MODEL ANSWER

WINTER - 2017 EXAMINATION

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

Important Instructions to examiners:

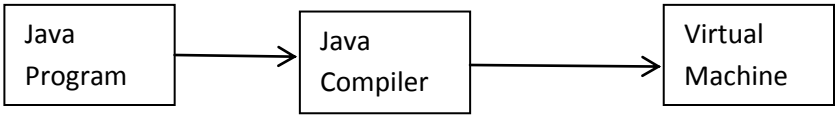
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling 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.

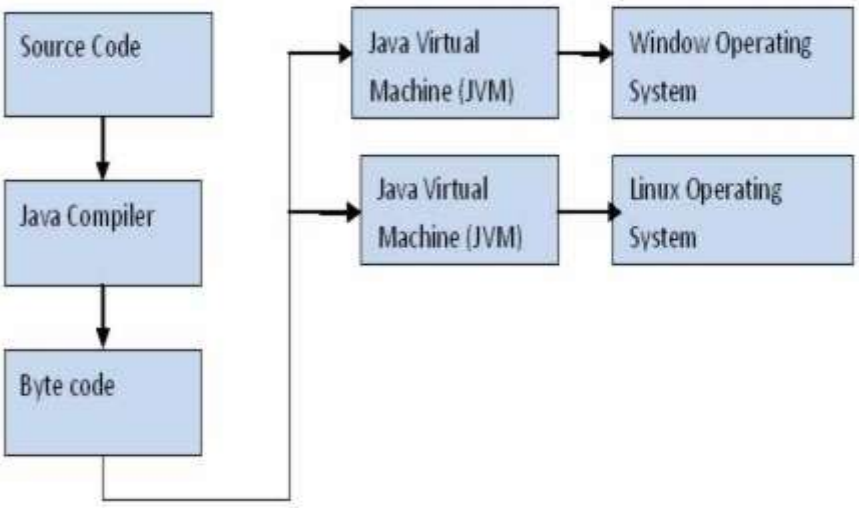
Q. No.	Sub Q.N.	Answer	Marking Scheme

1.	**(A) (a) Ans.**	**Attempt any THREE of the following:**
		Explain inheritance and polymorphism features of Java.
		Inheritance: inheritance is the process by which one object acquires the properties of another object. It supports the concept of hierarchical classification. Without the use of hierarchies, each object would need to define all the characteristics explicitly. By use of inheritance, an object need only define those qualities that make it unique within its class. It can inherit the general attributes from its parent. It is the inheritance mechanism that makes it possible for one object to be a specific instance of a more general case.
		For e.g.: Parrot is a classification of Bird. Therefore Parrot is a subclass of Bird. Parrot inherits a lot many features of the class Bird plus some additional features.
		class Bird {
		
		}
		class Parrot extends Bird {
		}
			3x4=12
			4M
			
			2M each for inheritance and polymorphism

		Polymorphism: it is a feature that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. By this concept it is possible to design a generic interface to a group of related activities.
		For E.g:- void add(int a, int b){
		int sum = a+b;
		System.out.println(sum);
		}
		void add(float a, float b){
		float sum = a+b;
		System.out.println(sum);
		}	

	(b) Ans.	**Write any two methods of array list class with their syntax.** **booleanadd(E e):** Appends the specified element to the end of this list **void add(int index, E element)** Inserts the specified element at the specified position in this list. **void clear():**Removes all of the elements from this list **Objectclone**():Returns a shallow copy of this ArrayList instance **booleancontains(Object o):** Returns true if this list contains the specified element. **Eget(int index):** Returns the element at the specified position in this list. **intindexOf(Object o)**: Returns the index position of the element in the list **booleanisEmpty()** :Returns true if the list is empty. **intlastIndexOf(Object o):** Returns the index of the last occurrence of the object specified. **boolean remove(Object o):** Removes the first occurrence of the object from the list if it is present.	**4M** ***Any two methods with proper syntax (return type and parameters) 2M each***

