



WINTER – 2023 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

Subject Name: Java Programming

Subject Code: **22412**

Important Instructions to examiners:

- 1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
- 2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
- 3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
- 4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
- 5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
- 6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
- 7) For programming language papers, credit may be given to any other program based on equivalent concept.
- 8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English + Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1		Attempt any <u>FIVE</u> of the following:	10 M
	a)	Enlist any two logical operators and two bitwise operators.	2 M
	A l Ans	<p>Logical Operators:</p> <ol style="list-style-type: none"> 1. AND Operator (&&) – if(a && b) [if true execute else don't] 2. OR Operator () – if(a b) [if one of them is true to execute else don't] 3. NOT Operator (!) – !(a<b) [returns false if a is smaller than b] <p>Bitwise Operator:</p> <ol style="list-style-type: none"> 1. Bitwise OR () 2. Bitwise AND (&) 3. Bitwise XOR (^) 4. Bitwise Complement (~) 5. Bitwise left shift(<<) 6. Bitwise right shift(>>) 	<p>List any two Logical operator : 2 marks</p> <p>List any two Bitwise operator : 2 marks</p>



	b)	Define constructor.	2 M
	Ans	A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.	Correct/suitable definition- 1 M Syntax or



		For Example: <pre>class Test { Test() { // constructor body } }</pre>	Example- 1 M								
	c)	Write down the syntax of array declaration, initialization.	2 M								
	Ans	<p>The syntax of declaring an array in Java is given below.</p> <div>datatype [] arrayName;</div> <p>Here, the datatype is the type of element that will be stored in the array, square bracket[] is for the size of the array, and arrayName is the name of the array.</p> <p>The syntax of initializing an array is given below.</p> <div>datatype [] arrayName = new datatype [size];</div>	1 M- array declaration and 1 M-array initialization								
	d)	List out different ways to access package from another package.	2 M								
	Ans	There are three ways to access the package from outside the package. <ul style="list-style-type: none">• import package.*;• import package.classname;• fully qualified name	Any 2 correct ways – 2 M								
	e)	Differentiate between starting thread with run() method and start() method.	2 M								
	Ans	<table><tr><th>start()</th><th>run()</th></tr><tr><td>Creates a new thread and the run() method is executed on the newly created thread.</td><td>No new thread is created and the run() method is executed on the calling thread itself.</td></tr><tr><td>Can't be invoked more than one time</td><td>Multiple invocation is possible</td></tr><tr><td>Defined in java.lang.Thread class.</td><td>Defined in java.lang.Runnable interface and must be overridden in the implementing class.</td></tr></table>	start()	run()	Creates a new thread and the run() method is executed on the newly created thread.	No new thread is created and the run() method is executed on the calling thread itself.	Can't be invoked more than one time	Multiple invocation is possible	Defined in java.lang.Thread class.	Defined in java.lang.Runnable interface and must be overridden in the implementing class.	Any 2 valid points- 2 M
start()	run()										
Creates a new thread and the run() method is executed on the newly created thread.	No new thread is created and the run() method is executed on the calling thread itself.										
Can't be invoked more than one time	Multiple invocation is possible										
Defined in java.lang.Thread class.	Defined in java.lang.Runnable interface and must be overridden in the implementing class.										



		<p>It starts thread to begin execution, JVM calls run method of this thread.</p> <p>Syntax: public void start()</p> <p>A new thread will be created and it is responsible to complete the job.</p>	<p>It is used to perform operations by thread.</p> <p>Syntax: public void run()</p> <p>No new thread will be created and main thread will be responsible to complete the job.</p>	
	f)	State the classes that can an applet extend.		2 M
	Ans	<ul style="list-style-type: none"> • Graphics • Font • Color 		Any 2 classes-2 M
	g)	Give syntax to open a file using InputStream class.		2 M
	Ans	<p>Attach a file to a FileInputStream as this will enable us to read data from the file as shown below as follows:</p> <p style="text-align: center;">FileInputStream input = new FileInputStream("input.txt");</p> <p>Now in order to read data from the file, we should read data from the FileInputStream as shown below:</p> <p style="text-align: center;">ch=fileInputStream.read();</p>		Correct syntax-2 M
2.		Attempt any <u>THREE</u> of the following:		12 M
	a)	Write a program lo display ASCII value of a number 9.		4 M
	Ans	<pre>public class asciivalue { public static void main(String args[]) { // Character whose ASCII is to be computed char ch = '9'; // Creating a new variable of type int and assigning the character value. int ascii = ch; // Printing the ASCII value of above character System.out.println("The ASCII value of " + ch+ " is: " + ascii); } }</pre> <p>Output:</p> <p>The ASCII value of 9 is: 57</p>		For any correct program: 4m



