WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Java Programming **Subject Code:** **17515**

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.

Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	**a)**	**Attempt any THREE of the following:**	**12**
	(i)	**Describe any four features of Java.**	**4M**
		(Note: Any four features shall be considered)	
	Ans.	**Features of Java:**	
		i. Java is an object oriented language:- It follows all the principles of object oriented programming namely inheritance, polymorphism and abstraction. Multiple inheritance is possible with the concept of interface	
		ii. Java is both compiled and interpreted:- Most of the programming languages either uses a compiler or an interpreter. Java programs are to be compiled to get an intermediate byte code (a .class file) and then interpreted making it more secure and platform independent.	***Any four features 1M each***
		iii. Java is secure: • Java does not use pointer. • Java programs run inside a virtual machine • Classloader adds security by separating the package for the classes of the local file system from those that are imported	

		from network sources • Bytecode Verifier checks the code fragments for illegal code that can violate access right to objects • Security Manager determines what resources a class can access such as reading and writing to the local disk iv. Robust: Java uses strong memory management. The lack of pointers avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java v. Architecture-neutral: There is no implementation dependent features e.g. size of primitive types is fixed vi. Platform independent and Portable: java byte code can be carried to any platform vii. Distributed: Distributed applications can be created in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet viii. Multithreaded: A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications etc.	
	(ii) Ans.	**Describe file output stream class with one example.** A file output stream is an output stream for writing data to a File. FileOutputStream class is used to write bytes of data into a file. **Constructor: (Any one)** FileOutputStream(File file) This creates a file output stream to write to the file represented by the specified File object. FileOutputStream(String name) This creates an output file stream to write to the file with the specified name. ***Example:*** import java.io.*; class FileoutputstreamExample { public static void main(String a[]) throws IOException	**4M** ***Description 1M*** ***Example 3M***

		{
		File fin = new File("in.txt");
		File fout = new File("out.txt");
		FileInputStreamfis = null;
		FileOutputStreamfos = null;
		try {
		fis =new FileInputStream(fin);
		fos = new FileOutputStream(fout);
		intch;
		while((ch=fis.read())!=-1)
		{
		fos.write(ch);
		}
		} catch(Exception e)
		{
		}
		finally{
		if(fis != null)
		{
		fis.close();
		}
		if(fos != null)
		{
		fos.close();
		}
		}
		}	

	(iii) **Ans.**	**Enlist all mathematical function and write a program based on pow( ) function.** **Methods of Math class:** static double abs(double a) -returns absolute value of a double value static float abs(float a) – returns absolute value of float value static int abs(int a)-returns absolute value of an int value static long abs(long a)-returns absolute value of a long value static double exp(double a)- returns Euler's number e raised to the power of a double value static double max(double a, double b)- returns greater of two double values.	**4M** ***Any four List of methods - function 2M***

		static int max(int a, int b)- returns greater of two integer values.
		static float max(float a, float b)- returns greater of two float values.
		static long max(long a, long b)- returns greater of two long values.
		static double min(double a, double b)- returns smallest of two double values.
		static int min(int a, int b)- returns smallest of two integer values.
		static float min(float a, float b)- returns smallest of two float values.
		static long min(long a, long b)- returns smallest of two long values.
		static double pow(double a, double b)- returns the value of the first argument raised to the power of second argument.
		
		pow() example:
		import java.io.*;
		class PowEg {
		double findPow(double a, double b) {
		double val = java.lang.Math.pow(a,b);
		return val;
		}
		public static void main(String ar[]) {
		double a,b, ans = 0.0;
		PowEg p = new PowEg();
		try {
		BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));
		System.out.println("Enter a value");
		a = Double.parseDouble(br.readLine());
		System.out.println("Enter a value");
		b = Double.parseDouble(br.readLine());
		ans=p.findPow(a,b);
		} catch(Exception e) {
		System.out.println("Exception caught"+e);
		}
		System.out.println("Ans is "+ans);
		}
		}
			Program 2M

	(iv) Ans.	**Explain try – catch statement with one example.** An exception is an event which occurs during the execution of a program and disrupts the normal execution of the program. When an exception occurs in a method, it creates an exception object and hands it to the run time system. The object contains information about	**4M** ***Explanation 1M***

		the exception. The exception can be handled using try-catch block. The code that may create an exception should be enclosed in a try block. If an exception occurs within the try block, an exception object is created of the specific type, and is thrown. That exception is handled by an exception handler associated with it. To associate an exception handler with a try block, a catch block is specified after try block. A try block can have any number of catch.
		