		int size(): returns the number of elements in the list.	
	(c) **Ans.**	**Why java became platform independent language? Explain.** ***(Note: Any other correct diagram may also be considered)*** Java is a platform independent language. This is possible because when a java program is compiled, an intermediate code called the byte code is obtained rather than the machine code. Byte code is a highly optimized set of instructions designed to be executed by the JVM which is the interpreter for the byte code. Byte code is not a machine specific code. Byte code is a universal code and can be moved anywhere to any platform. JVM is a virtual machine which exists inside the computer memory and is a simulated computer within a computer which does all the functions of a computer. Only the JVM needs to be implemented for each platform. Although the details of the JVM will defer from platform to platform, all interpret the same byte code.  Source Code Byte Code Process of Compilation  Virtual Machine Real Machine Process of converting byte code into machine code OR	**4M** ***Explanation 3M*** ***1M for diagram***



(d) **Write a program to input name and balance of customer and thread an user defined exception if balance less than 1500.** **4M**

Ans.

```
import java.io.*;
class MyException extends Exception{
MyException(String str) {
super(str);
}
}

class AccountDetails {
public static void main(String a[]) {
try {
BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
String name;
int balance;
System.out.println("Enter name");
name = br.readLine();
System.out.println("Enter balance");
balance = Integer.parseInt(br.readLine());
try {
if(balance<1500) {
throw new MyException("Balance is less");
```

Correct logic 3M

Syntax 1M

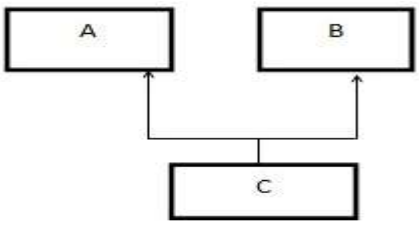
		} else {
		System.out.println("Everything alright");
		}
		} catch(MyException me) {
		System.out.println("Exception caught"+me);
		}
		} catch(Exception e) {
		System.out.println("Exception caught"+e);
		}
		}
		}	

1.	**(B)**
	(a)
	
	Ans.
		Attempt any ONE of the following:
		Design an applet which display equals size three rectangle one below the other and fill them with orange, white and green color respectively.
		import java.awt.*;
		importjava.applet.*;
		/*
		<applet code = DisplayRectangle.class height = 300 width = 300></applet>
		*/
		public class DisplayRectangle extends Applet {
		public void init() {
		setBackground(Color.PINK);
		}
		public void paint(Graphics g) {
		
		g.setColor(Color.ORANGE);
		g.fillRect(40,40,40,30);
		g.setColor(Color.WHITE);
		g.fillRect(40,90,40,30);
		g.setColor(Color.GREEN);
		g.fillRect(40, 140,40,30);
		}
		}
		OR
		
		import java.awt.*;
		importjava.applet.*;
			1x6=6
			6M
			
			Correct logic 4M
			
			Correct syntax 2M

```
/*
<applet code = DisplayRectangle.class height = 300 width = 300></applet>
*/
public class DisplayRectangle extends Applet {
public void paint(Graphics g) {
g.setColor(Color.ORANGE);
g.fillRect(40,40,40,30);
g.setColor(Color.BLACK);
g.drawRect(40,90,40,30);
g.setColor(Color.GREEN);
g.fillRect(40, 140,40,30);
}
}
```

(b) **What is the multiple inheritance? Write a java program to implement multiple inheritance.** **6M**

Ans. **Multiple inheritance:** is a feature in which a class inherits characteristics and features from more than one super class or parent class. ***Explanation with diagram 2M***



Java cannot have more than one super class. Therefore interface is used to support multiple inheritance in java. Interface specifies what a class must do but not how it is done.

