If Finish[i] =true for all i, then the system is in a safe state.

When a request of resources is made by a process the following algorithm is used.
Resource-Request Algorithm:-
Let request i be the vector for process Pi. If request i[j]=k,then process Pi wants k instances of resource type Rj.
When a request of resources is made by a process the following actions are taken:
1. If request i <= Need i, go to step 2. Otherwise, raise an error condition, since the process has exceeded its maximum claim.

2. If request i <= Available, go to step 3. Otherwise, Pi must wait, since the resources are not available.
3. Assume that the system pretends to have allocated the requested resources to the process Pi by modifying the state as follows:

a) Available = Available - Request i
b) Allocation i = Allocation i + Request i
c) Need i = Need i - Request i

If the resulting resource-allocation state is safe, the transaction is completed and process Pi is allocated its resources i.e. the request is granted immediately. If the resulting resource-allocation state is unsafe, then the process Pi must wait for Request i and the old transaction i.e. assumed resource- allocation state is restored. That is because the system is not in safe state, so request is not granted immediately.

Set A

Q.1) Consider the following snapshot of a system:

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

1. Display the contents of Need array.
2. Check whether the system is in safe state or not. If yes, give the safe sequence.

Q.2) Consider the following snapshot of a system:

1. Display the contents of Need array.
2. Check whether the system is in safe state or not. If yes, give the safe sequence.

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	6	3	2	0	6	5	2				
P4	0	0	1	4	0	6	5	6				

Set B

Q.1) Consider the following snapshot of a system:

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

1. Display the contents of Need array.
2. Check whether the system is in safe state or not. If yes, give the safe sequence.
3. If a request from process P1 arrives for (1, 0, 2) can it be immediately granted?

Q.2) Consider the following snapshot of a system:

Consider the following snapshot of a system. A system has five processes, A through E and four types of the resources, R1 to R4.Answer the following questions using Banker's algorithm.

1. Display the contents of Need array.
2. Check whether the system is in safe state or not. If yes, give the safe sequence.
3. If a request from process P4 arrives for (0, 4, 2, 0) can it be immediately granted.

Process	Allocation				Max				Available			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
A	3	0	1	1	4	1	1	1	6	3	4	2
B	0	1	0	0	0	2	1	2				
C	1	1	1	0	4	2	1	0				
D	1	1	0	1	1	1	1	1				
E	0	0	0	0	2	1	1	0				

Q.3) Consider the following snapshot of a system:

1. Display the contents of Need array.
2. Check whether the system is in safe state or not. If yes, give the safe sequence.
3. If a request from process P4 arrives for (0, 0, 1) can it be immediately granted.

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	0	1	0	0	0	0
P1	2	0	0	4	0	2			
P2	3	0	3	3	0	3			
P3	2	1	1	3	1	1			
P4	0	0	2	0	0	1			

Assignment Evaluation

0:Not Done [ ] 1:Incomplete [ ] 2.Late Complete [ ]

3:Needs Improvement [ ] 4:Complete [ ] 5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 3

Paging

Ready reference

Basic memory management concept:

1. **Main memory** is a central part of the computer system.CPU and I/O system interact with memory.
   - Logical address is the address generated by CPU or programs.
   - Physical address is the address where actual program data are stored in memory.
2. **Virtual memory** is something that appears to exist, but actually does not exist. Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available. If memory requirement of a program is larger than available memory then virtual memory is used.
3. **Paging** is a non-contiguous memory management scheme. User program will allocate a memory wherever available. In the paging memory-management scheme, the operating system retrieves data from secondary storage in same-size blocks called **pages**. The main advantage of paging is that it allows the physical address space of a process to be noncontiguous.
4. **Logical memory** is the user's memory .It is divided into number of equal units called
   Logical memory is the user's memory .It is divided into number of equal units called **pages.**
5. **Physical memory** is divided into number of equal units called **frames.**
6. **Page table** is used to map a page on a frame.

Demand paging –A page is demanded or requested by a user program, if not present in the memory i.e. in frame .Then it is given from secondary storage device. When a user program tries to access the page which is not present in memory, the situation is called as **page fault.**

Page replacement:

Memory management is one of the important responsibilities of the operating system. In program execution it may happen that there is no free frame available then a page fault occurs, System cannot terminate the user program, because of virtual memory.

To bring user page into the frame, a frame is freed and user page brought in. The process of freeing the frame is called as **page replacement.**

The important part is that which frame should be freed. This frame is called as **victim frame** and the page present on that frame is called **victim page.**

Steps involved in page replacement are:

1. Find the location of the desired page on the disk.
2. Find a free frame:
    a. If there is a free frame, use it.
    b. If there is no free frame, use a page replacement algorithm to select a victim frame
    c. write the victim frame to the disk; change the page and frame tables accordingly.
3. Read the desired page into the newly freed frame; change the page and frame tables.
4. Restart the user process.

Page Replacement Algorithm:

O. S. uses one of the page replacement schemes lesser the page fault rate, better the page replacement algorithm.

The various page replacement algorithms are:

- FIFO
- Optimal Replacement
- LRU
- LRU approximation using reference bit/bits
- MRU
- Second chance algorithm
- LFU
- MFU

Input of page replacement algorithm is **reference string** reference string is stored page numbers of demand pages and next input is the no of page frames available & its main function is to load required page in free frame.

If there is no free frame then it select victim page.

Generation of reference string as follows:

Consider a address sequence 0100, 0432, 0101,0612,0611,0105....
At 100 bytes per page, this sequence is reduced to the following reference string

1, 4, 1, 6, 1, 6, 1.....

FIFO (First In First Out):

In this algorithm all available frames are given to the pages from reference string serially. i.e. first frame to first pass, second frame to second page & so on. All available/free frames are over at that time, the first frame is selected as victim frame, next time second frame is selected and so on.

Note: If page is not present in frame then page fault is counted.

Data Structure:

1. M: total no of reference string.
2. RS: Stores reference string i.e demand page number.
3. F: available n free frames.
4. Rear: page is stored at rear frame after insertion rear is increments by one.
5. Front: It always points to first page after deletion it is increment by one.

Algorithm:

1. Find out first free frame.
2. If free frame is not available then free front frame and update rear and Front
3. Load F[rear] =RS [currp] page
4. Update rear, front, currp.
5. Count the page fault.
6. Repeat steps 1 to 5 till reference string not over.
7. Stop.

LRU (Least Recently Used):

It selects least recently used page of the main memory as a victim page. It refers to the past reference of the pages. The one which is not used for longest time is selected for replacement.
In short LRU selects the farthest page in the left hand side direction of currently faulted page.
LRU is implemented using **(A) counter (B) stack**.

(A) Using Counter

Data Structure:

1. M: total no of reference string.

2. RS: Stores reference string i.e demand page number.
3. F: available n free frames.
4. Counter: is attached with each page. It counts page is referenced or not .Select smallest count page which is least recently used is selected for the victim page.

Algorithm:

1. Find out first free frame.
2. If free frame not available then free least recently used page from frame using past reference i.e. from RS [0] to RS[currp]
3. Load RS [currp] into the LRU page frame
4. currp++
5. Count the page fault
6. Repeat step 1 to 5 till reference string is not over
7. Stop.

(B) Using stack

Data Structure:

1. M: total no of reference string.
2. RS: Stores reference strings i.e. demand page number.
3. F: available n free frames.
4. Stack: To keep a stack of page numbers. Whenever a page is referenced, it is removed from stack and put on the top. In this way least recently used pages are found at bottom of the stack.

Algorithm:

1. Find out first free frame
2. If free frame not available then free least recently used page stack (access the page which is stored at bottom)
3. Load RS[currp] into the LRU page frame
4. currp++
5. count the page fault
6. Repeat step 1 to 5 till reference string is not over
7. Stop.

Second Chance Algorithm:

Data Structure:

1. M: total no of reference string.
2. RS: Stores reference strings i.e. demand page number.
3. F: available n free frames.
4. Ref_bit: Reference bit of page which stores 1 or 0.

Algorithm:

1. Find out first free frame.
2. If free frame is not available then Ref_bit of page is checked serially, if it is 0 then the bit is set to 1and page is replaced. If it is 1 the bit is set to zero. The first page along with reference bit 0 will appear at first position and will be selected for replacement.
3. Load F =RS [currp] page
4. Count the page fault.
5. Repeat steps 1 to 4 till reference string not over.
6. Stop.

MFU (Most Frequently Used):

Data Structure:

1. M: total no of reference string.
2. RS: Stores reference string I. E demand page number.
3. F: available n free frames.
4. Counter: A reference counter.

Algorithm:

1. Find out first free frame.
2. If free frame is not available then select MFU page using reference counter

3. Load F =RS [currp] page
4. Update currp and reference counter.
5. Count the page fault.
6. Repeat steps 1 to 5 till reference string not over.
7. Stop.

Optimal replacement:
Data Structure:

1. M: total no of reference string.
2. RS: Stores reference string i.e. demand page number.
3. F: available n free frames.

Algorithm

1. Find out first free frame
2. If free frame not available then free optimal page from frame using future reference i.e. from RS [currp] to RS[M]
3. Load RS[currp] into the optimal page frame
4. count the page fault
5. Increment currp by one
6. Repeat step 1 to 5 till reference string is not over
7. Stop.

Note: Students can use suitable data structure, if necessary.

Set A

Q1. Consider the following page reference string:

1,2,3,4,1,2,5,1,2,3,4,5

Write the simulation program to find page faults occur for the following page replacement algorithm. Assume three frames.

a) LRU page replacement. **b) Optimal page replacement.**

Q2. Write the simulation program for demand paging and show the page scheduling and total number of page faults using following algorithms. Assume memory of n frames. Consider

following page reference string:

a) 7,0,1,2,0,3,0,4,2,3,0,3,2,,1,2,0,1,7,0,1
b) 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

1) Implementation of FIFO
2) Implementation of LRU using STACK
3) Implementation of LRU using COUNTER

Set B

Write the simulation program for demand paging and show the page scheduling and total number of page faults using following algorithms. Assume memory of n frames. Consider following page reference string:

a) 9,14,10,11,15,9,11,9,15,10,9,15,10,12,15
b) 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

1) Implementation of second chance.
2) Implementation of MFU

Set C

Write the simulation program for demand paging and show the page scheduling and total number of page faults using following algorithms. Assume memory of n frames. Consider following page reference string:

a) 9,14,10,11,15,9,11,9,15,10,9,15,10,12,15
b) 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

1) Implementation of Optimal page replacement
2) Implementation of LFU
3) Implementation of MRU

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 4

FILE ALLOCATION METHODS

The main idea behind allocation is effective utilization of file space and fast access of the files. There are three types of allocation:

Sequential (contiguous)

Each file occupies a set of contiguous blocks on the disk.