	b)	Write a program to sort the elements of an array in ascending order.	4 M
	Ans	<pre>class arraysort { public static void main(String args[]) { int a[]={85,95,78,45,12,56,78,19}; int i=0; int j=0; int temp=0; int l=a.length; for(i=0;i<l;i++) { //apply bubble sort for(j=(i+1);j<l;j++) { if(a[i]>a[j]) { temp=a[i]; a[i]=a[j]; a[j]=temp; } } } System.out.println("Ascending order of numbers:"); for(i=0;i<l;i++) System.out.println(""+a[i]); } }</pre> <p>Output: Ascending order of numbers: 12 19 45 56 78 78 85 95</p>	For any correct logic and program: 4m
	c)	Define Thread. Draw life cycle of Thread.	4 M



Ans	<p>Thread is a smallest unit of executable code or a single task is also called as thread.</p> <p>Each tread has its own local variable, program counter and lifetime.</p> <p>A thread is similar to program that has a single flow of control.</p> <div data-bbox="342 321 1224 953"><pre>graph TD Newborn[Newborn] -- start() --> Running([Running]) Newborn -- stop() --> Dead[Dead] Running -- yield() --> Runnable([Runnable]) Runnable -- stop() --> Dead Runnable -- suspend() / sleep(t) / wait() --> Blocked[Blocked] Blocked -- resume() --> Runnable Blocked -- notify() --> Runnable Blocked -- stop() --> Dead</pre><p>The diagram illustrates the life cycle of a thread. It starts in the 'Newborn' state. Calling 'start()' transitions the thread to the 'Running' state. From 'Running', calling 'yield()' transitions it to the 'Runnable' state. From 'Runnable', calling 'suspend()', 'sleep(t)', or 'wait()' transitions it to the 'Blocked' state. From 'Blocked', calling 'resume()' or 'notify()' transitions it back to 'Runnable'. From either 'Running' or 'Runnable', calling 'stop()' transitions the thread to the 'Dead' state.</p></div> <p>Fig: Life cycle of Thread</p>	<p>Definition of thread- 2 M</p> <p>Diagram: 2M</p>
d)	Write a program to read a file and then count number of words.	4 M
Ans	<pre>// Java program to count the no. of words in a file import java.io.*; public class Test11 { public static void main(String[] args) throws IOException { File file = new File("C:\\Program Files\\Java\\jdk1.7.0_80\\bin\\a.txt"); FileInputStream fileInputStream = new FileInputStream(file); InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream); BufferedReader bufferedReader = new BufferedReader(inputStreamReader); String line; int wordCount = 0; int paraCount = 0; while ((line = bufferedReader.readLine()) != null) { if (line.equals("")) {</pre>	<p>2 M - correct variable and object creation</p> <p>2 M - valid logic to count words from file.</p>



		<pre>paraCount += 1; } else { String words[] = line.split("\\s+"); wordCount += words.length; } } System.out.println("Total word count = " + wordCount); } }</pre> <p>C:\Program Files\Java\jdk1.7.0_80\bin>javac Test11.java C:\Program Files\Java\jdk1.7.0_80\bin>java Test11 Total word count = 8</p>	
3.		Attempt any <u>THREE</u> of the following:	12 M
	a)	Write a program which displays functioning of ATM machine, (Hint: Withdraw, Deposit, Check Balance and Exit)	4 M
	Ans	<pre>import java.util.Scanner; public class ATM_Transaction { public static void main(String args[]) { int balance = 5000, withdraw, deposit; Scanner s = new Scanner(System.in); while(true) { System.out.println("Automated Teller Machine"); System.out.println("Choose 1 for Withdraw"); System.out.println("Choose 2 for Deposit"); System.out.println("Choose 3 for Check Balance"); System.out.println("Choose 4 for EXIT"); System.out.print("Choose the operation you want to perform:"); int n = s.nextInt(); switch(n) {</pre>	4 M for correct program Or any other relevant logic should be considered



		<pre>case 1: System.out.print("Enter money to be withdrawn:"); withdraw = s.nextInt(); if(balance >= withdraw) { balance = balance - withdraw; System.out.println("Please collect your money"); } else { System.out.println("Insufficient Balance"); } System.out.println(""); break; case 2: System.out.print("Enter money to be deposited:"); deposit = s.nextInt(); balance = balance + deposit; System.out.println("Your Money has been successfully deposite"); System.out.println(""); break; case 3: System.out.println("Balance : "+balance); System.out.println(""); break; case 4: System.exit(0); } }</pre>	
	b)	Differentiate between method overloading and method overriding.	4 M



