S.Y. Diploma : Sem. IV
[CO/CM/IF/CW]

Java Programming

Time: 3 Hrs.] Prelim Question Paper Solution [Marks : 70

Q.1 Attempt any FIVE of the following : **[10]**

Q.1(a) Give syntax and example of following math functions. **[2]**

(i) sqrt ( ) (ii) pow ( )

Ans.: (i) sqrt ( ): sqrt method is used for finding square root of a number.

Syntax: Math.sqrt(variable)

Here variable is double datatype

Example: Math.sqrt(9) = 3

(ii) pow ( ): Power of the number. Here variable1 is base value and variable2 is power value

Syntax: Math.pow(variable1,variable2)

Here variable is double datatype

Example: Math.pow(2,3) = 8

Q.1(b) Enlist access specifiers in Java. **[2]**

Ans.: There are 4 types of java access specifiers:

(i) public
(ii) private
(iii) default (Friendly)
(iv) protected

Q.1(c) State two ways to create thread. **[2]**

Ans.: Two ways for creating thread are:

(i) By implementing runnable interface
(ii) By extending thread class

Q.1(d) State the use of static keyword. **[2]**

Ans.: (i) The static keyword is used in java mainly for memory management. It is used with variables, methods, blocks and nested class.
(ii) It is a keyword that are used for share the same variable or method of a given class.
(iii) This is used for a constant variable or a method that is the same for every instance of a class. The main method of a class is generally labelled static.

Q.1(e) Enlist any four keywords used for exception handling in Java. **[2]**

Ans.: (i) throw
(ii) throws
(iii) finally
(iv) try
(v) catch

Q.1(f) Give syntax of <param> tag to pass parameters to an Applet. **[2]**

Ans.: To pass parameters to an applet <PARAM… > tag is used. Each <PARAM…> tag has a name attribute and a value attribute. Inside the applet code, the applet can refer to that parameter by name to find its value. The syntax of <PARAM…> tag is as follows

<PARAM NAME = name1 VALUE = value1>

Q.1(g) List two bitwise operators and two logical opertors. **[2]**

Ans.: Bitwise operators:

Bitwise AND - &
Bitwise OR - |
Bitwise XOR - ^

Bitwise NOT - ~
Logical operators:
Logical And - &&
Logical OR - ||
Logical NOT - !

Q.2 Attempt any THREE of the following : [12]

Q.2(a) Write a program to find largest between two numbers using '?:' operator. [4]

Ans.:

```
public class test
{
public static void main(String args[])
{
int minval;
int a=10;
int b=20;
minval=(a<b)?a:b;
System.out.println("Minimum Value="+minval);
}
}
```

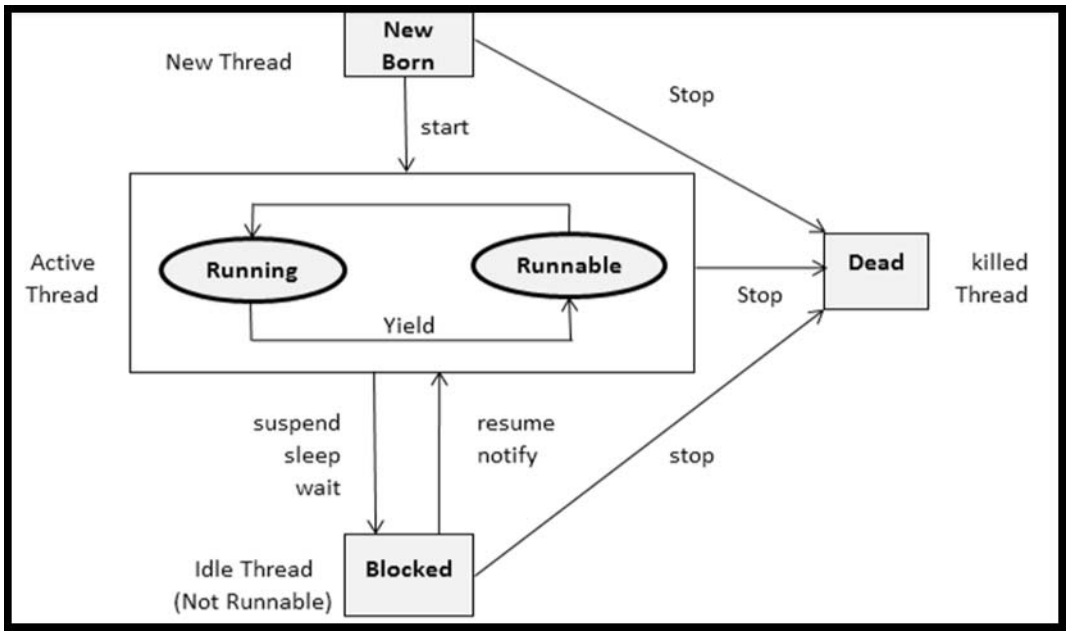
Q.2(b) Define class Student with suitable data members create two objects using two different constructors of the class. [4]