- Simple – only starting location (block #) and length (number of blocks) are required.

Random access.

- Wasteful of space (dynamic storage-allocation problem).
- Files cannot grow.

Linked

- Simple – need only starting address
- Free-space management system – no waste of space
- No random access

Indexed

- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.

Set A

Write a C Program to simulate Sequential (contiguous) File Allocation Method. Assume Disk of size‘d’, value of which should be taken from user. Use separate Tables (implemented using linked lists) to keep track of Used and Free space respectively. Make use of the following Menu to perform Operations:

1) Allocate space for newly created file.
2) Deallocate space for now-deleted file.
3) Show Used and/or Free Space on Disk.
4) Exit

Set B

Write a C Program to simulate Linked File Allocation Method. Assume Disk of size'd', value of which should be taken from user. Use separate Tables (implemented using linked lists) to keep track of Used and Free space respectively. Make use of the following Menu to perform Operations:

1) Allocate space for newly created file.
2) Deallocate space for new-deleted file.
3) Show Used and/or Free Space on Disk.
4) Exit

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 5

Disk Scheduling

- File systems must be accessed in an efficient manner, especially with hard drives, which are the slowest part of a computer. As a computer deals with multiple processes over a period of time, a list of requests to access the disk builds up. For efficiency purposes, all requests (from all processes) are aggregated together. The technique that the operating system uses to determine which requests to satisfy first is called disk scheduling.
- One of the responsibilities of the OS is to use the hardware efficiently. For the disk drives, meeting this responsibility entails having fast access time and large disk bandwidth.
- The access time has two major components.
- The **seek time** is the time for the disk arm to move the heads to the cylinder containing the desired sector.
- The **rotational latency** is the additional time for the disk to rotate the desired sector to the disk head.
- The disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
- We can improve both the access time and the bandwidth by scheduling the servicing of disk I/O requests in a good order.
- For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus, when one request is completed, the OS chooses which pending request to service next. How does the OS make this choice?

Disk scheduling algorithms:

FCFS Scheduling:

- The simplest form of disk scheduling is the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it generally does not provide the fastest service.

SSTF Scheduling:

- It seems reasonable to service all the requests close to the current head position before moving the head far away to service other requests. This assumption is the basis for the shortest-seek-time-first (SSTF) algorithm.
- The SSTF algorithm selects the request with the minimum seek time from the current head position.
- Since seek time increases with the number of cylinders traversed by the head, SSTF chooses the pending request closest to the current head position.

SCAN Scheduling:

- In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
- At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk.
- The SCAN algorithm is sometimes called the elevator algorithm, since the disk arms behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.

C-SCAN Scheduling:

- Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time.
- Like SCAN, CSCAN moves the head from one end of the disk to the other, servicing requests along the way.
- When the head reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.

LOOK Scheduling:

- As we described them, both SCAN and C-SCAN move the disk arm across the full width of the disk.
- In practice, neither algorithm is often implemented this way. More commonly, the arm goes only as far as the final request in each direction.
- Then, it reverses direction immediately, without going all the way to the end of the disk.
- Versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling, because they look for a request before continuing to move in a given direction.

C-LOOK Scheduling:

- This is just an enhanced version of C-SCAN. In this the scanning doesn't go past the last request in the direction that it is moving. It too jumps to the other end but not all the way to the end. Just to the furthest request.

Set A

1. Write an OS program to implement FCFS algorithm.
2. Write an OS program to implement SSTF algorithm.

Set B

1. Write an OS program to implement SCAN algorithm.
2. Write an OD program to implement S-SCAN algorithm.

Set C

1. Write an OS program to implement LOOK algorithm.
2. Write an OS program to implement C-LOOK algorithm.

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor

Date of Completion

SECTION II

CORE JAVA

SESSION 1

Java Tools

Reading

You should read the following topics before starting this exercise

1. Creating, compiling and running a java program.
2. The java virtual machine.
3. Java tools like javac, java, javadoc, javap and jdb.
4. Java keywords Syntax of class.

Ready Reference

Java Tools

- javac:-javac is the java compiler which compiles .java file into .class file(i.e. bytecode). If the program has syntax errors, javac reports them. If the program is error-free, the output of this command is one or more .class files.

  Syntax: javac fileName.java

- java:- This command starts Java runtime environment, loads the specified .class file and executes the main method.

  Syntax: java fileName

- javadoc:-javadoc is a utility for generating HTML documentation directly from comments written in Java source code.Javadoc comments have a special form but seems like an ordinary multiline comment to the compiler.

  Syntax of the comment:

  /** A sample doc comment */

  Syntax: javadoc [options] [packagenames] [sourcefiles] [@files]

  Where,

  - packagenames: A series of names of packages, separated by spaces
  - sourcefiles: A series of source file names, separated by spaces
  - @files: One or more files that contain packagenames and sourcefiles in any order, one name per line. Javadoc creates the HTML documentation on the basis of the javadoc tags used in the source code files.

- jdb: - jdb helps you find and fix bugs in Java language programs. This debugger has limited functionality.
  Syntax: jdb [options] [class] [arguments]

  - options : Command-line options.
  - class : Name of the class to begin debugging.
  - arguments : Arguments passed to the main() method of class.

After starting the debugger, the jdb commands can be executed. The important jdb commands are:

1. help, or?: The most important jdb command, help displays the list of recognized commands with a brief description.
2. run: After starting jdb, and setting any necessary breakpoints, you can use this command to start the execution the debugged application.
3. cont: Continues execution of the debugged application after a breakpoint, exception, or step.
4. print: Displays Java objects and primitive values. For variables or fields of primitive types, the actual value is printed. For objects, a short description is printed.

   Examples: print MyClass.myStaticField

   print myObj.myInstanceField

   print i + j + kprint myObj.myMethod()//if myMethod returns non-null .

5. dump: For primitive values, this command is identical to print. For objects, it prints the current value of each field defined in the object. Static and instance fields are included.
6. next: The next command advances execution to the next line in the current stack frame.
7. step: The step commands advances execution to the next line whether it is in the current stack frame or a called method. Breakpoints can be set in jdb at line numbers, constructors, beginning of a method.

- javap: - The javap tool allows you to query any class and find out its list of methods and constants.
  javap [options] class

Setting CLASSPATH

The classpath is the path that the Java runtime environment searches for classes and other resource files. The class path can be set using either the –classpath option or by setting the CLASSPATH environment variable.

The -classpath option is preferred because you can set it individually for each application without affecting other applications and without other applications modifying its value. The default value of the class path is ".", meaning that only the current directory is searched. Specifying either the CLASSPATH variable or the -cp scommand line switch overrides this value.

javac –classpath \myProg\myPackage; \myProg\otherclasses

Or

CLASSPATH= classpath1;classpath2...

export CLASSPATH

Lab Assignments

SET A

1. Using javap, view the methods of the following classes from the lang package: Object, String and Math. Type the following command and view the bytecodes. javap –c MyClass

2. Execute it using the following command. This gives a list of the classes loaded by the JVM. java –verbose MyClass

SET B

1. Define a class MyMaths having two private int data members. Write a default constructor to initialize it to 0 and another constructor to initialize it to a value (Use this). Write methods additionIS, subtractionIs , divisionIs, and multiplicationIs. Create an object in main. Use command line arguments to pass a value to the object (Hint : convert string argument) to integer) and perform the above tests. Provide javadoc comments for all constructors and methods and generate the html help file.

2. Save program (MyClass.java with a command line argument) in a folder named javaprgs. Set the CLASSPATH to this folder. Compile the program and use jdb to trace the program execution. Type the following commands and see the execution. jdb MyClass 10
    a. help
    b. stop in MyClass.main
    c. run
    d. next
    e. next
    f. Continue typing next till the program ends. Repeat the above process and type command step instead of next. Observe the output.

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 2

Array of Objects, Packages

Reading

You should read the following topics before starting this exercise:

1. Structure of a class in java.
2. Declaring class reference.
3. Creating an object using new.
4. Declaring an array of references.
5. Creating an array of objects.
6. Syntax of the package command.
7. Syntax of the import command.

Ready Reference

- Static fields and methods:
  Static fields are class variables which have the "static" modifier. They don't belong to the instance but belong to the class. Static methods are used to access static members of a class. To access a static member, use the following syntax:

  ClassName.staticMember

  ClassName.staticMethod(arguments)

- Creating objects:
  ClassName referenceName;

  referenceName = new ClassName();

  OR

  ClassName referenceName = new ClassName();

- Overriding toString method of the Object class:
  The toString method gives a string representation of an object. To over-ride the

  toString method for a user defined class, use the syntax:

  public String toString()

  {

  // return a string representation of the object

  }

- Declaring an array of references:
  ClassName[] arrayName = new ClassName[size];

- Creating an array of objects:
  for each reference in the array

  { Create an object using new

  }

  Example:

  Student[] studentArray = new Student[10];

  for(i=0; i<10; i++)

  studentArray[i] = new Student();

- Command line arguments :
  We can pass information to main from the command line using command line arguments. These are stored in an array of Strings which is passed to main as an argument.

  public static void main(String[] args)

  Here, args is the name of the array. The total number of arguments can be obtained using args.length . To access each argument, use a for loop as shown:

  for(int i=0; i<args.length ; i++)

  System.out.println("Argument "+ i + " = " + args[i]);

  To convert the argument from String to any type, use Wrapper classes.

- Simple I/O :
  To read a String from the console, use the following code:

  InputStreamReader isr = new InputStreamReader(System.in);

  BufferedReader br = new BufferedReader(isr);

  Or

  BufferedReader br = new BufferedReader(new

  InputStreamReader(System.in));

For this, you will have write the following statement at the beginning:

import java.io.*;

- Packages:
  A package is a collection of related classes and interfaces. It provides a mechanism for compartmentalizing classes. The Java API is organized as a collection of several predefined packages. The java.lang package is the default package included in all java programs. The commonly used packages are:

  java.lang Language support classes such as Math, Thread, String

  java.util Utility classes such as LinkedList, Vector , Date.

  java.io Input/Output classes

  java.awt For graphical user interfaces and event handling.

  javax.swing For graphical user interfaces

  java.net For networking

  java.applet For creating applets.