	Ans			4 M for any four correct point
		Method Overloading	Method Overriding	
		Method overloading is a compile-time polymorphism.	Method overriding is a run-time polymorphism.	
		Method overloading helps to increase the readability of the program.	Method overriding is used to grant the specific implementation of the method which is already provided by its parent class or superclass.	
		It occurs within the class.	It is performed in two classes with inheritance relationships.	
		Method overloading may or may not require inheritance.	Method overriding always needs inheritance.	
		In method overloading, methods must have the same name and different signatures.	In method overriding, methods must have the same name and same signature.	
		In method overloading, the return type can or can not be the same, but we just have to change the parameter.	In method overriding, the return type must be the same or co-variant.	
		Static binding is being used for overloaded methods.	Dynamic binding is being used for overriding methods.	
		Poor Performance due to compile time polymorphism.	It gives better performance. The reason behind this is that the binding of overridden methods is being done at runtime.	
	c)	Explain applet life cycle in detail.		4 M



Ans	<p>The applet life cycle can be defined as the process of how the object is created, started, stopped, and destroyed during the entire execution of its application. It basically has five core methods namely <code>init()</code>, <code>start()</code>, <code>stop()</code>, <code>paint()</code> and <code>destroy()</code>. These methods are invoked by the browser to execute.</p> <pre>graph TD Begin([Begin (Load Applet)]) --> Born([Born]) Born -- "Start()" --> Running([Running]) Running -- "stop()" --> Idle([Idle]) Idle -- "Start()" --> Running Running -- "paint()" --> Running Idle -- "destroy()" --> Dead([Dead]) Dead -- "End" --> Exit([Exit of Browser])</pre> <p>1. Initialization State (The <code>init()</code> method):</p> <ul style="list-style-type: none">• The life cycle of an Applet is begin on that time when the applet is first loaded into the browser and called the <code>init()</code> method.• The <code>init()</code> method is called only one time in the life cycle on an Applet. The <code>init()</code> method is basically called to read the “PARAM” tag in the html file.• The <code>init ()</code> method retrieve the passed parameter through the “PARAM” tag of html file using <code>getParameter()</code> method All the initialization such as initialization of variables and the objects like image, sound file are loaded in the <code>init ()</code> method.• After the initialization of the <code>init()</code> method user can interact with the Applet and mostly applet contains the <code>init()</code> method.• Syntax: <pre>public void init() { --- --- }</pre> <p>2. Running State (The <code>start()</code> method):</p> <ul style="list-style-type: none">• The <code>start</code> method of an Applet is called after the initialization method <code>init()</code>. This method may be called multiples time when the Applet needs to be started or restarted.	2 M for diagram and 2 M for explanation
------------	--	---



- 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.
- Syntax:-
public void start()
{
.....
.....
}

3. 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.
- **Syntax:-**
public void stop()
{
.....
.....
}

4. 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.
- **Syntax:-**
public void destroy()
{
.....
.....
}



		<p><u>5. Display State (The paint() method):</u></p> <ul style="list-style-type: none">• 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.• Syntax:- public void paint(Graphics g) { }	
	d)	Differentiate between Byte Stream Class and Character Stream Class. (Any four points)	4 M