		Example:
		import java.io.*;
		class ExceptionHandling {
		int num1, num2, answer;
		void acceptValues() {
		BufferedReader bin = new BufferedReader(new InputStreamReader(System.in));
		try {
		System.out.println("Enter two numbers");
		num1 = Integer.parseInt(bin.readLine());
		num2 = Integer.parseInt(bin.readLine());
		} catch(IOExceptionie) {
		System.out.println("Caught IOException"+ie);
		} catch(Exception e) {
		System.out.println("Caught the exception "+e);
		}
		}
		void doArithmetic() {
		acceptValues();
		try {
		answer = num1/num2;
		System.out.println("Answer is: "+answer);
		} catch(ArithmeticExceptionae) {
		System.out.println("Divide by zero"+ae);
		}
		}
		public static void main(String a[]) {
		ExceptionHandling e = new ExceptionHandling();
		e.doArithmetic();
		}
			Example 3M

WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Java Programming **Subject Code:** **17515**

1.	**b)**	**Attempt any ONE of the following:**	**6**
	(i)	**Explain constructor with its type, Give example of parameterized constructor.**	**6M**

	Ans.	Constructors are used to initialize an object as soon as it is created. Every time an object is created using the 'new' keyword, a constructor is invoked. If no constructor is defined in a class, java compiler creates a default constructor. Constructors are similar to methods but with two differences, constructor has the same name as that of the class and it does not return any value. The types of constructors are:
		1. Default constructor: when the constructor is not defined in the program, the java compiler creates a constructor. Such a constructor is called a default constructor.
		2. Parametrized constructor: Such constructor are defined in the program and consists of parameters. Such constructors can be used to create different objects with data members having different values.
		class Student
		{
		int roll_no;
		String name;
		Student(int r, String n)
		{
		roll_no = r;
		name=n;
		}
		void display()
		{
		System.out.println("Roll no is: "+roll_no);
		System.out.println("Name is : "+name);
		}
		public static void main(String a[])
		{
		Student s = new Student(20,"ABC");
		s.display();
		}
		}
		3. No argument constructor- when the defined constructor in the
			Definition and explanation of the types of constructors 4M
			Example 2M

		program does not contain any arguments, then it is called no argument constructor.	
	(ii) **Ans.**	**Enlist Build-in packages of java, explain any four in details with example.** **Package** in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. There are two types of packages. **Built in packages** **User defined packages- are packages defined by user.** Built in packages are part of java API. Some of the built in packages in java are: java.io-this package contains classes and interfaces for performing input and output operations and also serialization. Eg of classes/interfaces: BufferedReader, BufferedWriter, Serializable, InputStream, OutputStream java.util-contains utility classes/interfaces for implementing data structures, and other utility class like Date. Eg: Vector, Scanner, ArrayList, Date java.net-this package consists of classes that support networking operations. Eg: URL, URLConnection, Socket,ServerSocket java.awt-this package contains a large number of classes used for GUI. Eg: Graphics, Panel, Container java.applet-this package contains classes/interfaces that are used to create and use applets Eg: Applet,AppletContext,AudioClip java.lang-this package is automatically imported and contains language support classes like String, Thread,Math	**6M** ***Definition of package 1M*** ***List packages 1M*** ***Any 4 explanation 1M each***

WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Java Programming **Subject Code:** **17515**

2.	**a)**	**Attempt any TWO of the following:** **Write a program to implement following inheritance. Refer Figure No.1**	**16** **8M**