- Creating a package:
  To create a user defined package, the package statement should be written in the source code file. This statement should be written as the first line of the program. Save the program in a directory of the same name as the package.

  package packageName;

- Accessing a package :
  To access classes from a package, use the import statement.

  import packageName.*; //imports all classes

  import packageName.className; //imports specified class

  Note that the package can have a hierarchy of subpackages. In that case, the package name should be qualified using its parent packages. Example: project.sourcecode.java

  Here, the package named project contains one subpackage named sourcecode which contains a subpackage named java.

- Access Rules :
  The access rules for members of a class are given in the table below.

Access to	Public	Protected	Default	Private
Same class	Yes	Yes	Yes	Yes
Class in same package	Yes	Yes	Yes	No
Subclass in any package	Yes	Yes	No	No
Non subclass in other package	Yes	No	No	No

Lab Assignments

SET A

1. Define a Employee class (name, position, salary). Define a default and parameterized constructor. Override the toString method. Keep a count objects created. Create objects using parameterized constructor and display the object count after each object is created. (Use static member and method). Also display the contents of each object.

2. Create a class MyDate with day, month and year as members. Write appropriate member functions. Create another class Employee, which has id, name, date of birth, date of joining and salary as members (use MyDate for date fields). Write appropriate constructor for the Employee which assigns values to the members. Accept the details as command line arguments and create a Employee object using the arguments. Display the payslip details in a proper format.

3. Define a class Student (name, roll_no, class and marks of 6 subjects). Create an array of n Student objects. Calculate the percentage of each student using a method per(). Define a static method “sortStudent” which sorts the array on the basis of percentage. Display the student details in sorted order.

SET B

1. Create a package named com. Define subpackages;
   transact: with class Transaction with static methods credit() and debit()

   loan: with class LoanAccount with method doTransaction() which calls Transaction class mehods.

Create one LoanAccount object in main to perform operations on it by accepting command line arguments.

2. Write a Java program to create a Package "MCA_I" which has a class McaIMarks (members – SemITotal, SemIITotal). Create another package MCA_II which has a class McaIIMarks (members – SemITotal, SemIITotal). Create n objects of Student class (having rollNumber, name, McaIMarks and McaIIMarks). Add the marks of McaI and McaII calculate the Grade ('A' for >= 70, 'B' for >= 60 'C' for >= 50 , Pass Class for > =40 else 'FAIL') and display the result of the student in proper format.

SET C

1. Create a package called 'ListPack' which contains the classes 'Node' , 'singLinkedList' , 'dablLinkedList' and 'cirLinkedList'. Write a program to create a linked list of 5 nodes in main (type will be define by user) and display the elements. The elements are passed as command line arguments.

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 3

Inheritance and Interfaces

Reading

You should read the following topics before starting this exercise:

1. Concept of inheritance.
2. Use of extends keyword.
3. Concept of abstract class.
4. Defining an interface.
5. Use of implements keyword.

Ready Reference

- Inheriting a class : The syntax to create a subclass is :
  class SubClassName extends SuperClassName

  { //class body

  }

  Example:

  class Manager extends Employee

  { //code }

- Types of Inheritance
  1. Single inheritance
  2. Multiple inheritance
  3. Multilevel inheritance
  4. Hierarchical inheritance
  5. Hybrid inheritance
- Access in subclass :
  The following members can be accessed in a subclass:
  - public or protected superclass members.
  - Members with no specifier if subclass is in same package.
- The “super” keyword :
  It is used for three purposes:
  - Invoking superclass constructor -super(arguments)
  - Accessing superclass members – super.member
  - Invoking superclass methods – super.method(arguments)

- Dynamic binding:
  When over-riding is used, the method call is resolved during run-time i.e. depending on the object type, the corresponding method will be invoked.

  Example:

  A ref;

  ref = new A();

  ref.method1(10); //calls method of class A

  ref = new B();

  ref.method1(20); //calls method of class B

- Overriding methods :
  Redefining superclass methods in a subclass is called overriding. The signature of the

  subclass method should be the same as the superclass method.

  class A {

  void method1(int num) {

  //code }

  }

  class B extends A{

  void method1(int x) {

  //code }

  }

- Abstract class :
  An abstract class is a class which cannot be instantiated. It is only used to create

  subclasses. A class which has abstract methods must be declared abstract. An abstract

  class can have data members, constructors, method definitions and method declarations.

abstract class ClassName

{

...

}

- Abstract method
  An abstract method is a method which has no definition. The definition is provided by the subclass.

  abstract returnType method(arguments);

- Interface
  An interface is a pure abstract class i.e. it has only abstract methods and final variables. An interface can be implemented by multiple classes.

  interface InterfaceName {

  //abstract methods

  //final variables

  }

- Cloning :
  Cloning creates an identical copy of an object. To clone an object of a class, the class

  must implement Cloneable interface. This is an empty interface (tagging or marker

  interface). To clone objects of a class, over-ride the clone() method of the Object class.

  class MyClass implements Cloneable {

  public Object clone() {

  MyClass cloned = super.clone();

  //clone members if required

  return cloned; } }

Lab Assignments

Set A

1. Create an abstract class Student with methods disp_roll_no and calc_total_marks. Derive three classes McaStudent (marks of 6 subject as members), MscStudentfrom(marks of 4 subject as members) and BcsStudent(marks of 8 subject as members) from it. Create set of n students in main. Calculate total and display roll number with total of each student.. (Use method overriding).

2. Define an interface "QueueOperations" which declares methods for a static queue. Define a class "MyQueue" which contains an array and front and rear as data members and implements the above interface. Initialize the queue using a constructor. Write a menu driven program to perform operations on a queue object.

3. Define a class Staff with memers id, name, DOB, joining_date and salary. Define class TeachingStaff with members subjects[], experience and extends Staff also define class NonTeachingStaff with members department, shift and extends Staff. now define array DepatStaff with members from TeachingStaff and NonTeachingStaff as per requirement . Display details of all objects.

SET B

1. Define a class "Student" which has members Roll_no, name, date of birth. Define another class "CR" which has members class name and date of selection and extends Student. Create n objects of the CR class and clone them. (Use the Cloneable interface)

2. Define an abstract class "car" with members reg_no, model, reg_date. Define two subclasses of this class – "transportVehicles " (validity_no, start_date, period) and "privateVehicle " (owner_name, owner_address). Define appropriate constructors. Create n objects which could be of either transportVehicles or privateVehicle class by asking the user's choice. Display details of all "privateVehicle" objects and all "transportVehicles" objects.

SET C

1. Create an interface "CreditCardInterface" with methods to viewCreditAmount, viewPin, changePin, useCard and payBalance. Create a class Customer (name, card number, pin, creditAmount – initialized to 0). Implement methods viewCreditAmount, viewPin, changePin and payBalance of the interface. From Customer, create classes RegularCardHolder (maxCreditLimit) and GoldCardHolder (String specialPrivileges) and define the remaining methods of the interface Create n objects of the RegularCardHolder and GoldCardHolder classes and write a menu driven program to perform the following actions
   1. Use Card
   2. Pay Balance
   3. Change Pin

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 4

Exception Handling and assertion

Reading

You should read the following topics before starting this exercise:

1. Concept of Exception
2. Exception class hierarchy.
3. Use of try, catch, throw, throws and finally keywords
4. Defining user defined exception classes
5. Assertions

Ready Reference

Exception: An exception is an abnormal condition that arises in a code at run time. When an exception occurs,

1. An object representing that exception is created.
2. The method may handle the exception itself.
3. If the method cannot handle the exception, it “throws” this exception object to the method which called it.
4. The exception is “caught” and processed by some method or finally by the default java exception handler.

Predefined Exception classes

Java provides a hierarchy of Exception classes which represent an exception type.

- Throwable
- Error Exception
- RuntimeException
- Unchecked
- errors
- Unchecked
- exceptions
- Checked
- exceptions

Exception handling keywords

Exception handling in java is managed using 5 keywords: try , catch , throw, throws, finally

```
Syntax:
try {
// code that may cause an exception
}
catch (ExceptionType1 object) {
 // handle the exception
}
catch (ExceptionType2 object) {
// handle the exception
}
finally {
// this code is always executed
}
```

Note: try-catch blocks can be nested.

throw keyword:

The throw keyword is used to throw an exception object or to rethrow an exception.

```
throw exceptionObject;
```

Example:

```
catch(NumberFormatException e) {
System.out.println("Caught and rethrown" ) ;
throw e;
}
```

We can explicitly create an exception object and throw it. For example:

```
throw new NumberFormatException();
```

throws keyword

If the method cannot handle the exception, it must declare a list of exceptions it may cause. This list is specified using the throws keyword in the method header. All checked exceptions muct be caught or declared.

Syntax:

```
returnType methodName(arguments) throws ExceptionType1
[,ExceptionType2...]
{
//method body
}
```

Exception Types:

There are two types of Exceptions, Checked exceptions and Unchecked exceptions. Checked exceptions must be caught or rethrown. Unchecked exceptions do not have to be caught.

- IllegalMonitorStateException : Illegal monitor operation, such as waiting on an unlocked thread.
- IllegalStateException : Environment or application is in incorrect state.
- IllegalThreadStateException : Requested operation not compatible with current thread state.
- IndexOutOfBoundsException: Some type of index is out-of-bounds.
- NegativeArraySizeException: Array created with a negative size.
- NullPointerException : Invalid use of a null reference.
- NumberFormatException : Invalid conversion of a string to a numeric format.
- SecurityException : Attempt to violate security.
- StringIndexOutOfBounds : Attempt to index outside the bounds of a string.
- UnsupportedOperationException : An unsupported operation was encountered.

Checked Exceptions:

- ClassNotFoundException : Class not found.
- CloneNotSupportedException : Attempt to clone an object that does not implement the Cloneable interface.
- IllegalAccessException : Access to a class is denied.
- InstantiationException : Attempt to create an object of an abstract class or interface.

- InterruptedException : One thread has been interrupted by another thread.
- NoSuchFieldException : A requested field does not exist.
- NoSuchMethodException : A requested method does not exist.