	Ans	<table><tr><th>Byte Stream Class</th><th>Character Stream Class</th></tr><tr><td>Byte streams access the file byte by byte (8 bits).</td><td>A character stream will read a file character by character (16 bits).</td></tr><tr><td>Byte stream classes are classified into: 1. Input Stream Classes 2. Output Stream Classes</td><td>Character stream classes are classified into: 1. Reader class 2. Writer class</td></tr><tr><td>InputStream/OutputStream class is byte-oriented.</td><td>The Reader/Writer class is character-oriented.</td></tr><tr><td>The methods for byte streams generally work with byte data type.</td><td>The methods for character streams generally accept parameters of data type <i>char</i> parameters.</td></tr><tr><td>Byte-stream classes end with the suffix InputStream and OutputStream.</td><td>Character-stream classes end with the suffix Reader or Writer.</td></tr><tr><td>It is possible to translate character stream into byte stream with OutputStreamWriter.</td><td>It is possible to translate byte stream into a character stream with InputStreamReader.</td></tr><tr><td>Byte streams specifically used for reading and writing data in byte format.</td><td>The advantage of character streams, that is make it easy to write programs, which is not dependent upon a specific character encoding.</td></tr><tr><td>No conversion needed.</td><td>Character streams convert the underlying data bytes to Unicode, which is a costly operation.</td></tr><tr><td>InputStream and OutputStream are used for reading or writing binary data.</td><td>Reader and Writer uses Unicode, hence they can be internationalized. Hence in some cases they are more efficient than byte streams.</td></tr></table>	Byte Stream Class	Character Stream Class	Byte streams access the file byte by byte (8 bits).	A character stream will read a file character by character (16 bits).	Byte stream classes are classified into: 1. Input Stream Classes 2. Output Stream Classes	Character stream classes are classified into: 1. Reader class 2. Writer class	InputStream/OutputStream class is byte-oriented.	The Reader/Writer class is character-oriented.	The methods for byte streams generally work with byte data type.	The methods for character streams generally accept parameters of data type <i>char</i> parameters.	Byte-stream classes end with the suffix InputStream and OutputStream.	Character-stream classes end with the suffix Reader or Writer.	It is possible to translate character stream into byte stream with OutputStreamWriter.	It is possible to translate byte stream into a character stream with InputStreamReader.	Byte streams specifically used for reading and writing data in byte format.	The advantage of character streams, that is make it easy to write programs, which is not dependent upon a specific character encoding.	No conversion needed.	Character streams convert the underlying data bytes to Unicode, which is a costly operation.	InputStream and OutputStream are used for reading or writing binary data.	Reader and Writer uses Unicode, hence they can be internationalized. Hence in some cases they are more efficient than byte streams.	4 M for any correct 4 point
Byte Stream Class	Character Stream Class																						
Byte streams access the file byte by byte (8 bits).	A character stream will read a file character by character (16 bits).																						
Byte stream classes are classified into: 1. Input Stream Classes 2. Output Stream Classes	Character stream classes are classified into: 1. Reader class 2. Writer class																						
InputStream/OutputStream class is byte-oriented.	The Reader/Writer class is character-oriented.																						
The methods for byte streams generally work with byte data type.	The methods for character streams generally accept parameters of data type <i>char</i> parameters.																						
Byte-stream classes end with the suffix InputStream and OutputStream.	Character-stream classes end with the suffix Reader or Writer.																						
It is possible to translate character stream into byte stream with OutputStreamWriter.	It is possible to translate byte stream into a character stream with InputStreamReader.																						
Byte streams specifically used for reading and writing data in byte format.	The advantage of character streams, that is make it easy to write programs, which is not dependent upon a specific character encoding.																						
No conversion needed.	Character streams convert the underlying data bytes to Unicode, which is a costly operation.																						
InputStream and OutputStream are used for reading or writing binary data.	Reader and Writer uses Unicode, hence they can be internationalized. Hence in some cases they are more efficient than byte streams.																						
4.		Attempt any <u>THREE</u> of the following:	12 M																				
	a)	Explain implicit and explicit type conversion with example in detail.	4 M																				
	Ans	<p><u>Widening (Implicit)</u></p> <ul style="list-style-type: none">The process of assigning a smaller type to a larger one is known as widening or implicit. <p>Byte → short → int → long → float → double</p>	2 M for Implicit with example And 2 M for Explicit with example																				



		<p>For e.g. class widening</p> <pre>{ public static void main(String arg[]) { int i=100; long l=i; float f=l; System.out.println("Int value is"+i); System.out.println("Long value is"+l); System.out.println("Float value is"+f); } }</pre> <p><u>Narrowing (Explicit)</u></p> <ul style="list-style-type: none">• The process of assigning a larger type into a smaller one is called narrowing.• Casting into a smaller type may result in loss of data. <p>double → long → int → short → byte</p> <p>For e.g. class narrowing</p> <pre>{ Public static void main(String[]) { Double d=100.04; Long l=(long) d; Int i=(int) l; System.out.println("Int value is"+i); System.out.println("Long value is"+l); System.out.println("Float value is"+f); } }</pre>	
	b)	Write a program to show the use of copy constructor.	4 M
	Ans	<pre>class student { int id; String name; student(int i, String n) { id=i; name=n; } }</pre>	4 M for any suitable correct program