Ans.:

```
class Student4
{
 int id;
 String name;

Student4()
{
id = 111;
name = "Kiran";
}

Student4(int i,String n)
{
id = i;
name = n;
}

void display()
{
System.out.println(id+" "+name);
}
 class  demo
{
public static void main(String args[])
{
Student4 s1 = new Student4();
Student4 s2 = new Student4(222,"Aryan");
s1.display();
s2.display();
}
}
```

Q.2(c) Describe life cycle of thread with suitable diagram. **[4]**

Ans.:



Thread Life Cycle Thread has five different states throughout its life.

- Newborn State
- Runnable State
- Running State
- Blocked State
- Dead State

Thread should be in any one state of above and it can be move from one state to another by different methods and ways.

- **Newborn state:** When a thread object is created it is said to be in a new born state. When the thread is in a new born state it is not scheduled running from this state it can be scheduled for running by start() or killed by stop(). If put in a queue it moves to runnable state.
- **Runnable State:** It means that thread is ready for execution and is waiting for the availability of the processor i.e. the thread has joined the queue and is waiting for execution. If all threads have equal priority then they are given time slots for execution in round robin fashion. The thread that relinquishes control joins the queue at the end and again waits for its turn. A thread can relinquish the control to another before its turn comes by yield().
- **Running State:** It means that the processor has given its time to the thread for execution. The thread runs until it relinquishes control on its own or it is pre-empted by a higher priority thread.
- **Blocked state:** A thread can be temporarily suspended or blocked from entering into the runnable and running state by using either of the following thread method.
  suspend(): Thread can be suspended by this method. It can be rescheduled by resume().
  wait(): If a thread requires to wait until some event occurs, it can be done using wait method andcan be scheduled to run again by notify().
  sleep(): We can put a thread to sleep for a specified time period using sleep(time) where time is in ms. It reenters the runnable state as soon as period has elapsed /over
- **Dead State:** Whenever we want to stop a thread form running further we can call its stop(). The statement causes the thread to move to a dead state. A thread will also move to dead state automatically when it reaches to end of the method. The stop method may be used when the premature death is required

Q.2(d) Write a program to count number of words form a text file using stream classes. [4]

Ans.:

```
import java.io.*;
class copycon
{
        int count1,counct,len;

        void getdata()
        {
        try
        {
RandomAccessFile ff=new RandomAccessFile("student.dat","rw");
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        String choice=new String();
        String s=" ";
        int countc=count1=len=0;

        do{
        countc++;
        count1++;
        System.out.print("Enter String : ");
        s=in.readLine();
        ff.writeUTF(s);
        len=s.length();
        s=s.trim();

        for(int i=0;i<len;i++)
        {
        if(s.charAt(i)==' ')
        countc++;
        }
        System.out.print("Insert next line(Y/N) : ");
        choice=in.readLine();
        }
        while(choice.compareTo("Y")==0||choice.compareTo("y")==0);
        System.out.println("Number of Lines : "+count1);
        System.out.println("Number of words : "+countc);
        }
        catch(Exception e)
        {    }
        }

        public static void main(String args[])
        {
        copycon c=new copycon();
        c.getdata();
        File inFile=new File("input.txt");
        File outFile=new File("output.txt");
        FileReader ins=null;
        FileWriter outs=null;
        try
        {
        ins =new FileReader(inFile);
        outs =new FileWriter(outFile);
        int ch;
```

```
while((ch=ins.read())!=-1)
{
outs.write(ch);
}
}
catch(Exception e)
{
System.out.println(e);
System.out.println(-1);
}
finally
{
try
{
ins.close();
outs.close();
}
catch(IOException e)
{    }
}
}
}
```