User Defined Exceptions:

A user defined exception class can be created by extending the Exception class.

```
class UserDefinedException extends Exception {

//code

}
```

When that exception situation occurs, an object of this exception class can be created and thrown. For example, if we are accepting an integer whose valid values are only positive, then we can throw an "InvalidNumberException" for any negative value entered.

```
class NegativeNumberException extends Exception {

NegativeNumberException(int n){

System.out.println("Negative input "+ n);

}

}
```

Unchecked Exceptions:

- ArithmeticException : Arithmetic error, such as divide-by-zero.
- ArrayIndexOutOfBoundsException : Array index is out-of-bounds.
- ArrayStoreException : Assignment to an array element of an incompatible type.
- ClassCastException : Invalid cast.
- IllegalArgumentException : Illegal argument used to invoke a method.

Assertions:

An assertion is a statement containing a boolean expression that is assumed to be true when the statement is executed. The system reports an AssertionError if the expression evaluates to false. For example, if you write a method that measures the current speed of a vehicle, you might want to ensure that the speed is less than the maximum vehicle speed.

Assertions can be in two forms:

assert Expression1;

assert Expression1 : Expression2;

where : Expression11 is a boolean expression, Expression2 is an expression that has a value.

Example: assert i > 0;

assert i % 3 == 2 : i;

Enabling and Disabling Assertions:

By default, assertions are disabled. You enable them by running the program with the enableassertionsor -eaoption. To disable assertions, use the –daor –disableassertions flag.

Examples:

java -enableassertions MyClass

java -ea:MyClass -ea:Pack1.MyClass2 MyApp

java –ea:MyClass –da:MyClass3 MyApp

Lab Assignments

SET A

1. Create a class Doctor with attributes id, name, age and department. Initialize values through parameterized constructor. If age of Doctor is not in between 25 and 65 then generate user-defined exception “AgeNotWithinRangeException”. If name contains numbers or special symbols raise exception “NameNotValidException”. Define the two exception classes.

2. A program accepts two numbers as command line arguments. It displays all odd numbers in between . Using assertions, validate the input for the following criteria: Both should be positive integers. The second should be larger than the first.

3. Define Exceptions VowelException ,BlankException,ExitException. Write another class Test which reads a character from command line. If it is a vowel, throw VowelException,if it is blank throw BlankException and for a character 'X' throw an ExitException and terminate program. For any other character, display “Valid character”.

SET B

1. Define class EmailId with members, username, domainId and password. Define default and parameterized constructors. Accept values from the command line and create a date object. Throw user defined exceptions – "InvalidUsernameException" or "InvalidPasswordException" if the uaername and password are invalid. If the date is valid, display message "Valid Email Id".

2. Write a program which accept two integers and an arithmetic operator from the command line and performs the operation. Fire the following user defined exceptions:
    a. If the no of arguments are less than 3 then fire "IllegalNumberOfArguments"
    b. If the operator is not an Arithmetic operator, throw "InvalidOperatorException".
    c. If result is –ve, then throw "NegativeResultException"

SET C

1. A program accepts two integers as command line arguments. It displays all prime numbers between these two. Validate the input for the following criteria: Both should be positive integers. The second should be larger than the first. Create user defined exceptions for both.

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 5

I/O and File Handling

Reading

You should read the following topics before starting this exercise:

1. Strings
2. Concept of streams
3. Types of streams
4. Byte and Character stream classes.
5. The File class

Strings Class

The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

String Constructor

String( )
Initializes a newly created String object so that it represents an empty character sequence.

String(String strobj)
Creates a String using string object.

String(char[] arrchar)
Constructs a new String by decoding the specified array of characters.
Eg. String s= new String(arrchar);

String(byte[] bytes, Charset charset)
Constructs a new String by decoding the specified array of bytes using the specified charset.