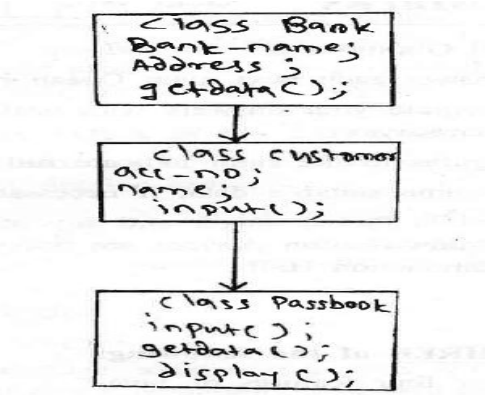


Fig. No. 1

Ans.

```
import java.io.*;
class Bank{
        String bank_name;
        String address;
        void getData(String b, String a){
                        bank_name = b;
                        address = a;
        }
}
class Customer extends Bank{
        int acc_no;
        String name;
        void input(String b, String a, int no, String n) {
                getData(b,a);
                acc_no = no;
                name=n;}
}

class Passbook extends Customer{
        void display() {
        System.out.println("Bank name is "+bank_name);
        System.out.println("Bank address is "+address);
        System.out.println("Account no of customer is "+acc_no);
```

Correct logic 6M

Correct syntax 2M

```
        System.out.println("Customer name is "+name);
        }
public static void main(String ar[]) {
try {
        String b,a,n;
        int no;
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        System.out.println("Enter bank name");
        b = br.readLine();
        System.out.println("Enter address");
        a = br.readLine();
        System.out.println("Enter customer name");
        n = br.readLine();
        System.out.println("Enter acc no");
        no = Integer.parseInt(br.readLine());
        Passbook p = new Passbook();
        p.input(b,a,no,n);
        p.display();
} catch(Exception e) {
        System.out.println("Exception caught"+e);}
}
}
```

b) **Write a java program for following Figure No.2** **8M**

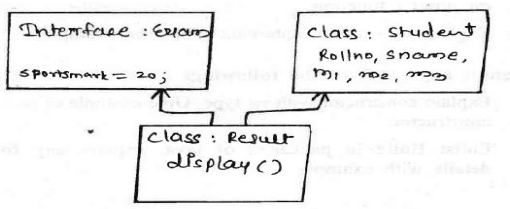


Fig. No. 2

Ans.

```
import java.io.*;
interface Exam {
        int sportsmark=20;
}
class Student {
        int rollno;
        String sname;
```

```
        int m1, m2, m3;
        Student(int r, String s, int m1, int m2, int m3) {
            rollno = r;
            sname=s;
            this.m1=m1;
            this.m2=m2;
            this.m3=m3;
        }
}
class Result extends Student implements Exam {
        Result(int r, String s, int m1, int m2, int m3) {
            super(r,s,m1,m2,m3);
        }
        void display() {
            int total = m1+m2+m3;
            double per = total/3;
            if(per <=80) {
                per = per+sportsmark;
            } else {
                double markstoadd = 100.00-per;
                per = per + markstoadd;
            }
            System.out.println("Roll No "+rollno);
            System.out.println("Name "+sname);
            System.out.println("Percentage "+per);
        }
public static void main(String ar[]){
            try {
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            System.out.println("Enter Student name");
            String n = br.readLine();
            System.out.println("Enter roll no");
            int no = Integer.parseInt(br.readLine());
            System.out.println("Enter Mark 1");
            int m1 = Integer.parseInt(br.readLine());
            System.out.println("Enter mark 2");
            int m2 = Integer.parseInt(br.readLine());
            System.out.println("Enter mark 3");
```

Correct logic 6M

Correct syntax 2M

```
                int m3 = Integer.parseInt(br.readLine());
                Result r = new Result(no,n,m1,m2,m3);
                r.display();
                } catch(Exception e) {
                        System.out.println("Exception caught"+e);
                }
        }
}
```

c) **Write an applet program that accepts string as a input using <param> tag and reverse the string and display it on status window.** **8M**