Q.3 Attempt any THREE of the following : **[12]**

Q.3(a) Write a program to divide any positive even integer by 2 using bitwise shift operator. **[4]**

Ans.:

```
import java.io.*;
class test
{
public static void main(String args[]) throws IOException
{
DataInputStream d=new DataInputStream(System.in);
System.out.println("Enter number");
int n=Integer.parseInt(d.readLine());
if(((n >> 1) << 1) == n)
{
System.out.println(n+ " is Divisible by 2");
}
else
{
System.out.println(n + " is Not Divisible by 2");
}
}
}
```

Q.3(b) State need of interface with suitable examples. **[4]**

Ans.:

- It is used to achieve total abstraction.
- Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance.
- It is also used to achieve loose coupling.
- Interfaces are used to implement abstraction.

```
import java.io.*;
interface in1
{
```

```
final int a = 10;
void display();
}

class testClass implements in1
{

public void display()
{
System.out.println("Hello");
}

public static void main (String[] args)
{
testClass t = new testClass();
t.display();
System.out.println(a);
}
}
```

Q.3(c) Give usage of following methods: **[4]**

(i) drawOval() (ii) getFont() (iii) drawArc() (iv) getFamily()

Ans.: **(i) drawOval():** Drawing Ellipses and circles: To draw an Ellipses or circles used drawOval() method can be used.

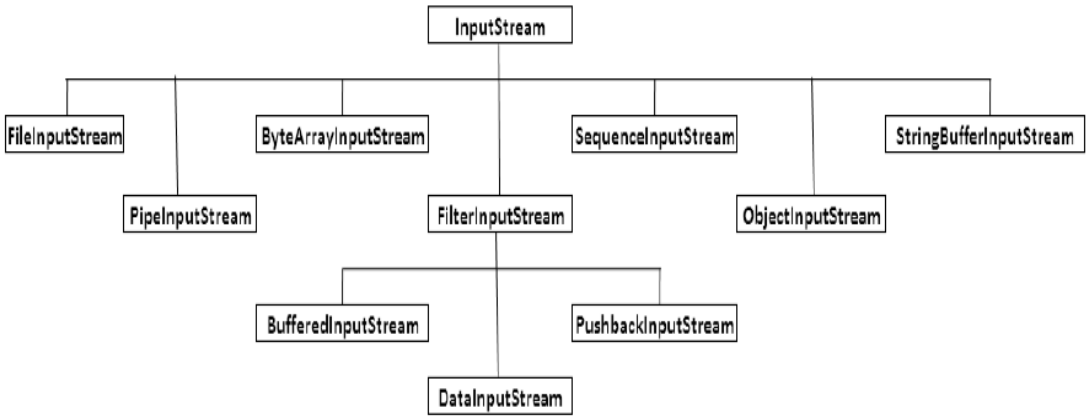
(ii) getFont(): Returns the font associated with the system property specified by property. null is returned if property does not exist.

(iii) drawArc(): It is used to draw arc.

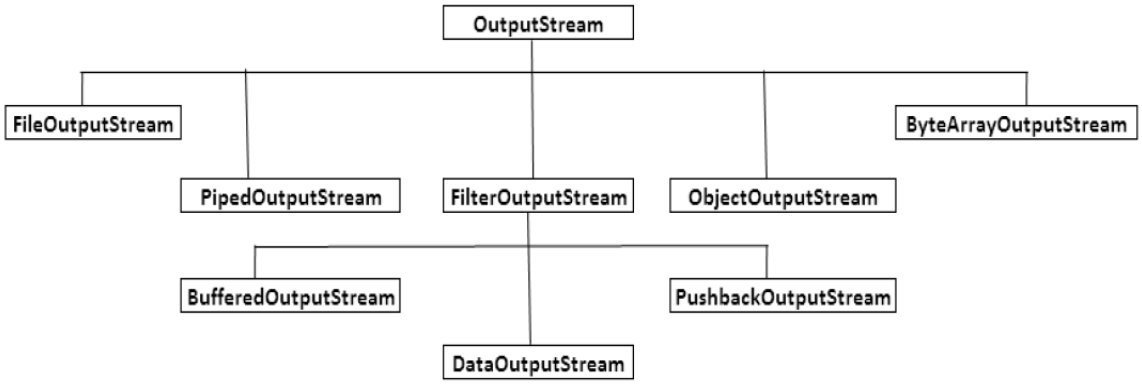
(iv) getFamily(): Returns the name of the font family to which the invoking font belongs.