Creating Strings

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
```

String str = new String(data);

String Methods

Methods	Description
char charAt(int index)	Returns the character at the specified index.
int compareTo(Object o)	Compares this String to another Object.
int compareTo(String anotherString)	Compares two strings lexicographically.
String concat(String str)	Concatenates the specified string to the end of this string.
boolean endsWith(String suffix)	Tests if this string ends with the specified suffix.
boolean equals(Object anObject)	Compares this string to the specified object.
int indexOf(int ch)	Returns the index within this string of the first occurrence of the specified character.
int lastIndexOf(int ch)	Returns the index within this string of the last occurrence of the specified character.
int length()	Returns the length of this string.
String replace(char oldChar, char newChar)	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
String[] split(String regex)	Splits this string around matches of the given regular expression.
boolean startsWith(String prefix)	Tests if this string starts with the specified prefix.
String substring(int beginIndex)	Returns a new string that is a substring of this string.
String substring(int beginIndex, int endIndex)	Returns a new string that is a substring of this string..
String toLowerCase()	Converts all of the characters in this String to lower case using the rules of the default locale.

String toString()	This object (which is already a string!) is itself returned.
String toUpperCase()	Converts all of the characters in this String to upper case using the rules of the default locale.
String trim()	Returns a copy of the string, with leading and trailing whitespace omitted.

static String valueOf(primitive data type x) Returns the string representation of the passed data type argument.

String Buffer class

The **java.lang.StringBuffer** class is a thread-safe, mutable sequence of characters.

Class constructors

StringBuffer()
This constructs a string buffer with no characters in it and an initial capacity of 16 characters.

StringBuffer(CharSequence seq)
This constructs a string buffer that contains the same characters as the specified CharSequence.

StringBuffer(int capacity)
This constructs a string buffer with no characters in it and the specified initial capacity.

StringBuffer(String strobj)
This constructs a string buffer initialized to the contents of the specified string.

Class methods

Methods	Description
StringBuffer append(char[] str)	This method appends the string representation of the char array argument to this sequence.
int capacity()	This method returns the current capacity.
StringBuffer delete(int start, int end)	This method removes the characters in a substring of this sequence.

StringBuffer deleteCharAt(int index) This method removes the char at the specified position in this sequence.

void ensureCapacity(int minimumCapacity) This method ensures that the capacity is at least equal to the specified minimum.

StringBuffer insert(int offset, boolean b) This method inserts the string representation of the boolean argument into this sequence.

StringBuffer insert(int offset, char c) This method inserts the string representation of the char argument into this sequence.

StringBuffer replace(int start, int end, String str) This method replaces the characters in a substring of this sequence with characters in the specified String.

StringBuffer reverse() This method causes this character sequence to be replaced by the reverse of the sequence.

String Tokenizer Class

The **java.util.StringTokenizer** class allows an application to break a string into tokens.

Class constructors

StringTokenizer(String str)
This constructor a string tokenizer for the specified string.

StringTokenizer(String str, String delim)
This constructor constructs string tokenizer for the specified string.

StringTokenizer(String str, String delim, boolean returnDelims)
This constructor constructs a string tokenizer for the specified string.

Class methods

Methods	Description
int countTokens()	This method calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.
boolean hasMoreElements()	This method returns the same value as the hasMoreTokens method.
boolean hasMoreTokens()	This method tests if there are more tokens available from this tokenizer's string.

Object nextElement()	This method returns the same value as the nextToken method, except that its declared return value is Object rather than String.
String nextToken()	This method returns the next token from this string tokenizer.

String nextToken(String delim) This method returns the next token in this string tokenizer's string.

Ready Reference

- java.io.File class

  This class supports a platform-independent definition of file and directory names. It also provides methods to list the files in a directory, to check the existence, readability, writeability, type, size, and modification time of files and directories, to make new directories, to rename files and directories, and to delete files and directories.

- Constructors:
  - public File(String path);
  - public File(String path, String name);
  - public File(File dir, String name);

  Example:

  File f1=new File("/home/java/a.txt");

- Methods
  1. boolean canRead()- Returns True if the file is readable.
  2. boolean canWrite()- Returns True if the file is writeable.
  3. String getName()- Returns the name of the File with any directory names omitted.
  4. boolean exists()- Returns true if file exists
  5. String getAbsolutePath()- Returns the complete filename. Otherwise, if the File is a relative file specification, it returns the relative filename appended to the current working directory.
  6. String getParent()- Returns the directory of the File. If the File is an absolute specification.
  7. String getPath()- Returns the full name of the file, including the directory name.
  8. boolean isDirectory()- Returns true if File Object is a directory
  9. boolean isFile()- Returns true if File Object is a file

10. long lastModified()- Returns the modification time of the file (which should be used for comparison with other file times only, and not interpreted as any particular time format).
11. long length()- Returns the length of the file.
12. boolean delete()- deletes a file or directory. Returns true after successful deletion of a file.
13. boolean mkdir ()- Creates a directory.
14. boolean renameTo (File dest)- Renames a file or directory. Returns true after successful renaming

- Directories :

  A directory is a File that contains a list of other files & directories. When you create a File object & it is a directory, the isDirectory() method will return true. In this case list method can be used to extract the list of other files & directories inside. The forms of list() method is-

  - public String[ ] list()
  - public String[ ] list(FilenameFilter filter)

- Streams :

  A stream is a sequence of bytes. When writing data to a stream, the stream is called an output stream. When reading data from a stream, the stream is called an input stream. If a stream has a buffer in memory, it is a buffered stream. Binary Streams contain binary data. Character Streams have character data and are used for storing and retrieving text. The two main types of Streams are ByteStream and CharacterStream.

  ByteStream CharacterStream

  InputStream OutputStream Reader Writer

  There are four top level abstract stream classes: InputStream, OutputStream, Reader, and Writer.

1. InputStream. -A stream to read binary data.
2. OutputStream-. A stream to write binary data.
3. Reader.- A stream to read characters.
4. Writer. -A stream to write characters.

- ByteStream Classes

a. InputStream Methods

1. int read ()- Returns an integer representation of next available byte of input.- is returned at the stream end.
2. int read (byte buffer[ ])- Read up to buffer.length bytes into buffer & returns actual number of bytes that are read. At the end returns –1.
3. int read(byte buffer[ ], int offset, int numbytes)- Attempts to read up to numbytes bytes into buffer starting at buffer[offset]. Returns actual number of bytes that are read. At the end returns –1.
4. void close()- to close the input stream
5. void mark(int numbytes)- places a mark at current point in input stream & remain valid till number of bytes are read.
6. void reset()- Resets pointer to previously set mark/ goes back to stream beginning.
7. long skip(long numbytes)- skips number of bytes.
8. int available()- Returns number of bytes currently available for reading.

b. OutputStream Methods

1. void close() - to close the OutputStream
2. void write (int b) - Writes a single byte to an output stream.
3. void write(byte buffer[ ]) - Writes a complete array of bytes to an output stream.
4. void write (byte buffer[ ], int offset, int numbytes) - Writes a sub range of numbytes bytes from the array buffer, beginning at buffer[offset].
5. void flush() - clears the buffer.

- The following table lists the Byte Stream classes
  - BufferedInputStream : Buffered input stream
  - BufferedOutputStream : Buffered output stream
  - ByteArrayInputStream : Input stream that reads from a byte array
  - ByteArrayOutputStream : Output stream that writes to a byte array
  - DataInputStream : An input stream that contains methods for reading the Java standard data types
  - DataOutputStream : An output stream that contains methods for writing the Java standard data types
  - FileInputStream : Input stream that reads from a file
  - FileOutputStream : Output stream that writes to a file
  - FilterInputStream : Implements InputStream
  - FilterOutputStream : Implements OutputStream
  - InputStream : Abstract class that describes stream input
  - OutputStream : Abstract class that describes stream output

- PipedInputStream : Input pipe
- PipedOutputStream : Output pipe
- PrintStream : Output stream that contains print( ) and println( )
- PushbackInputStream : Input stream that supports one-byte "unget," which returns a byte to the input stream
- RandomAccessFile : Supports random access file I/O
- SequenceInputStream : Input stream that is a combination of two or more input streams that will be read

- CharacterStream Classes

a. Reader:

Reader is an abstract class that defines Java's method of streaming character input. All methods in this class will throw an IOException.

Methods in this class-

1. int read ()- Returns an integer representation of next available character from invoking stream. -1 is returned at the stream end.
2. int read (char buffer[ ])- Read up to buffer.length chacters to buffer & returns actual number of characters that are successfully read. At the end returns –1.
3. int read(char buffer[ ], int offset, int numchars)- Attempts to read up to numchars into buffer starting at buffer[offset]. Returns actual number of characters that are read. At the end returns –1.
4. void close()- to close the input stream
5. void mark(int numchars)- places a mark at current point in input stream & remain valid till number of characters are read.
6. void reset()- Resets pointer to previously set mark/ goes back to stream beginning.
7. long skip(long numchars)- skips number of characters.
8. int available()- Returns number of bytes currently available for reading.

b. Writer:

Is an abstract class that defines streaming character output. All the methods in this class returns a void value & throws an IOException.
The methods are

1. void close() - to close the OutputStream
2. void write (int ch) - Writes a single character to an output stream.
3. void write(char buffer[ ]) - Writes a complete array of characters to an output stream.

4. void write (char buffer[ ], int offset, int numchars) - Writes a sub range of numchars from the array buffer, beginning at buffer[offset].
5. void write(String str)- Writes str to output stream.
6. void write(String str, int offset, int numchars)- Writes a subrange of numchars from string beginning at offset.
7. void flush() - clears the buffer.

c. The following table lists the Character Stream classes

1. BufferedReader : Buffered input character stream
2. BufferedWriter : Buffered output character stream
3. CharArrayReader : Input stream that reads from a character array
4. CharArrayWriter : Output stream that writes to a character array
5. FileReader : Input stream that reads from a file
6. FileWriter : Output stream that writes to a file
7. FilterReader : Filtered reader
8. FilterWriter : Filtered writer
9. InputStreamReader : Input stream that translates bytes to characters
10. LineNumberReader : Input stream that counts lines
11. OutputStreamWriter : Output stream that translates characters to bytes
12. PipedReader : Input pipe
13. PipedWriter : Output pipe
14. PrintWriter : Output stream that contains print( ) and println( )
15. PushbackReader : Input stream that allows characters to be returned to the input stream

d. Reader:

Abstract class that describes character stream input .

StringReader : Input stream that reads from a string

StringWriter :Output stream that writes to a string

Writer : Abstract class that describes character stream output

e. RandomAccessFile :

Random access files permit nonsequential, or random, access to a file's contents. To access a file randomly, you open the file, seek a particular location, and read from or write to that file. When opening a file using a