Ans.

```
import java.applet.*;
import java.awt.*;
/*<applet code = Question2c.class height = 400 width = 400>
<param name = "string1" value = "Hello">
</applet>*/
public class Question2c extends Applet {
        String str1;
        public void init() {
                str1 = new
StringBuffer(getParameter("string1")).reverse().toString();
        }
        public void paint(Graphics g) {
                showStatus(str1);
        }
}
```

Correct logic 6M

Correct syntax 2M

OR

```
import java.applet.*;
import java.awt.*;
/*<applet code = ReverseStringApplet.class height = 400 width =
400>
<param name = "string1" value = "Hello">
</applet>*/
public class ReverseStringApplet extends Applet {
        String str1;
        String rev="";
        public void init() {
                String str = getParameter("string1");
                char ch[]=str.toCharArray();
```

```
                for(int i=ch.length-1;i>=0;i--){
                        rev+=ch[i];
                }
        }
        public void paint(Graphics g) {
                showStatus(rev);
        }
}
```

3. **a) Ans.** **Attempt any FOUR of the following:** **16**

State any four methods of wrapper class. **4M**

Four methods of Wrapper classes:

1) valueOf()
2) xxxValue()
3) parseXxx()
4) toString()

Method	Use
valueOf()	Every wrapper class except Character class contains a static valueOf() method to create Wrapper class object for given String. Integer i=Integer.valueOf("10");
xxxValue()	xxxValue() methods are used to get the primitive for the given Wrapper Object. Every number type Wrapper class(Byte, Short, Integer, Long, Float, Double) contains the following 6 methods to get primitive for the given Wrapper object: syntax : public static datatype parseXxx(String s); 1. public byte byteValue() 2. public short shortValue() 3. public int intValue() 4. public long longValue() 5. public float floatValue() 6. public float doubleValue() Any one method can be described out of 6
parseXxx()	parseXxx() methods can be used to convert String to primitive. Xxx contains data type as follows: parseInt(String str) parseFloat(String str)

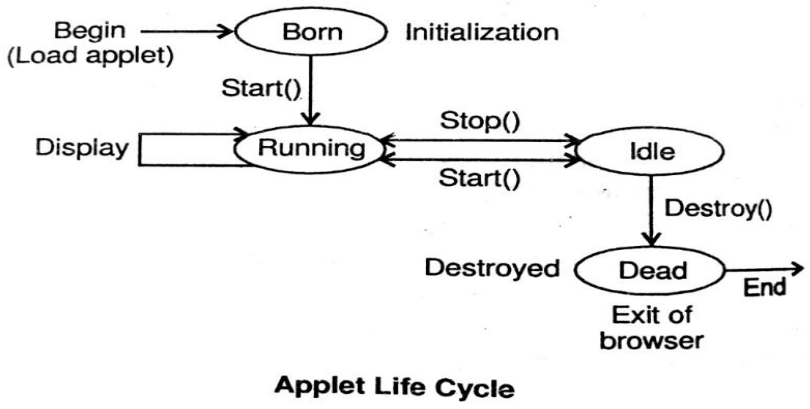
Any four methods list 1M each

		<table><tr><td></td><td>parseLong(Strung str) parseDouble(String str) Any one method can be described out of 4</td></tr><tr><td>toString()</td><td>Every Wrapper class contains the following toString() method to convert primitive to String. Syntax: public String toString(); for eg: Integer.toString(5);</td></tr></table>	

	b)
	
	Ans.
		Write a program to find out whether entered no. is prime or not.
		(Note: Any other logic shall be considered)
		import java.util.*;
		class prime
		{
		public static void main(String args[])
		{
		int flag=0;
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter a number :");
		int n=sc.nextInt();
		for(int i=2;i<n;i++)
		{
		if(n%i==0)
		{
		flag=1;
		break;
		}
		}
		if(flag==0)
		System.out.println(n +" is prime");
		else
		System.out.println(n +" is not prime");
		}
		}
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

	c)	**Write a program to copy contents of one file to another file using character stream classes.** ***(Note: Any other logic shall be considered)***	**4M**