Q.3(d) Enlist types of stream classes and describe methods for reading and writing data for each type. **[4]**

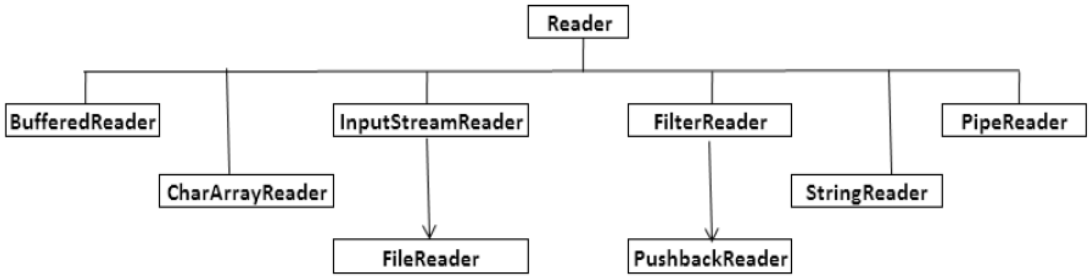
Ans.:



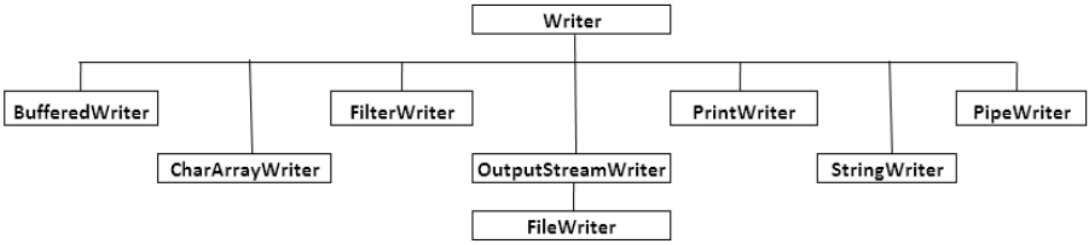
InputStream class hierarchy



OutputStream class hierarchy



Hierarchy of reader stream classes



Hierarchy of writer stream classes

Input stream class methods:

(i) int read () - Returns an integer representation of next available byte of input.-1 is returned at the stream end.

(ii) 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.

(iii) 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.

Output stream class methods:

(i) void write (int b) - Writes a single byte to an output stream.

(ii) void write(byte buffer[ ]) - Writes a complete array of bytes to an output stream.

(iii) void write (byte buffer[ ], int offset, int numbytes) - Writes a sub range of numbytes bytes from the array buffer, beginning at buffer[offset].

Methods in Reader Class:

(i) int read () - Returns an integer representation of next available character from invoking stream. - 1 is returned at the stream end.

(ii) 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.
(iii) 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.

Methods in Writer Class:
(i) void write (int ch) - Writes a single character to an output stream.
(ii) void write(char buffer[ ]) - Writes a complete array of characters to an output stream.
(iii) void write (char buffer[ ], int offset, int numchars) - Writes a sub range of numchars from the array buffer, beginning at buffer[offset].
(iv) void write(String str) - Writes str to output stream.
(v) void write(String str, int offset, int numchars) - Writes a subrange of numchars from string beginning at offset.

Q.4 Attempt any THREE of the following : [12]

Q.4(a) Describe types of variables in Java with their scope. [4]

Ans.: The scope of a variable defines the section of the code in which the variable is visible.

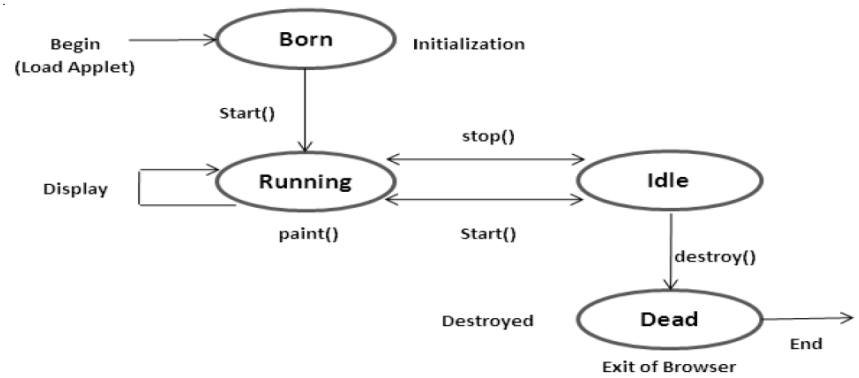
Instance variables
Instance variables are those that are defined within a class itself and not in any method or constructor of the class. They are known as instance variables because every instance of the class (object) contains a copy of these variables. The scope of instance variables is determined by the access specifier that is applied to these variables.