		<pre>student (student s)//copy constructor { id=s.id; name=s.name; } void display() { System.out.println(id+" "+name) } public static void main(String args[]) student s1=new student(111, "ABC"); s1.display(); student s2= new student(s1); s2.display(); } }</pre>	
	c)	Write a program to show the Hierarchical inheritance.	4 M
	Ans	<pre>import java.io.*; abstract class shape { float dim1,dim2; void getdata() { DataInputStream d=new DataInputStream(System.in); try { System.out.println("Enter the value of Dimension1: "); dim1=Float.parseFloat(d.readLine()); System.out.println("Enter the value of Dimension2: "); dim2=Float.parseFloat(d.readLine()); } catch(Exception e) { System.out.println("General Error"+e); } } void disp() { System.out.println("Dimension1= "+dim1); System.out.println("Dimension2= "+dim2); } abstract void area(); } class rectangle extends shape</pre>	4 M for correct program (Any relevant example can be consider)



```
{
double area1;
void getd()
{
super.getdata();
}
void area()
{
area1=dim1*dim2;
System.out.println("The Area of Rectangle is: "+area1);
}
}
class triangle extends shape
{
double area1;
void getd()
{
super.getdata();
}
void area()
{
area1=(0.5*dim1*dim2);
System.out.println("The Area of Triangle is: "+area1);
}
}
class methodover1
{
public static void main(String args[])
{
rectangle r=new rectangle();
System.out.println("For Rectangle");
r.getd();
r.disp();
r.area();
triangle t=new triangle();
t.getd();
t.disp();
t.area();
}
}
```

OR

```
class A
{
```




		<pre>public void methodA() { System.out.println("method of Class A"); } } class B extends A { public void methodB() { System.out.println("method of Class B"); } } class C extends A { public void methodC() { System.out.println("method of Class C"); } } class D extends A { public void methodD() { System.out.println("method of Class D"); } } class JavaExample { public static void main(String args[]) { B obj1 = new B(); C obj2 = new C(); D obj3 = new D(); //All classes can access the method of class A obj1.methodA(); obj2.methodA(); obj3.methodA(); } }</pre>	
	d)	Explain any four font methods with example.	4 M
	Ans	Font is a class that belongs to the java.awt package. Following are the methods of Font class:	2 M for any four- font method with



description and 2
M for example

Methods	Description
String getFamily()	Returns the name of the font family to which the invoking font belongs.
static Font getFont(String property)	Returns the font associated with the system property specified by <i>property</i> . null is returned if <i>property</i> does not exist.
String getFontName()	Returns the face name of the invoking font.
String getName()	Returns the logical name of the invoking font.
int getSize()	Returns the size, in points, of the invoking font.
int getStyle()	Returns the style values of the invoking font.
int hashCode()	Returns the hash code associated with the invoking object.
boolean isBold()	Returns true if the font includes the BOLD style value. Otherwise, false is returned.
boolean isItalic()	Returns true if the font includes the ITALIC style value. Otherwise, false is returned.
boolean isPlain()	Returns true if the font includes the PLAIN style value. Otherwise, false is returned.

Example:

```
import java.awt.*;
import java.applet.*;
public class Shapes extends Applet
{
    Font f,f1;
    String s,msg;
    String fname;
    String ffamily;
    int size;
    int style;
    public void init()
    {
        f= new Font("times new roman",Font.ITALIC,20);
        setFont(f);
        msg="is interesting";
        s="java programming";
        fname=f.getFontName();
        ffamily=f.getFamily();
        size=f.getSize();
        style=f.getStyle();
    }
}
```



		<pre>String f1=f.getName(); } public void paint(Graphics g) { g.drawString("font name"+fname,60,44); g.drawString("font family"+ffamily,60,77); g.drawString("font size "+size,60,99); g.drawString("fontstyle "+style,60,150); g.drawString("fontname "+f1,60,190); } } /*<applet code=Shapes.class height=300 width=300></applet>*/</pre>	
	e)	Write a program to append content of one file into another file.	4 M
	Ans	<pre>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(); } } }</pre>	4 M for correct program