	Ans.	import java.io.*;
		class FileCopyExample {
		public static void main(String[] args) throws IOException
		{
		FileReaderfr = new FileReader("input.txt");
		FileWriterfw = new FileWriter("output.txt");
		int c=0;
		while(c!=-1)
		{
		c=fr.read();
		fw.write(c);
		}
		fr.close();
		fw.close();
		System.out.println("file copied");
		}
		}
			Correct logic 2M
			
			Correct syntax 2M

	d) Ans.	**Explain Life-cycle of an applet.** Java applet inherits features from the class Applet. Thus, whenever an applet is created, it undergoes a series of changes from initialization to destruction. Various stages of an applet life cycle are depicted in the figure below:  **Applet Life Cycle** **Initial State:** When a new applet is born or created, it is activated by calling init() method. At this stage, new objects to the applet are created, initial values are set, images are loaded and the colors of the images are set. An applet is initialized only once in its lifetime.	**4M** ***Diagram 1M***

		It's general form is:
		public void init( )
		//Action to be performed
		}
		
		Running State: An applet achieves the running state when the system calls the start() method. This occurs as soon as the applet is initialized. An applet may also start when it is in idle state. At that time, the start() method is overridden.
		It's general form is:
		public void start( )
		{
		//Action to be performed
		}
		
		Idle State: An applet comes in idle state when its execution has been stopped either implicitly or explicitly. An applet is implicitly stopped when we leave the page containing the currently running applet. An applet is explicitly stopped when we call stop() method to stop its execution.
		It's general form is:
		public void stop()
		{
		//Action to be performed
		}
		
		Dead State: An applet is in dead state when it has been removed from the memory. This can be done by using destroy() method.
		
		It's general form is:
		public void destroy( )
		{
		//Action to be performed
		}
		
		Display State (paint ()): Apart from the above stages, Java applet also possess paint( )method. The paint() method is used for applet display on the screen. This method helps in drawing, writing and creating colored backgrounds of the applet. It takes an argument of
			Explanation 3M

the graphics class. To use The graphics, it imports the package java.awt.Graphics

e) Ans. **Compare between string and string buffer class.** **4M**

Sr. No.	String	String buffer class
1	String is a major class	StringBuffer is a peer class of String
2	The length of the String object is fixed.	The length of the StringBuffer object is flexible.(can be changed)
3	String object is immutable.	StringBuffer object is mutable.
4	It is slower during concatenation.	It is faster during concatenation.
5	String object cannot be modified.	StringBuffer object contents can be modified.
6	String object can be created without calling a constructor of String class. Eg: String str="abc";	StringBuffer object needs constructor to initialize created object. Eg: StringBuffer str=new StringBuffer("abc");

Any four points 1M each

4. **a) (i) Ans.** **Attempt any THREE of the following:** **12**
Explain robust and secure feature of java. **4M**

Robust: Robust simply means strong. Java is robust because:

- It uses strong memory management.
- It is portable across many operating systems
- There is a lack of pointers that avoids security problems.
- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Robust 2M

Secure: Unlike C, there are no pointers in Java. The basic use of Pointers is to refer to the actual memory location where the value is stored, and hence enabling the programmer to further modify the

Secure 2M

		actual value. As java does not consist of pointers, there is no way one can refer to the actual value, and hence the value remains protected or unaltered. Due to these reasons, Java is known as a Secure language.	
	(ii)	**Explain the following terms w.r.t exception handling:** **(1) Try-catch** **(2) Throw** **(3) Throws** **(4) Finally**	**4M**

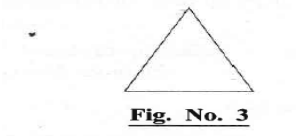
	Ans.	**(1) Try-catch:** Program statements that you want to monitor for exceptions are contained within a try block. If an exception occurs within the **try** block, it is thrown. Your code can catch this exception (using **catch**) and handle it in some rational manner. System-generated exceptions are automatically thrown by the Java runtime system. A catch block immediately follows the try block. The catch block can have one or more statements that are necessary to process the exception.
		
		Syntax:
		try
		{
		// block of code to monitor for errors
		}
		catch (*ExceptionType1 exOb*)
		{
		// exception handler for *ExceptionType1*
		}
		
		(2) Throw: It is mainly used to throw an instance of user defined exception.
		