Argument variables
These are the variables that are defined in the header oaf constructor or a method. The scope of these variables is the method or constructor in which they are defined.

Local variables
A local variable is the one that is declared within a method or a constructor (not in the header). The scope and lifetime are limited to the method itself.

Q.4(b) Describe life cycle of applet with suitable diagram. [4]

Ans.:



Applet Life Cycle: An Applet has a life cycle, which describes how it starts, how it operates and how it ends. The life cycle consists of four methods: init(), start(), stop() and destroy().

Initialization State (The init() method):
The life cycle of an Applet is begin on that time when the applet is first loaded into the browser and called the init() method. The init() method is called only one time in the life cycle on an Applet. The init() method is basically called to read the "PARAM" tag in the html file. The init () method retrieve the passed parameter through the "PARAM" tag of html file using get Parameter() method All the initialization such as initialization of variables and the

objects like image, sound file are loaded in the init () method .After the initialization of the init() method user can interact with the Applet and mostly applet contains the init() method. We may do following thing if required.

- Create objects needed by the applet
- Set up initial values
- Load images or fonts
- Set up colors

Running State (The start() method): The start method of an Applet is called after the initialization method init(). This method may be called multiples time when the Applet needs to be started or restarted. For Example if the user wants to return to the Applet, in this situation the start() method of an Applet will be called by the web browser and the user will be back on the applet. In the start method user can interact within the applet.

```
public void start()
{
……..
……..
}
```

Idle (The Stop() method): An applet becomes idle when it is stopped from running. The stop() method stops the applet and makes it invisible. Stopping occurs automatically when we leave the page containing the currently running applet. We can also do so by calling the stop() method explicitly. The stop() method can be called multiple times in the life cycle of applet like the start () method or should be called at least one time. For example the stop() method is called by the web browser on that time When the user leaves one applet to go another applet and the start() method is called on that time when the user wants to go back into the first program or Applet.

```
public void stop()
{
…….
……..
}
```

Dead State (The destroy() method): The destroy() method is called to terminate an Applet. An Applet is said to be dead when it is removed from memory. This occurs automatically by invoking the destroy() method when we quit the browser. It is useful for clean-up actions, such as releasing memory after the applet is removed, killing off threads and closing network/database connections. Thus this method releases all the resources that were initialized during an applet's initialization.

```
public void destroy()
{
……..
……..
}
```

Display State (The paint() method): The paint() method is used for applet display on the screen. The display includes text, images, graphics and background. This happens immediately after the applet enters into the running state. Almost every applet will have a paint() method and can be called several times during an applet's life cycle. The paint() method is called whenever a window is required to paint or repaint the applet.

```
public void paint(Graphics g)
{
……..
……..
}
```

Q.4(c) Write a program to create package Math_s having two classes as addition and subtraction. Use suitable methods in each class to perform basic operations. **[4]**

Ans.:

```
package Math_s;
public class addition
{
Int a,b;
public add(int x,int y)
{
a=x;
b=y;
return(a+b);
}
}
public class subtraction
{
Int a,b;
public sub(int x, int y)
{
a=x;
b=y;
return(a-b);
}
}

import Math_s.addition;
import Math_s.subtraction;

class Basic
{
public static void main(String args[])
{
addition a=new addition( );
int sum=a.add(10,5);
subtraction s = new subtraction();
int subt=s.sub(10,5);
System.out.println("Addition= "+sum);
System.out.println("Subtraction= "+subt);
}
}
```

Q.4(d) Differentiate between Java Application and Java Applet (any 4 points) **[4]**

Ans.:

Java Application	Applet
An application is the programs executed on the computer independently.	It is small program uses another application program for its execution.
Uses the main method for execution	Do not use the main method
Can run alone but require JRE.	Cannot run independently require API's (Ex. Web API).
Requires prior explicit installation on the local computer.	Prior installation is not needed
Applications are capable of performing those operations to the files on the local computer.	The files cannot be read and write on the local computer through applet.
No security concerns are there.	Requires security for the system as they are untrusted.