		<pre>} } }</pre>																											
5.		Attempt any <u>TWO</u> of the following:	12 M																										
	a)	Explain vector with the help of example. Explain any 3 methods of vector class.	6 M																										
Ans	<ul style="list-style-type: none">Vector is a data structure that is used to store a collection of elements. Elements can be of all primitive types like int, float, Object, etc. Vectors are dynamic in nature and accordingly, grow or shrink as per the requirement.Vector Class in Java is found in the java.util package.Vector class is a child class of the AbstractList class and implements the List interface. Therefore, we can use all the methods of the List interface.Vectors are known to give ConcurrentModificationException when accessed concurrently at the time of modification.When a Vector is created, it has a certain capacity to store elements that can be defined initially. This capacity is dynamic in nature and can be increased or decreased.By definition, Vectors are synchronized, which implies that at a time, only one thread is able to access the code while other threads have to wait. <p>Vectors are created like arrays. It has three constructor methods</p> <pre>Vector list = new Vector(); //declaring vector without size Vector list = new Vector(3); //declaring vector with size Vector list = new Vector(5,2); //create vector with initial size and whenever it need to grows, it grows by value specified by increment capacity</pre> <p>Methods of Vector class:</p> <table><tr><th>Method Name</th><th>Task performed</th></tr><tr><td>list.firstElement()</td><td>It returns the first element of the vector.</td></tr><tr><td>list.lastElement()</td><td>It returns last element of the vector</td></tr><tr><td>list.addElement(item)</td><td>Adds the item specified to the list at the end.</td></tr><tr><td>list.elementAt(n)</td><td>Gives the name of the object at nth position</td></tr><tr><td>list.size()</td><td>Gives the number of objects present in vector</td></tr><tr><td>List.capacity()</td><td>This method returns the current capacity of the vector.</td></tr><tr><td>list.removeElement(item)</td><td>Removes the specified item from the list.</td></tr><tr><td>list.removeElementAt(n)</td><td>Removes the item stored in the nth position of the list.</td></tr><tr><td>list.removeAllElements()</td><td>Removes all the elements in the list.</td></tr><tr><td>list.insertElementAt(item, n)</td><td>Inserts the item at nth position.</td></tr><tr><td>List.contains(object element)</td><td>This method checks whether the specified element is present in the Vector. If the element is been found it returns true else false.</td></tr><tr><td>list.copyInto(array)</td><td>Copies all items from list of array.</td></tr></table> <p>Example: import java.util.*;</p>		Method Name	Task performed	list.firstElement()	It returns the first element of the vector.	list.lastElement()	It returns last element of the vector	list.addElement(item)	Adds the item specified to the list at the end.	list.elementAt(n)	Gives the name of the object at nth position	list.size()	Gives the number of objects present in vector	List.capacity()	This method returns the current capacity of the vector.	list.removeElement(item)	Removes the specified item from the list.	list.removeElementAt(n)	Removes the item stored in the nth position of the list.	list.removeAllElements()	Removes all the elements in the list.	list.insertElementAt(item, n)	Inserts the item at nth position.	List.contains(object element)	This method checks whether the specified element is present in the Vector. If the element is been found it returns true else false.	list.copyInto(array)	Copies all items from list of array.	<p>Correct explanation-2 M</p> <p>List of constructors and methods of vector class-2 M</p> <p>Example – 2 M</p>
Method Name	Task performed																												
list.firstElement()	It returns the first element of the vector.																												
list.lastElement()	It returns last element of the vector																												
list.addElement(item)	Adds the item specified to the list at the end.																												
list.elementAt(n)	Gives the name of the object at nth position																												
list.size()	Gives the number of objects present in vector																												
List.capacity()	This method returns the current capacity of the vector.																												
list.removeElement(item)	Removes the specified item from the list.																												
list.removeElementAt(n)	Removes the item stored in the nth position of the list.																												
list.removeAllElements()	Removes all the elements in the list.																												
list.insertElementAt(item, n)	Inserts the item at nth position.																												
List.contains(object element)	This method checks whether the specified element is present in the Vector. If the element is been found it returns true else false.																												
list.copyInto(array)	Copies all items from list of array.																												



		<pre>public class Main { public static void main(String args[]) { Vector v = new Vector(); v.addElement(new Integer(10)); v.addElement(new Integer(20)); v.addElement(new Integer(30)); v.addElement(new Integer(40)); v.addElement(new Integer(10)); v.addElement(new Integer(20)); System.out.println(v.size()); // display original size System.out.println("Initial Vector: " + v); v.removeElementAt(2); // remove 3rd element System.out.println("Current Vector: " + v); v.removeElementAt(3); // remove 4th element System.out.println("Current Vector: " + v); v.insertElementAt(11,2); // new element inserted at 3rd position System.out.println("Current Vector: " + v); System.out.println("Size of vector after insert delete operations: " + v.size()); } }</pre> <p>Output: 6 Initial Vector: [10, 20, 30, 40, 10, 20] Current Vector: [10, 20, 40, 10, 20] Current Vector: [10, 20, 40, 20] Current Vector: [10, 20, 11, 40, 20] Size of vector after insert delete operations: 5</p>	
	b)	Develop and Interest Interface which contains Simple Interest and Compound Interest methods and static final field of rate 25%. Write a class to implement those methods.	6 M
	Ans	<pre>import java.util.Scanner; import static java.lang.Math.pow; interface Interest { int roi=25; public void simpleInterest(float principle,float time); public void compoundInterest(float principle,float time); } public class InterestTest implements Interest { public void simpleInterest(float principle,float time) { float si = (principle*roi*time)/100;</pre>	<p>Creating correct interface with-2M</p> <p>Implementing interface-1M</p> <p>Calculating simple interest and compound interest-2M</p>