		Example:
		throw new myException("Invalid number");
		assuming myException as a user defined exception
		
		(3) Throws: This keyword can be used along with method declaration with any exception so that each statement from that method is monitored for the errors and if there is any errors, then the system shows its own error message. Throws does not require separate try catch block to monitor the errors.
			Each term 1M

		Syntax:
		datatype method() throws ExceptionType
		{
		//method body;
		}
		Example :
		public static void main(String args[]) throws IOException
		
		(4) Finally: Finally block is a block that is used to execute important code such as closing connection, stream etc. Java finally block is always executed whether exception is handled or not. Java finally block follows try or catch block.
		Syntax:
		finally
		{
		// block of code to be executed before try block ends
		}	

	(iii) Ans.	**Explain the use of ternary (?:) operator with example.**
		Java ternary operator is the only conditional operator that takes three operands. Java ternary operator is a one liner replacement for if-then-else statement and used a lot in java programming.
		The first operand in java ternary operator should be a boolean or a statement with boolean result. If the first operand is **true** then java ternary operator returns second operand else it returns third operand.
		Syntax of java ternary operator is:
		result = testStatement ? value1:value2;
		If testStatement is true then value1 is assigned to result variable else value2 is assigned to result variable. java ternary operator can be used to avoid if-then-else and switch case statements. This way we can reduce the number of lines of code in java program.
		Example:
		class test
		{
		public static void main(String args[])
		{
		int a=5;
		String result="";
		result = (a>0 ?"positive" : "negative");
		System.out.println(result);
			4M
			
			Explanation 2M
			
			Example 2M

		} }	
	(iv) **Ans.**	**Explain following thread methods with suitable example.** **(1) setpriority (int max)** **(2) getpriority( )** **(1) setpriority (int max):** The **setPriority()** method of thread class is used to change the thread's priority. Every thread has a priority which is represented by the integer number between 1 to 10. Thread class provides 3 constant properties: 1. **public static int MIN_PRIORITY:** It is the minimum priority of a thread. The value of it is 1. 2. **public static int NORM_PRIORITY:** It is the normal priority of a thread. The value of it is 5. 3. **public static int MAX_PRIORITY:** It is the maximum priority of a thread. The value of it is 10. We can also set the priority of thread between 1 to 10. This priority is known as custom priority or user defined priority. ***Syntax:*** public final void setPriority(int a) **(2) getpriority( ):** The **getPriority()** method of thread class is used to check the priority of the thread. When we create a thread, it has some priority assigned to it. Priority of thread can either be assigned by the JVM or by the programmer explicitly while creating the thread. The thread's priority is in the range of 1 to 10. The default priority of a thread is 5. ***Syntax:*** public final int getPriority() ***Example:*** public class PriorityExample extends Thread { public void run() { System.out.println("Priority of thread is: "+Thread.currentThread().g	**4M** ***setPriority (int max) 2M*** ***getPriority ( ) 2M***

		etPriority());
		}
		public static void main(String args[])
		{
		PriorityExample t1=new PriorityExample ();
		t1.setPriority(Thread.MAX_PRIORITY);
		t1.start();
		}
		}	

4.	**b)**	**Attempt any ONE of the following:**	**6**
	(i)	**Write a program to generate following output using drawLine( ) method. Refer Figure No.3**  **Fig. No. 3**	**6M**

	Ans.	import java.awt.*;
		import java.applet.*;
		public class triangle extends Applet
		{
		public void paint(Graphics g)
		{
		g.drawLine(35,10,50,100);
		g.drawLine(25,100,50,100);
		g.drawLine(25,100,35,10);
		}
		}
		/*<applet code=triangle width=100 height=100>
		</applet>*/
			Correct logic 2M
			Use of drawLine( ) 3M
			Applet tag 1M

	(ii)	**Explain in detail garbage collection mechanism in java.**	**6M**
	Ans.	Java garbage collection is the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory. Java garbage collection is an automatic process.	***Need 2M***

		In C/C++, programmer is responsible for both creation and destruction of objects. Usually programmer neglects destruction of useless objects. Due to this negligence, at certain point, for creation of new objects, sufficient memory may not be available and entire program will terminate abnormally causing OutOfMemoryErrors**.**
		But in Java, the programmer need not to care for all those objects which are no longer in use. Garbage collector destroys these objects.
		Main objective of Garbage Collector is to free heap memory by destroying unreachableobjects**.**
		Just before destroying an object, Garbage Collector calls finalize() method on the object to perform cleanup activities. Once finalize() method completes, Garbage Collector destroys that object.
		