Q.4(e) Write a program to copy content of file "input.txt into file "output.txt". [4]

Ans.:

```
import java.io.*;
class copyf
{
public static void main(String args[]) throws IOException
{
BufferedReader in=null;
BufferedWriter out=null;
try
{
in=new BufferedReader(new FileReader("input.txt"));
out=new BufferedWriter(new FileWriter("output.txt"));
int c;
while((c=in.read())!=-1)
{
out.write(c);
}
System.out.println("File copied successfully");
}
finally
{
if(in!=null)
{
in.close();
}
if(out!=null)
{
out.close();
}
}
}
}
```

Q.5 Attempt any TWO of the following : [12]

Q.5(a) Write a step to declare and define two and three dimensional arrays of a class. [6]

Ans.: Two – dimensional array is the simplest form of a multidimensional array. A two – dimensional array can be seen as an array of one – dimensional array for easier understanding.

Declaration Syntax:

data_type[][] array_name = new data_type[x][y];

For example: int[][] arr = new int[10][20];

Initialization Syntax:

array_name[row_index][column_index] = value;

For example: arr[0][0] = 1;

For example:

```
class GFG
{
public static void main(String[] args)
{
int[][] arr = new int[10][20];
arr[0][0] = 1;
System.out.println("arr[0][0] = " + arr[0][0]);
}
}
```

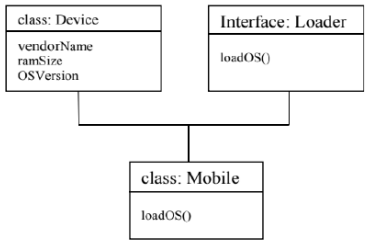
Three – dimensional array is a complex form of a multidimensional array. A three – dimensional array can be seen as an array of two – dimensional array for easier understanding.

Declaration Syntax:
data_type[][][] array_name = new data_type[x][y][z];
For example: int[][][] arr = new int[10][20][30];
Initialization Syntax:
array_name[array_index][row_index][column_index] = value;
For example: arr[0][0][0] = 1;

For Example:

```
class GFG
{
public static void main(String[] args)
{
int[][][] arr = new int[10][20][30];
arr[0][0][0] = 1;
System.out.println("arr[0][0][0] = " + arr[0][0][0]);
}
}
```

Q.5(b) Implement following inheritance: **[6]**



Display details of devices from load OS( ) method of class Mobile.

Ans.:

```
import java.util.*;
class device
{
String vendorname;
int ramsize;
double OSversion;
void input()
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter VendorName:");
vendorname=sc.nextLine();
System.out.println("Enter RAM size:");
ramsize=sc.nextInt();
System.out.println("Enter OSversion");
OSversion=sc.nextDouble();
}
}
interface loader
{
void loados();
}
class mobile extends device implements loader
{
public void loados()
```

```
{
super.input();
System.out.println("VendorName:"+vendorname+"\nRAM
size:"+ramsize+"\nOSversion:"+OSversion);
}
}
class Phone
{
public static void main(String args[])
{
mobile obj=new mobile();
obj.loados();
}
}
```

Q.5(c) Write a program to define two threads for displaying even and odd numbers from 1 to 20 respectively with a delay of 200 ms after each number. **[6]**

Ans.:

```
import java.lang.*;
class even extends Thread
{
public void run()
{
try
{
for(int i=0;i<=20;i=i+2)
{
System.out.println("\tEven thread="+i);
Thread.sleep(200);
}
}
catch(InterruptedException e)
{ }
}
}
class odd extends Thread
{
public void run()
{
try
{
for(int i=1;i<=20;i=i+2)
{
System.out.println("\todd thread="+i);
Thread.sleep(200);
}
}
catch(InterruptedException e)
{ }
}
}
class evenodd
{
public static void main(String args[])
{
```

```
new even().start();
new odd().start();
System.out.println("Exit from main");
}
}
```

Q.6 Attempt any TWO of the following: **[12]**

Q.6(a) Write a program to define class Employee with members as id and salary. Accept data for five employees and display details of employees getting highest salary. **[6]**