		<pre>System.out.println("Simple interested calculate by program is : " + si); } public void compoundInterest(float principle,float time) { double ci = principle * (Math.pow((1.0 +(roi/100)), time)) - principle; System.out.println("Compound interested calculate by program is : " + ci); } public static void main(String args[]) { InterestTest i1 = new InterestTest(); i1.simpleInterest(1000,2); i1.compoundInterest(1000,2); } }</pre>	Correct Main method-1M
	c)	Write a program that throws an exception called "NoMatchException" when a string is not equal to "India".	6 M
	Ans	<pre>import java.io.*; class NoMatchException extends Exception { private String str; NoMatchException(String str1) { str=str1; } public String toString() { return "NoMatchException --> String is not India and string is "+str; } } class Main { public static void main(String args[]) { String str1= new String("India"); String str2= new String("Australlia"); try { if(str1.equals("India")) System.out.println(" String is : "+str1); else throw new NoMatchException(str1); if(str2.equals("India")) System.out.println("\n String is : "+str2); } } }</pre>	Any Correct program – 6 M



		<pre>else throw new NoMatchException(str2); } catch(NoMatchException e) { System.out.println("\nCaught "+e); } } }</pre> <p>OUTPUT: String is : India Caught.... NoMatchException --> String is not India and string is Australlia</p>	
6.		Attempt any <u>TWO</u> of the following:	12 M
	a)	Write a program to print the sum, difference and product of two complex numbers by creating a class named "Complex" with separate methods for each operation whose real and imaginary parts are entered by user.	6 M
	Ans	<pre>// Java program to add and subtract two // complex numbers using Class import java.util.*; // User Defined Complex class class Complex { // Declaring variables int real, imaginary; // Empty Constructor Complex() { } // Constructor to accept // real and imaginary part Complex(int tempReal, int tempImaginary) { real = tempReal; imaginary = tempImaginary; } // Defining addComp() method // for adding two complex number Complex addComp(Complex C1, Complex C2) { // creating temporary variable</pre>	Correct program – 6 M



		<pre>Complex temp = new Complex(); // adding real part of complex numbers temp.real = C1.real + C2.real; // adding Imaginary part of complex numbers temp.imaginary = C1.imaginary + C2.imaginary; // returning the sum return temp; } // Defining subtractComp() method // for subtracting two complex number Complex subtractComp(Complex C1, Complex C2) { // creating temporary variable Complex temp = new Complex(); // subtracting real part of complex numbers temp.real = C1.real - C2.real; // subtracting Imaginary part of complex numbers temp.imaginary = C1.imaginary - C2.imaginary; // returning the difference return temp; } Complex productComp(Complex C1, Complex C2) { // creating temporary variable Complex temp = new Complex(); // product of of complex numbers //(a + ib) (c + id)= (ac - bd) + i(ad + bc). temp.real = ((C1.real*C2.real)-(C1.imaginary*C2.imaginary)); temp.imaginary = ((C1.real*C2.imaginary) + (C1.imaginary*C2.real)); // returning the difference return temp; } // Function for printing complex number void printComplexNumber() { System.out.println("Complex number: " + real + " + " + imaginary + "i"); } }</pre>	
--	--	--	--



```
// Main Class
public class Main
{

    // Main function
    public static void main(String[] args)
    {

        // First Complex number
        Complex C1 = new Complex(3, 2);

        // printing first complex number
        C1.printComplexNumber();

        // Second Complex number
        Complex C2 = new Complex(9, 5);

        // printing second complex number
        C2.printComplexNumber();

        // for Storing the sum
        Complex C3 = new Complex();

        // calling addComp() method
        C3 = C3.addComp(C1, C2);

        // printing the sum
        System.out.print("Sum of ");
        C3.printComplexNumber();

        // calling subtractComp() method
        C3 = C3.subtractComp(C1, C2);

        // printing the difference
        System.out.print("Difference of ");
        C3.printComplexNumber();

        // calling productComp() method
        C3 = C3.productComp(C1, C2);

        // printing the product
        System.out.print("product of ");
        C3.printComplexNumber();

    }
}
```

OUTPUT:

Complex number: 3 + 2i

Complex number: 9 + 5i



		Sum of Complex number: $12 + 7i$ Difference of Complex number: $-6 + -3i$ product of Complex number: $17 + 33i$	
	b)	i) Explain Errors and its types in detail. ii) Explain thread methods to set and get priority.	6 M
Ans		<p>An error is an issue in a program that prevents the program from completing its task. There are several types of errors that occur in Java, including syntax errors, runtime errors, and logical errors. They are</p> <ul style="list-style-type: none">● Syntax Errors or Compilation Errors: These occur when the code violates the rules of the Java syntax. These errors are usually caught by the Java compiler during the compilation phase.● Example of compile time error: public class Main { public static void main(String[] args) { int x = "5"; } } <p>OUTPUT:</p> <pre>Main.java:5: error: incompatible types: String cannot be converted to int int x = "5"; ^ 1 error</pre> <ul style="list-style-type: none">● Runtime Errors: These errors occur when the code encounters an unexpected behaviour during its execution. These errors are usually caused by flawed logic or incorrect assumptions in the code and can be difficult to identify and fix.● The most common run-time errors are:<ol style="list-style-type: none">a) Dividing an integer by zerob) Accessing an element that is out of bounds of an arrayc) Trying to store value into an array of an incompatible class or typed) Trying to illegally change status of threade) Attempting to use a negative size for an arrayf) Converting invalid string to a numberg) Accessing character that is out of bound of a string <p>These errors can be handled by exception handling with help of try-catch- final block</p>	Types of errors with example – 3 M and thread methods with any relevant/correct example – 3 M



(i) **Priorities in threads**

To get and set priority of a thread in java following methods are used,

1. **public final int getPriority():** java.lang.Thread.getPriority() method returns priority of given thread.
2. **public final void setPriority(int newPriority):** java.lang.Thread.setPriority() method changes the priority of thread to the value newPriority. This method throws IllegalArgumentException if value of parameter newPriority goes beyond minimum(1) and maximum(10) limit.

Example:

// Java Program to Illustrate Priorities in Multithreading
// via help of getPriority() and setPriority() method

// Importing required classes
import java.lang.*;

// Main class
class ThreadDemo extends Thread {

 // Method 1
 // run() method for the thread that is called
 // as soon as start() is invoked for thread in main()
 public void run()
 {
 // Print statement
 System.out.println("Inside run method");
 }

 // Main driver method
 public static void main(String[] args)
 {
 // Creating random threads
 // with the help of above class
 ThreadDemo t1 = new ThreadDemo();
 ThreadDemo t2 = new ThreadDemo();
 ThreadDemo t3 = new ThreadDemo();

 // Thread 1
 // Display the priority of above thread using getPriority() method
 System.out.println("t1 thread priority : " + t1.getPriority());

 // Thread 2
 // Display the priority of above thread
 System.out.println("t2 thread priority : " + t2.getPriority());

 // Thread 3
 System.out.println("t3 thread priority : " + t3.getPriority());
 }



		<pre>// Setting priorities of above threads by passing integer arguments t1.setPriority(2); t2.setPriority(5); t3.setPriority(8); System.out.println("t1 thread priority : "+ t1.getPriority()); System.out.println("t2 thread priority : "+ t2.getPriority()); System.out.println("t3 thread priority : "+ t3.getPriority()); // Main thread // Displays the name of currently executing Thread System.out.println("Currently Executing Thread : " + Thread.currentThread().getName()); System.out.println("Main thread priority : " + Thread.currentThread().getPriority()); // Main thread priority is set to 10 Thread.currentThread().setPriority(10); System.out.println("Main thread priority : " + Thread.currentThread().getPriority()); } }</pre> <p>OUTPUT: t1 thread priority : 5 t2 thread priority : 5 t3 thread priority : 5 t1 thread priority : 2 t2 thread priority : 5 t3 thread priority : 8 Currently Executing Thread : main Main thread priority : 5 Main thread priority : 10</p>	
	c)	Write a program to draw a chessboard in Java Applet.	6 M
	Ans	<pre>import java.applet.*; import java.awt.*; /*<applet code="Chess" width=600 height=600> </applet>*/ // Extends Applet Class public class Chess extends Applet { static int N = 10; // Use paint() method public void paint(Graphics g) {</pre>	Correct program – 6 M



			<pre>int x, y; for (int row = 0; row < N; row++) { for (int col = 0; col < N; col++) { // Set x coordinates of rectangle // by 20 times x = row * 20; // Set y coordinates of rectangle // by 20 times y = col * 20; // Check whether row and column are in even position // If it is true set Black color if ((row % 2 == 0) == (col % 2 == 0)) g.setColor(Color.BLACK); else g.setColor(Color.WHITE); // Create a rectangle with // length and breadth of 20 g.fillRect(x, y, 20, 20); } } }</pre>	
--	--	--	--	--