		Syntax :
		protected void finalize() throws Throwable
		{
		//code ;
		}
		Based on our requirement, we can override finalize() method for perform our cleanup activities like closing connection from database.
			Working 3M
			
			Finalize ( ) 1M

5.		**Attempt any <u>TWO</u> of the following:**	**16**
	a)	**How synchronization is achieved in multi threading? Explain with suitable example.**	**8M**
	Ans.	**Synchronization:** When two or more threads need access to a shared resource, they need some way to ensure that the resource will be used by only one thread at a time. The process by which this is achieved is called synchronization. Synchronization is used when we want to maintain consistency if multiple threads require an access to an object. ***Example:*** class Callme { void call(String msg) { System.out.print("[" +msg); try {	***Description 3M*** ***Example 5M***

```
                        Thread.sleep(1000);
                }
                catch(InterruptedException e)
                {
                        System.out.println("Interrupted ");
                }
                System.out.print("]");
        }
}
class Caller implements Runnable
{
        String msg;
        Callme target;
        Thread t;
        public Caller(Callmetarg,String s)
        {
                target=targ;
                msg=s;
                t=new Thread(this);
                t.start();
        }
        public void run()
        {
                synchronized(target)
                {
                        target.call(msg);
                }
        }
}

class Synch
{
        public static void main(String args[])
        {
                Callme target=new Callme();
                Caller ob1=new Caller(target,"Hello");
                Caller ob2=new Caller(target,"Synchronized");
                try
                {
```

```
                        ob1.t.join();
                        ob2.t.join();
                    }
                    catch(InterruptedException e)
                    {
                        System.out.println("Interrupted ");
                    }
                }
}
```

b) **Write a program to create user defined exception "Minimum Balance" if the account balance is less than Rs.1000/-.** **8M**

Ans.

```
import java.io.*;
class MinimumBalance extends Exception
{
MinimumBalance(String s)
{
super(s);
}
}
class Minbal
{
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
String name;
int bal;
void getdata() throws MinimumBalance
{
try
{
        System.out.println("Enter name");
        name=br.readLine();
        System.out.println("Enter Balance");
        bal = Integer.parseInt(br.readLine());
        if(bal<10000)
        {
throw new MinimumBalance("Account balance is less than Minimum
Balance");
}
        else
```

Creation of user defined exception 4M

Throw and Catch Exception 4M

		{
		System.out.println("Successfully received data ");
		}
		}
		catch(Exception ex)
		{
		System.out.println("Exception occured: "+ex);
		}
		}
		public static void main(String are[])
		{
		Minbal m = new Minbal();
		try
		{
		m.getdata();
		}
		catch(Exception e)
		{}
		}
		}	

	c)
	
	Ans.
		Write an applet program to draw a rectangle filled with different colors randomly on the applet window.
		import java.applet.Applet;
		import java.awt.Graphics;
		import java.awt.*;
		import java.lang.Math;
		import java.util.Random;
		
		public class rectangle extends Applet {
		public void init()
		{
		// set size
		setSize(400, 400);
		repaint();
		}
		// paint the applet
		public void paint(Graphics g)
		{
		// set Color for rectangle
			8M
			
			Creating applet for drawing rectangle 5M
			
			Logic for random color filling 3M

```
        Random rand = new Random();
           int red, green, blue;
           red = rand.nextInt(256);
           green = rand.nextInt(256);
           blue = rand.nextInt(256);
           g.setColor(new Color(red, green, blue));
         // draw a rectangle
        g.fillRect(100, 100, 200, 200);
    }
}
```