RandomAccessFile, you can choose whether to open it read-only or read write.

RandomAccessFile (File file, String mode) throws FileNotFoundException

RandomAccessFile (String filePath, String mode) throws FileNotFoundException

The value of mode can be one of these:

"r" Open for reading only.

"rw" Open for reading and writing.

Methods:

1. position– Returns the current position
2. position(long)– Sets the position
3. read(ByteBuffer) – Reads bytes into the buffer from the stream
4. write(ByteBffer) – Writes bytes from the buffer to the stream
5. truncate(long) – Truncates the file (or other entity) connected to the stream

Lab Assignments

SET A

1. Write a Java program to perform all string operation given in ready reference using String class and StringBuffer class.

2. Write a program to accept a string as command line argument and check whether it is a file or directory. If it is a directory, list the contents of the directory, count how many files the directory has and delete all files in that directory having extension .txt. (Ask the user if the files have to be deleted). If it is a file, display all information about the file (path, size, attributes etc).

3. Write a java program to accept two file names as command line arguments and copy the contains of first to second in such a manner the case of all alphabet is changed and digits are replaced by '*'. Display appropriate error message if the first file does not exist. (Use methods from Character class)

4. Write a program to display the contents of a file in the reverse order.

SET B

1. Write a program to store item information (id, name, price, qty) in file "item.dat". Write a menu driven program to perform the following operations: i. Search for a specific item by name. ii. Find costliest item. iii. Display all items and total cost

2. Write a program to store student information (roll number, name, percentage) in a RandomAccessFile “student.dat”. Display the details of the student having a specific roll number.

3. Write a Java program to accept an option, string and file name from user. Perform following operations:

1. If no option is passed then print all lines in the file containing the string.
2. If the option passed is –c then print the count of lines containing the string.
3. If the option passed is –v then print the lines not containing the string.

SET C

1. Write a menu driven program to perform the following operations on a binary file “item.dat” which contains id, name, price and quantity.

i. Add an item
ii. Search for an item.
iii. Delete an item
iv. Modify details of an item.
v. Display all items.

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor

Date of Completion

SESSION 6

GUI using Swing

Reading

You should read the following topics before starting this exercise

1. AWT and Swing concepts.
2. Layout managers in java
3. Containers and Components
4. Adding components to containers

Ready References

Swing Classes:

The following table lists some important Swing classes and their description.

Class	Description
Box	Container that uses a BoxLayout
JApplet	Base class for Swing applets
JButton	Selectable component that supports text/image display
JCheckBox	Selectable component that displays state to user
JCheckBoxMenuItem	Selectable component for a menu; displays state to user
JComboBox	For selecting from a drop-down list of choices
JColorChooser	For selecting colors
JComponent	Base class for Swing components
JDesktopPane	Container for internal frames
JDialog	Base class for pop-up subwindows
JEditorPane	For editing and display of formatted content
JFileChooser	For selecting files and directories
JFormattedTextField	For editing and display of a single line of formatted text
JFrame	Base class for top-level windows
JInternalFrame	Base class for top-level internal windows
JLabel	For displaying text/images
JLayeredPane	Container that supports overlapping components
JList	For selecting from a scrollable list of choices
JMenu	Selectable component for holding menu items; supports text/image display
JMenuBar	For holding menus
JMenuItem	Selectable component that supports text/image display
JOptionPane	For creating pop-up messages
JPanel	Basic component container
JPasswordField	For editing and display of a password

JPopupMenu	For holding menu items and popping up over components
JProgressBar	For showing the progress of an operation to the user
JRadioButton	Selectable component that displays state to user; included in Button Group toensure that only one button is selected
JRadioButtonMenuItem	Selectable component for menus; displays state to user; included in ButtonGroup to ensure that only one button is selected
JRootPane	Inner container used by JFrame, JApplet, and others
JScrollBar	For control of a scrollable area
JScrollPane	To provide scrolling support to another component
JSeparator	For placing a separator line on a menu or toolbar
JSlider	For selection from a numeric range of values
JSpinner	For selection from a set of values, from a list, a numeric range, or a date range
JSplitPane	Container allowing the user to select the amount of space for each of two components
JTabbedPane	Container allowing for multiple other containers to be displayed; each container appears on a tab
JTable	For display of tabular data
JTextArea	For editing and display of single-attributed textual content
JTextField	For editing and display of single-attributed textual content on a single line
JTextPane	For editing and display of multi-attributed textual content
JToggleButton	Selectable component that supports text/image display; selection triggers component to stay “in”
JToolBar	Draggable container
JToolTip	Internally used for displaying tool tips above components
JTree	For display of hierarchical data
JViewport	Container for holding a component too big for its display area
JWindow	Base class for pop-up windows

Layout Manager

The job of a layout manager is to arrange components on a container. A layout manager is an object of any class that implements the **LayoutManager** interface.
Each container has a layout manager associated with it. To change the layout manager for a container, use the **setLayout()** method.

Syntax
setLayout(LayoutManager obj)

The predefined managers are listed below:

1. FlowLayout	2.BorderLayout	3.GridLayout
4. BoxLayout	5.CardLayout	6.GridBagLayout

Examples:

JPanel p1 = new JPanel()

p1.setLayout(new FlowLayout());
p1.setLayout(new BorderLayout());
p1.setLayout(new GridLayout(3,4));

Important Containers:

1. JFrame – This is a top-level container which can hold components and containers like panels.

Constructors

JFrame()
JFrame(String title)

Important Methods

setSize(int width, int height)	-Specifies size of the frame in pixels
setLocation(int x, int y)	-Specifies upper left corner
setVisible(boolean visible)	-Set true to display the frame
setTitle(String title)	-Sets the frame title
setDefaultCloseOperation(int mode)	-Specifies the operation when frame is closed. The modes are: JFrame.EXIT_ON_CLOSE JFrame.DO_NOTHING_ON_CLOSE JFrame.HIDE_ON_CLOSE JFrame.DISPOSE_ON_CLOSE
pack()	-Sets frame size to minimum size required to hold components

2. JPanel – This is a middle-level container which can hold components and can be added to other containers like frame and panels.

Constructors

public javax.swing.JPanel(java.awt.LayoutManager, boolean);
public javax.swing.JPanel(java.awt.LayoutManager);
public javax.swing.JPanel(boolean);
public javax.swing.JPanel();

Important Components:

1. Label
With the JLabel class, you can display unselectable text and images.

Constructors-

JLabel(Icon i) JLabel(Icon I , int n)

JLabel(String s) JLabel(String s, Icon i, int n)
JLabel(String s, int n) JLabel()

The int argument specifies the horizontal alignment of the label's contents within its drawing area; defined in the SwingConstants interface (which JLabel implements): LEFT (default), CENTER, RIGHT, LEADING, or TRAILING.

Methods

1. Set or get the text displayed by the label.
   void setText(String) String getText()

2. Set or get the image displayed by the label.
   void setIcon (Icon) Icon getIcon()

3. Set or get the image displayed by the label when it's disabled. If you don't specify a disabled image, then the look-and-feel creates one by manipulating the default image.
   void setDisabledIcon(Icon) Icon getDisabledIcon()

4. Set or get where in the label its contents should be placed. For vertical alignment: TOP, CENTER (the default), and BOTTOM.
   void setHorizontalAlignment(int) void setVerticalAlignment(int)
   int getHorizontalAlignment() int getVerticalAlignment()

2. Button

A Swing button can display both text and an image. The underlined letter in each button's text shows the *mnemonic* which is the keyboard alternative.

Constructors-
JButton(Icon I)
JButton(String s)
JButton(String s, Icon I)
Methods
void setDisabledIcon(Icon) void setPressedIcon(Icon)
void setSelectedIcon(Icon) void setRolloverIcon(Icon)
String getText() void setText(String)
Event- ActionEvent

3. Check box

Class- JCheckBox

Constructors-
JCheckBox(Icon i) JCheckBox(Icon i,boolean state)
JCheckBox(String s) JCheckBox(String s, boolean state)

JCheckBox(String s, Icon i) JCheckBox(String s, Icon I, boolean state)

Methods

void setSelected(boolean state) String getText()
void setText(String s)

Event- ItemEvent

4. Radio Button

Class- JRadioButton
Constructors

JRadioButton (String s) JRadioButton(String s, boolean state)
JRadioButton(Icon i) JRadioButton(Icon i, boolean state)
JRadioButton(String s, Icon i) -JRadioButton(String s, Icon i, boolean state)
JRadioButton() ”

To create a button group- ButtonGroup()

Adds a button to the group, or removes a button from the group.
void add(AbstractButton) void remove(AbstractButton)

5. Combo Box

Class- JComboBox

Constructors- JComboBox()

Methods
void addItem(Object) Object getItemAt(int)
Object getSelectedItem() int getItemCount()

Event- ItemEvent

6. List

Constructor- JList(ListModel)

List models-

1. SINGLE_SELECTION - Only one item can be selected at a time. When the user selects an item, any previously selected item is deselected first.
2. SINGLE_INTERVAL_SELECTION- Multiple, contiguous items can be selected. When the user begins a new selection range, any previously selected items are

deselected first.

3. MULTIPLE_INTERVAL_SELECTION- The default. Any combination of items can be selected. The user must explicitly deselect items.

Methods

boolean isSelectedIndex(int) void setSelectedIndex(int)
void setSelectedIndices(int[]) void setSelectedValue(Object, boolean)
void setSelectedInterval(int, int) int getSelectedIndex()
int getMinSelectionIndex() int getMaxSelectionIndex()
int[] getSelectedIndices() Object getSelectedValue()
Object[] getSelectedValues()

Event- ActionEvent

7. Text classes

All text related classes are inherited from JTextComponent class

a. JTextField

Creates a text field. The int argument specifies the desired width in columns. The String argument contains the field's initial text. The Document argument provides a custom document for the field.

Constructor

JTextField() JTextField(String)
JTextField(String, int) JTextField(int)
JTextField(Document, String, int)

b. JPasswordField

Creates a password field. When present, the int argument specifies the desired width in columns. The String argument contains the field's initial text. The Document argument provides a custom document for the field.

Constructors-

JPasswordField() JPasswordField(String)
JPasswordField(String, int) JPasswordField(int)
JPasswordField(Document, String, int)

Methods

1. Set or get the text displayed by the text field.
   void setText(String) String getText()
2. Set or get the text displayed by the text field.
   char[] getPassword()
3. Set or get whether the user can edit the text in the text field.
   void setEditable(boolean) boolean isEditable()

4. Set or get the number of columns displayed by the text field. This is really just a hint for computing the field's preferred width.
   void setColumns(int); int getColumns()
5. Get the width of the text field's columns. This value is established implicitly by the font.
   int getColumnWidth()
6. Set or get the echo character i.e. the character displayed instead of the actual characters typed by the user.
   void setEchoChar(char) char getEchoChar()

Event- ActionEvent

c. JTextArea
Represents a text area which can hold multiple lines of text

Constructors-

JTextArea (int row, int cols)
JTextArea (String s, int row, int cols)

Methods

void setColumns (int cols) void setRows (int rows)
void append(String s) void setLineWrap (boolean)

8. Dialog Boxes

Types

1. Modal- wont let the user interact with the remaining windows of application until first deals with it. Ex- when user wants to read a file, user must specify file name before prg. can begin read operation.

2. Modeless dialog box- Lets the user enters information in both, the dialog box & remainder of application ex- toolbar.
   Swing has a JOptionPane class, that lets you put a simple dialog box.

Methods in JOption Class

1. static void showMessageDialog()- Shows a message with ok button.
2. static int showConfirmDialog()- shows a message & gets users options from set of options.

3. static int showOptionDialog- shows a message & get users options from set of options.
4. String showInputDialog()- shows a message with one line of user input.

9. Menu

Creating and Setting Up Menu Bars

Constructor or Method	**Purpose**
JMenuBar()	-Creates a menu bar.