Ans.:

```
import java.lang.*;
import java.io.*;
class employee
{
int empid;
int empsalary;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
void getdata()
{
try
{
System.out.println("Enter Employee ID=");
empid=Integer.parseInt(br.readLine());

System.out.println("Enter Employee Salary=");
empsalary=Integer.parseInt(br.readLine());
}
catch(Exception e)
{
System.out.println("Error");
}
}
int getsalary()
{
  return empsalary;
}
void display()
{

System.out.println("Employee ID="+empid);
System.out.println("Employee Salary="+empsalary);
}
}
class emp
{
public static void main(String args[])
{
employee e[]=new employee[5];

for(int i=0;i<5;i++)
{
     e[i]=new employee();
     e[i].getdata();
}
for(int i=0;i<5;i++)
```

```
{
    e[i].display();
}

int maxsalary=0;
int id=0;
for(int i=0;i<5;i++)
{
    if(e[i].getsalary()>maxsalary)
    {
        maxsalary = e[i].getsalary();
        id = i;
    }
}
System.out.println("Employee with Highest Salary");
e[id].display();
}
}
```

Q.6(b) Describe types of Errors and exceptions in details. **[6]**

Ans.: Errors are broadly classified into two categories:-

- Compile time errors
- Runtime errors

(i) Compile time errors: All syntax errors will be detected and displayed by java compiler and therefore these errors are known as compile time errors. The most of common problems are:

- Missing semicolon
- Missing (or mismatch of) bracket in classes & methods
- Misspelling of identifiers & keywords
- Missing double quotes in string
- Use of undeclared variables.
- Bad references to objects.

(ii) Runtime errors: Sometimes a program may compile successfully creating the .class file but may not run properly. Such programs may produce wrong results due to wrong logic or may terminate due to errors such as stack overflow. When such errors are encountered java typically generates an error message and aborts the program. The most common run-time errors are:

- Dividing an integer by zero
- Accessing an element that is out of bounds of an array
- Trying to store value into an array of an incompatible class or type
- Passing parameter that is not in a valid range or value for method
- Trying to illegally change status of thread
- Attempting to use a negative size for an array
- Converting invalid string to a number
- Accessing character that is out of bound of a string

Types of exceptions:

- **Built in exceptions:**
  1. Arithmetic Exception
     It is thrown when an exceptional condition has occurred in an arithmetic operation.
  2. ArrayIndexOutOfBoundException
     It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.

3. ClassNotFoundException
   This Exception is raised when we try to access a class whose definition is not found
4. FileNotFoundException
   This Exception is raised when a file is not accessible or does not open.
5. IOException
   It is thrown when an input-output operation failed or interrupted
6. InterruptedException
   It is thrown when a thread is waiting , sleeping , or doing some processing , and it is interrupted.
7. NoSuchFieldException
   It is thrown when a class does not contain the field (or variable) specified
8. NoSuchMethodException
   It is thrown when accessing a method which is not found.
9. NullPointerException
   This exception is raised when referring to the members of a null object. Null represents nothing
10. NumberFormatException
    This exception is raised when a method could not convert a string into a numeric format.
11. RuntimeException
    This represents any exception which occurs during runtime.
12. StringIndexOutOfBoundsException
    It is thrown by String class methods to indicate that an index is either negative than the size of the string

- **User-Defined Exceptions**
  Sometimes, the built-in exceptions in Java are not able to describe a certainsituation. In such cases, user can also create exceptions which are called 'user-defined Exceptions'.
  The user should create an exception class as a subclass of Exception class. Since all the exceptions are subclasses of Exception class, the user should also make his class a subclass of it. This is done as:
  class MyException extends Exception

Q.6(c) Design an Applet to pass username and password as parameters and check if password contains more than 8 characters. **[6]**

Ans.:

```
import java.lang.*;
import java.awt.*;
import java.applet.*;s
public class Hello extends Applet
{
String message;
int l;
public void init()
{
message=getParameter("password");
l= message.length();
}
public void paint(Graphics g)
{
if(l>8)
{
g.drawString("Password is greater than 8",50,50);
}
```

```
else
{
g.drawString("Password is not greater than 8",50,50);
}
}
}
/*<applet code= Hello height=300 width=300>
<param name="username" value="Abc">
<param name="password" value="qwert">
</applet>*/
```

❑ ❑ ❑ ❑ ❑