6. **Attempt any FOUR of the following:** **16**

a) **How command line argument passed to the java program, explain with suitable example.** **4M**

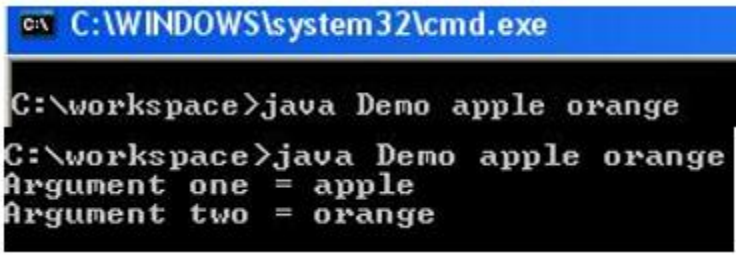
Ans. Command **line arguments** is a methodology which user will give inputs through the console using commands. Command Line Argument is information passed to the program when you run the program. The passed information is stored as a string array in the main method. Later, you can use the command line arguments in your program. ***Description 2M***

Example While running a class Demo, you can specify command line arguments as
java Demo arg1 arg2 arg3 …

Example

```
class Demo{
    public static void main(String b[]){
        System.out.println("Argument one = "+b[0]);
        System.out.println("Argument two = "+b[1]);
    }
}
```

Example 2M



b) Give the difference between Buffered Reader and Buffered Writer class. — **4M**

Ans.

Sr. No.	Buffered Reader	Buffered Writer
1	It extends from the reader class.	It extends from the writer class.
2	Its key constructor arguments are Reader.	Its key constructor arguments are Writer
3	It provides the read() and readLine() methods.	It provides the close(), flush(), newLine(), and write() methods.
4	It implements the mark() and reset() methods.	It does not implement the mark() and reset() methods.

Any four differences 1M each

c) Write an applet program for each of the following graphics method: — **4M**

(i) drawOval ( )
(ii) drawLine( )

Ans.

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
public class DrawOval extends Applet{
public void paint(Graphics g){

setForeground(Color.red);
g.drawOval(10,10,50,100);
}
}
/*
<applet code="DrawOval.class" width=500 height=500>
</applet>
*/
```

drawOval Program 2M

Program for DrawLine:

```
/*
<applet code="DrawLine.class" width=200 height=200>
</applet>
```

drawLine Program 2M

		*/
		import java.applet.Applet;
		import java.awt.Graphics;
		public class DrawLine extends Applet{
		public void paint(Graphics g){
		g.drawLine(10,10,50,50);
		g.drawLine(10,50,10,100);
		g.drawLine(10,10,50,10);
		}
		}	

	d)
	
	Ans.
		Design a package containing a class which defines a method to find area of circle. Import it in java application to calculate area of circle.
		
		File 1:
		package pack;
		public class A_circle
		{
		public void areaofcircle(int r)
		{
		System.out.println(3.14*r*r);
		}
		}
		
		File 2:
		import pack.A_circle;
		import java.io.*;
		class aoc
		{
		public static void main(String are[])
		{
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		int r;
		try
		{
		System.out.println("Enter a radius of circle");
		r = Integer.parseInt(br.readLine());
			4M
			
			Creating user defined package for area calculation 2M
			
			Using user defined package 2M

		A_circle c = new A_circle();
		c.areaofcircle(r);
		}
		catch(Exception ex)
		{}
		}
		}	

	e)
	
	Ans.
		Write a program to find out sum of even and odd numbers. Use suitable range.
		class sum
		{
		public static void main(String are[])
		{
		int i,j,osum=0,esum=0;
		for(i=0;i<=10;i++)
		{
		if((i%2)==0)
		{
		esum=esum+i;
		}
		}
		for(j=0;j<=10;j++)
		{
		if((j%2)==1)
		{
		osum=osum+j;
		}
		}
		System.out.println("Sum of odd number is " + osum);
		System.out.println("Sum of even number is " + esum);
		}
		}
			4M
			
			Logic for computing sum of even number and odd numbers 2M respectively