JMenu add(JMenu)	-Creates a menu bar.
void setJMenuBar(JMenuBar)	- Sets or gets the menu bar of an applet, dialog, frame, internal frame, or root pane.
JMenuBar getJMenuBar()	

Creating and Populating Menus

JMenu()	-Creates a menu. The string specifies the text to display for the menu.
JMenu(String)	-
JMenuItem add(JMenuItem)	- Adds a menu item to the current end of the menu. If the argument is an Action object, then the menu creates a menu item. If the argument is a string, then the menu automatically creates a JMenuItem object that displays the specified text.
JMenuItem add(Action)	-
JMenuItem add(String)	-
void addSeparator()	Adds a separator to the current end of the menu.
JMenuItem insert(Action, int)	- Inserts a menu item or separator into the menu at the specified position. The first menu item is at position 0, the second at position 1, and so on. The JMenuItem, Action, and String arguments are treated the same as in the corresponding add methods.
void insert(String, int)	”
void insertSeparator(int)	”
JMenuItem insert(JMenuItem, int)	”
void remove(JMenuItem)	-Removes the specified item(s) from the menu.
If the	argument is an integer, then it specifies the position of the menu item to be removed.

void remove(int)	”
void removeAll()	”

Implementing Menu Items

JMenuItem()	-Creates an ordinary menu item. The icon argument, if present, specifies the icon that the menu item should display. Similarly, the string argument specifies the text that the menu item should display. The integer argument specifies the keyboard mnemonic to use. You can specify any of the relevant VK constants defined in the KeyEvent class. For example, to specify the A key, use KeyEvent.VK_A.
JMenuItem(String)	”
JMenuItem(Icon)	”
JMenuItem(String, Icon)	”
JMenuItem(String, int)	”
JCheckBoxMenuItem()	-Creates a menu item that looks and acts like a check box. The string argument, if any, specifies the text that the menu item should display. If you specify true for the boolean argument, then the menu item is initially selected (checked). Otherwise
JCheckBoxMenuItem(String)	”
JCheckBoxMenuItem(Icon)	”
JCheckBoxMenuItem(String, Icon)	”
JCheckBoxMenuItem(String, boolean)	”
JCheckBoxMenuItem(String, Icon, boolean)	”
JRadioButtonMenuItem()	-Creates a menu item that looks and acts like a radio button. The string argument, if any, specifies the text that the menu item should display. If you specify true for the boolean argument, then the menu item is initially selected. Otherwise, the menu item is initially unselected.
JRadioButtonMenuItem(String)	”
JRadioButtonMenuItem(Icon)	”
JRadioButtonMenuItem(String, Icon)	”
JRadioButtonMenuItem(String, boolean)	”

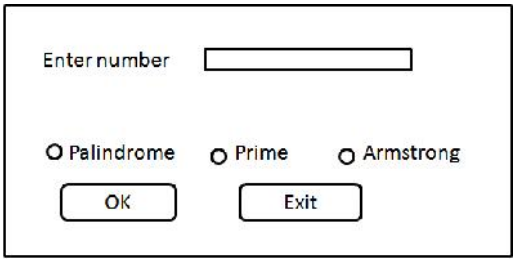
JRadioButtonMenuItem(Icon, boolean)	”
JRadioButtonMenuItem(String, Icon, boolean)	”
void setState(boolean)	- Set or get the selection state of a check box menu item.
boolean getState()	”
void setEnabled(boolean)	-If the argument is true, enable the menu item. Otherwise, disable the menu item.

GUI Using Swing

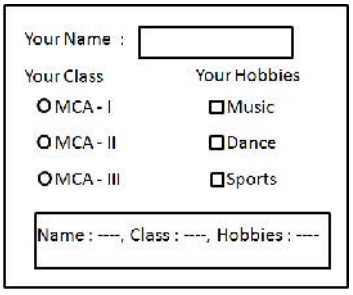
Lab Assignments

SET A

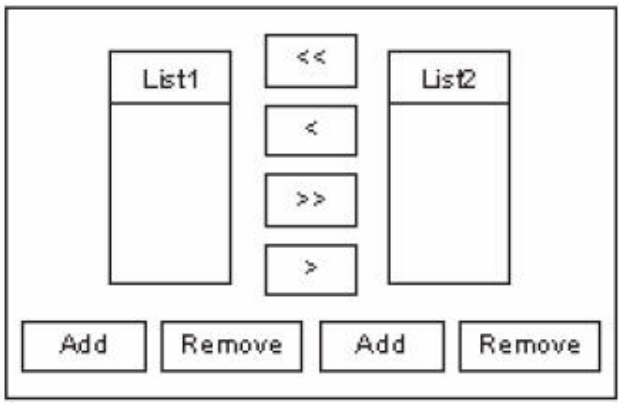
1. Write a java program to create the following GUI screen using appropriate layout managers.



2. Write a java program to create the following GUI screen using appropriate layout managers.

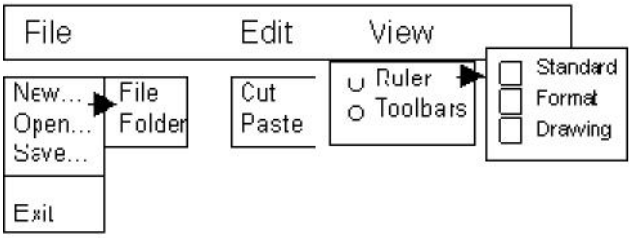


3. Write a java program to create the following GUI screen using appropriate layout managers.

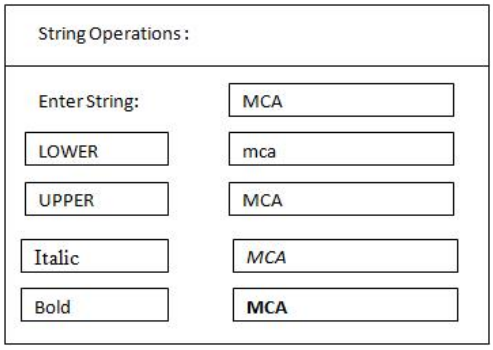


SET B

1. Write a program to display the following menus and sub-menus.

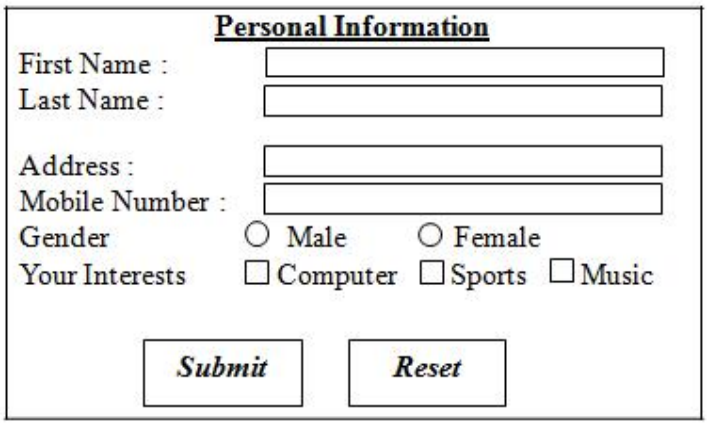


2. Write a program to create the following GUI.



SET C

1. Write a program to create the following GUI.



Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor

Date of Completion

SESSION 7

Event Handling

Reading

You should read the following topics before starting this exercise
1. Delegation event model
2. Event sources, Event listeners, Event classes, Adapter classes.
3. Anonymous class.

Ready Reference

Java has two types of events:
1. **Low-Level Events:** Low-level events represent direct communication from user. A low level event is a key press or a key release, a mouse click, drag, move or release, and so on. Following are low level events.

Event	**Description**
ComponentEvent	Indicates that a component object (e.g. Button, List, TextField) is moved, resized, rendered invisible or made visible again.
FocusEvent	Indicates that a component has gained or lost the input focus.
KeyEvent	Generated by a component object (such as TextField) when a key is pressed,released or typed.
MouseEvent	Indicates that a mouse action occurred in a component. E.g. mouse is pressed,releases, clicked (pressed and released), moved or dragged.
ContainerEvent	Indicates that a container's contents are changed because a component was added or removed.
WindowEvent	Indicates that a window has changed its status. This low level event is generated by a Window object when it is opened, closed, activated, deactivated, iconified, deiconified or when focus is transferred into or out of the Window.

2. High-Level Events: High-level (also called as semantic events) events encapsulate the meaning of a user interface component. These include following events.

Event	**Description**
ActionEvent	Indicates that a component-defined action occurred. This high-level event is generated by a component (such as Button) when the component-specific action occurs (such as being pressed).

AdjustmentEvent	The adjustment event is emitted by Adjustable objects like scrollbars.
ItemEvent	Indicates that an item was selected or deselected. This high-level event is generated by an ItemSelectable object (such as a List) when an item is selected or deselected by the user.
TextEvent	Indicates that an object's text changed. This high-level event is generated by an object (such as TextComponent) when its text changes.

The following table lists the events, their corresponding listeners and the method to add the listener to the component.

Event	**Event Source**	**Event Listener**	**Method to add listener to event source**
Low-level Events			
ComponentEvent	Component	ComponentListener	addComponentListener()
FocusEvent	Component	FocusListener	addFocusListener()
KeyEvent	Component	KeyListener	addKeyListener()
MouseEvent	Component	MouseListener MouseMotionListener	addMouseListener() addMouseMotionListener()
ContainerEvent WindowEvent	Container Window	ContainerListener WindowListener	addContainerListener() addWindowListener()
High-level Events			
ActionEvent	Button List MenuItem TextField	ActionListener	addActionListener()
AdjustmentEvent	Scrollbar	AdjustmentListener	addAdjustmentListener()

TextEvent TextField	TextArea	TextListener	addTextLIstener()

Listener Methods:

Methods	**Description**
ComponentListener	
componentResized(ComponentEvent e)	Invoked when component's size changes.
componentMoved(ComponentEvent e)	Invoked when component's position changes.
componentShown(ComponentEvent e)	Invoked when component has been made visible.
componentHidden(ComponentEvent e**)**	Invoked when component has been made invisible.
FocusListener	
focusGained(FocusEvent e)	Invoked when component gains the keyboard focus.
focusLost(FocusEvent e)	Invoked when component loses the keyboard focus.
KeyListener	
keyTyped(KeyEvent e) .	Invoked when a key is typed.
keyPressed(KeyEvent e)	Invoked when a key is pressed.
keyReleased(KeyEvent e)	Invoked when a key is released
MouseListener	
mouseClicked(MouseEvent e)	Invoked when a mouse button is clicked (i.e.

.	pressed and released) on a component.
mousePressed(MouseEvent e)	Invoked when a mouse button is pressed on a component.
mouseReleased(MouseEvent e)	Invoked when a mouse button is released on a component.
mouseEntered(MouseEvent e)	Invoked when a mouse enters a component
mouseExited(MouseEvent e)	Invoked when a mouse exits a component.
MouseMotionListener	
mouseDragged(MouseEvent e)	Invoked when a mouse button is pressed on a component and then dragged.
mouseMoved(MouseEvent e)	Invoked when a the mouse cursor is moved on to a component but mouse button is not pressed.
ContainerListener	
componentAdded(ContainerEvent e)	Invoked when a component is added to the container.
componentRemoved(ContainerEvent e)	Invoked when a component is removed from the container.
WindowListener	
windowOpened(WindowEvent e)	Invoked the first time a window is made visible
windowClosing(WindowEvent e)	Invoked when the user attempts to close the window from the window's system menu.
windowClosed(WindowEvent e)	Invoked when a window has been closed as the

	result of calling dispose on the window.
windowIconified(WindowEvent e)	Invoked when a window is changed from a normal to a minimized state.
windowDeiconified(WindowEvent e)	Invoked when a window is changed from minimized to normal state.
windowActivated(WindowEvent e)	Invoked when the window is set to be the active window.
windowDeactivated(WindowEvent e)	Invoked when the window is no longer the active window.
ActionListener	
actionPerformed(ActionEvent e)	Invoked when an action occurs.
ComponentListsner	
mouseExited(MouseEvent e)	Invoked when a mouse exits a component.
MouseMotionListener	
mouseDragged(MouseEvent e)	Invoked when a mouse button is pressed on a component and then dragged.
mouseMoved(MouseEvent e)	Invoked when a the mouse cursor is moved on to a component but mouse button is not pressed.
ContainerListener	
componentAdded(ContainerEvent e)	Invoked when a component is added to the container.
componentRemoved(ContainerEvent e)	Invoked when a component is removed from the container.

WindowListener	
windowOpened(WindowEvent e)	Invoked the first time a window is made visible
windowClosing(WindowEvent e)	Invoked when the user attempts to close the window from the window's system menu.
windowClosed(WindowEvent e)	Invoked when a window has been closed as the result of calling dispose on the window.
windowIconified(WindowEvent e)	Invoked when a window is changed from a normal to a minimized state.
windowDeiconified(WindowEvent e)	Invoked when a window is changed from minimized to normal state.
windowActivated(WindowEvent e)	Invoked when the window is set to be the active window.
windowDeactivated(WindowEvent e)	Invoked when the window is no longer the active window.
ActionListener	
actionPerformed(ActionEvent e)	Invoked when an action occurs.
ComponentListsner	
ComponentListsner	
itemStateChanged(ActionEvent e)	Invoked when an item has been selected or deselected by the user.

AdjustmentListener	
adjustmentValueChanged(ActionEvent e)	Invoked when the value of the adjustable has changed.
TextListener	
textValueChanged(ActionEvent e)	Invoked when the value of the text has changed.

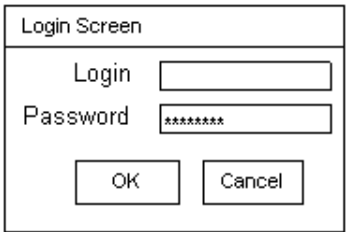
Adapter Classes:

The Adapter classes provided by AWT are as follows:
java.awt.event.ComponenentAdapter java.awt.event.ContainerAdapter
java.awt.event.FocusAdapter java.awt.event.KeyAdapter
java.awt.event.MouseAdapter java.awt.event.MouseMotionAdapter
java.awt.event.WindowAdapter

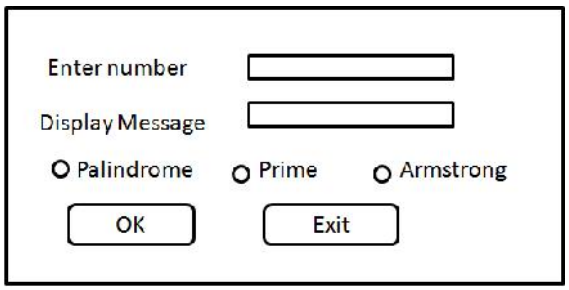
Event Handling
Lab Assignments

SET A

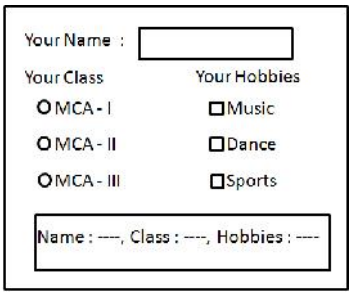
1. Write a java program to validate user login and password. If they do not match, display appropriate message in a dialog box. The user is allowed maximum 3 chances.



2. Write a java program to accept a number in a text box. Accept the choice of the user (Palindrome/prime/Armstrong) using radio buttons and .After clicking OK button Display message in the textbox(Display Message) if number correspond to the option or not.

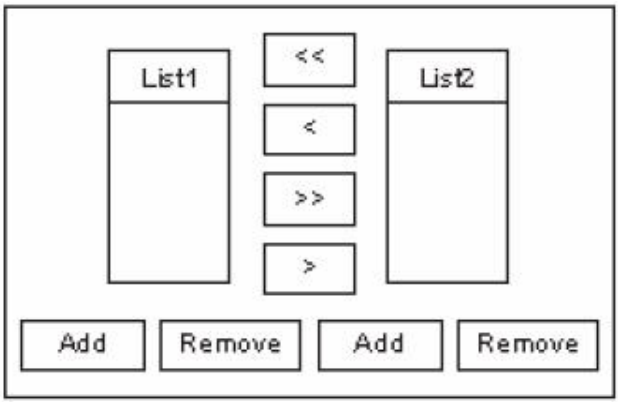


3. Write a java program to accept user name in a text box. Accept the class of the user (MCA-I/MCA-II/MCA-III) using radio buttons and the hobbies of the user using check boxes. Display the selected options in a text box.



SET B

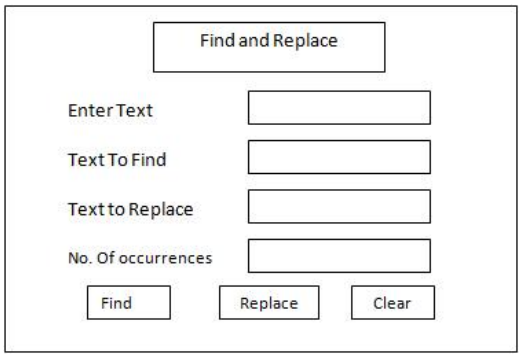
1. Write a program to create two lists and transfer elements from one list to another. Multiple selection is allowed. The Add button allows an element to be added and the Remove button allows an element to be removed (Accepted in an input dialog). Do not add duplicate elements.



2. Write Java program to design three text boxes and two buttons using swing . Enter different strings in first and second textbox. On clicking the First command button, concatenation of two strings should be displayed in third text box and on clicking second command button , reverse of string should display in third text box.

SET C

1. Write a Java program to design a screen using Swing that will create four text fields. First for the text, second for what to find and third for replace. Display result in the fourth text field. Also display the count of total no. of replacements made. The button clear to clear the text boxes.



Assignment Evaluation

0:Not Done [ ] 1:Incomplete [ ] 2.Late Complete [ ]

3:Needs Improvement [ ] 4:Complete [ ] 5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 8

Applet

Reading

You should read the following topics before starting this exercise
1. Concept of Applet
2. Difference between application and applet.
3. The Applet class
4. Applet lifecycle
5. Passing parameters to applets
6. The APPLET tag
7. Running an applet using browser and appletviewer

Ready Reference

Creating an applet

All applets are subclasses of the **java.applet.Applet** class. You can also create an applet by extending the **javax.swing.JApplet** class. The syntax is:

```
class MyApplet extends Applet
{
//applet methods
}
```

Applet methods:

Method	Purpose
init( )	-Automatically called to perform initialization of the applet. Executed only once.
start( )	-Called every time the applet moves into sight on the Web browser to allow the applet to start up its normal operations.
stop( )	- Called every time the applet moves out of sight on the Web browser to allow the applet to shut off expensive operations.
destroy( )	- Called when the applet is being unloaded from the page to perform final release of resources when the applet is no longer used.
paint()	- Called each time the applets output needs to be redrawn.

Running an applet

1. Compile the applet code using javac
2. Use the java tool – appletviewer to view the applet (embed the APPLET tag in comments in the code)

3. Use the APPLET tag in an HTML page and load the applet in a browser

Using appletviewer:

1. Write the HTML APPLET tag in comments in the source file.
2. Compile the applet source code using javac.
3. Use appletviewer ClassName.class to view the applet.

Using browser:

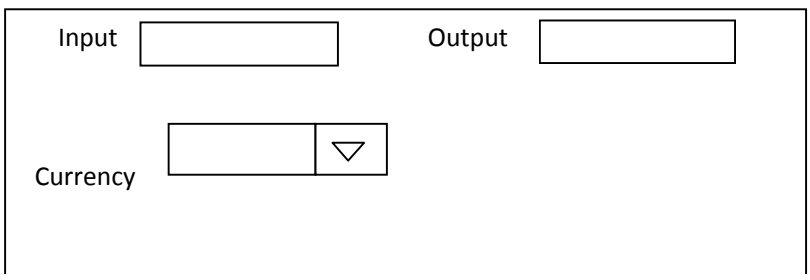
1. Create an HTML file containing the APPLET tag.
2. Compile the applet source code using javac.
3. In the web browser, open the HTML file.

The APPLET tag
< APPLET
[CODEBASE = *appletURL*]
CODE = appletClassFile
[ALT = alternateText]
[ARCHIVE = archiveFile]
[NAME = appletInstanceName]
WIDTH = pixels
HEIGHT = pixels
[ALIGN = alignment]
[VSPACE = pixels]
[HSPACE = pixels]
>
[< **PARAM NAME** = *AttributeName* **VALUE** = *AttributeValue* />]
</**APPLET**>

Lab Assignments
SET A

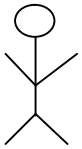
1. Create an applet to display a message at the left upper most corner of the applet. The message is passed as a parameter to the applet.
2. Create an applet to accept a number in a textbox and on click of OK button, it displays if given number is even or odd.

3. Create a conversion applet which accepts currency of other country in one unit and converts it to Indian currency. The input unit is selected from a list.



SET B

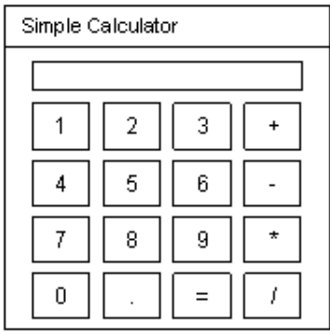
1. Create a Stickman in a applet.



2. Create an Applet which displays a message in the center of the screen. The message indicates the events taking place on the applet window. Handle events like mouse click, mouse moved, mouse dragged, mouse pressed, and key pressed. The message should update each time an event occurs. The message should give details of the event such as which mouse button was pressed, which key is pressed etc. (Hint: Use repaint(), KeyListener, MouseListener, MouseEvent method getButton, KeyEvent methods getKeyChar)

SET C

1.Create a calculator in an applet.



Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor Date of Completion

SESSION 9

Multithreading

Reading

You should read the following topics before starting this exercise:

1. Thread class
2. Runnable interface
3. Thread lifecycle
4. Thread methods

Ready Reference

Important methods of the Thread class:

Method	**Description**
static int activeCount()	-Returns the number of active threads in the current thread's thread group.
static Thread currentThread()	-Returns a reference to the currently executing thread object
String getName()	-Returns this thread's name.
int getPriority()	-Returns this thread's priority.
ThreadGroup getThreadGroup()	-Returns the thread group to which this thread belongs.
void interrupt()	-Interrupts this thread.
boolean isAlive()	-Tests if this thread is alive.
boolean isDaemon()	-Tests if this thread is a daemon thread.
boolean isInterrupted()	-Tests whether this thread has been interrupted.
void join()	-Waits for this thread to end.
boolean isAlive()	-Tests if this thread is alive.
boolean isDaemon()	- Tests if this thread is a daemon thread.
boolean isInterrupted()	-Tests whether this thread has been interrupted.
void join()	-Waits for this thread to end.
void setName(String name)	- Changes the name of this thread.
void setPriority(int newPriority)	-Changes the priority of this thread.
void sleep(long mSec)	-Causes the currently executing thread to sleep.
void start()	-Causes this thread to begin execution.
String toString()	-Returns a string representation of this thread, including the thread's name, priority, and thread group.
static void yield()	- Causes the currently executing thread object to temporarily pause and allow other threads to execute.
void setName(String name)	-Changes the name of this thread.
void setPriority(int newPriority)	-Changes the priority of this thread.
void sleep(long mSec)	-Causes the currently executing thread to sleep.
void start()	-Causes this thread to begin execution.

String toString()	- Returns a string representation of this thread, including the thread's name, priority, and thread group.
static void yield()	- Causes the currently executing thread object to temporarily pause and allow other threads to execute.

Lab Assignments

SET A

1. Write a JAVA program to accept the number from the user and do the following
- Calculate Factorial of a given Number.
- To check whether given number is prime or not.
(use Thread)

2. Write a JAVA program which will create two child threads by implementing Runnable interface; one thread will print even nos from 1 to 50 and other display vowels.

3.Define a thread called "PrintText_Thread" for printing text on command prompt for n number of times. Create three threads and run them. Pass the text and n as parameters to the thread constructor. Example:

i. First thread prints "I am in MCA-I" 10 times
ii. Second thread prints "I am in MCAII" 20 times
iii. Third thread prints "I am in MCA-III" 30 times

SET B

1. Write a JAVA program Design a screen with two buttons start thread and stop thread. Clicking on start ,it should start printing "Thread running" until stop button is pressed.

2. Write a JAVA program to design a screen with two textboxes and start button. Clicking on start should start two threads printing 1 to 100 in two textboxes.

3. Write a java program to create a class called FileWatcher that can be given several filenames that may or may not exist. The class should start a thread for each file name. Each thread will periodically check for the existence of its file. If the file exists, the thread will write a message to the console and then end.

SET C

1. Define a thread which moves the text "JAVA" in a panel as per the direction of Button clicked (Left, Right).

2. Write a program to show how three thread manipulate same stack , two of them are pushing elements on the stack, while the third one is popping elements off the stack.

Assignment Evaluation

0:Not Done [ ]	1:Incomplete [ ]	2.Late Complete [ ]
3:Needs Improvement [ ]	4:Complete [ ]	5:WellDone [ ]

Signature of the Instructor

Date of Completion