Eg: interface MyInterface{
int strength=60;
void method1();
}
class MyBaseClass {
String str;
MyBaseClass(String str) {
this.str = str;
}

Correct logic 2M

Correct syntax 2M

		public void display() {
		System.out.println("Class: "+str);
		}
		}
		public class MyClass extends MyBaseClass implements MyInterface
		{
		float total;
		MyClass(String str, float t) {
		super(str);
		total = t;
		}
		public void method1() {
		float avg = total/strength;
		System.out.println("Avg is "+avg);
		}
		public static void main(String a[]) {
		MyClass c = new MyClass("Fifth Sem",1300.0f);
		c.display();
		c.method1();
		}
		}	

2.		**Attempt any TWO of the following:**	**2x8=16**
	(a)	**Define a class person with data member as Aadharno, name, Panno implement concept of constructor overloading. Accept data for 5 object and print it.**	**8M**

	Ans.	import java.io.*;
		class Person {
		intAadharno;
		String name;
		String Panno;
		Person(intAadharno, String name, String Panno) {
		this.Aadharno = Aadharno;
		this.name = name;
		this.Panno = Panno;
		}
		Person(intAadharno, String name) {
		this.Aadharno = Aadharno;
		this.name = name;
		Panno = "Not Applicable";
			Correct logic 5M
			
			Syntax 3M

```
}
void display() {
System.out.println("Aadharno is :"+Aadharno);
System.out.println("Name is: "+name);
System.out.println("Panno is :"+Panno);
}
public static void main(String ar[]) {
BufferedReaderbr = new
BufferedReader(newInputStreamReader(System.in));
Person p, p1, p2, p3, p4;
int a;
String n, pno;
try {
System.out.println("Enter Aadhar no");
a = Integer.parseInt(br.readLine());
System.out.println("Enter name");
n = br.readLine();
System.out.println("Enter panno");
pno = br.readLine();
p = new Person(a,n,pno);
System.out.println("Enter Aadhar no");
a = Integer.parseInt(br.readLine());
System.out.println("Enter name");
n = br.readLine();
System.out.println("Enter panno");
pno = br.readLine();
p1 = new Person(a,n,pno);
System.out.println("Enter Aadhar no");
a = Integer.parseInt(br.readLine());
System.out.println("Enter name");
n = br.readLine();
p2 = new Person(a,n);
System.out.println("Enter Aadhar no");
a = Integer.parseInt(br.readLine());
System.out.println("Enter name");
n = br.readLine();
p3 = new Person(a,n);
System.out.println("Enter Aadhar no");
a = Integer.parseInt(br.readLine());
```

		System.out.println("Enter name");
		n = br.readLine();
		System.out.println("Enter panno");
		pno = br.readLine();
		p4 = new Person(a,n,pno);
		p.display();
		p1.display();
		p2.display();
		p3.display();
		p4.display();
		} catch(Exception e) {
		System.out.println("Exception caught"+e);
		}
		}
		}	

	(b)
	
	Ans.
		What is package? How do we create it? Give the example to create and to access package.
		Package is a name space that organizes a set of related classes and interfaces. It also provides access protection and removes name collision.
		Packages can be categorized into two: - built-in and user defined.
		
		Creation of user defined package:
		To create a package a physical folder by the name of the package should be created in the computer.
		
		Example: we have to create a package myPack, so we create a folder d:\myPack
		The java program is to be written and saved in the folder myPack.
		The first line in the java program should be package <name>;
		followed by imports and the program logic.
		package myPack;
		importjava.util.*;
		public class Myclass {
		public void myMethod() {
		System.out.println("Inside package");
		}
		}
			8M
			
			Definition of package 2M
			
			Creation of package and its example 3M

		Access user defined package:
		To access a user defined package, we need to import the package in our program. Once we have done the import we can create the object of the class from the package and thus through the object we can access the instance methods.
		
		import myPack.*;
		public class MyClassExample{
		public static void main(String a[]) {
		Myclass c= new Myclass();
		c.myMethod();
		}
		}
			Use of package and its example 3M

	(c) **Ans.**	**Give the syntax of following methods of graphics class. Explain their use with suitable program:** **(i) drawRoundReel( )** **(ii) drawPolygon( )** **(iii) drawOval( )** **(iv) drawstring( )** ***(Note: Solution is given for drawRoundRect() method)*** **(i) void drawRoundRect( ):** void drawRoundRect(int x, int y, int width, int height, intarcwidth, intarcheight) - draws an outlined round cornered rectangle.int x and y represents the top left corner of the rectangle. Width and height represents the length and breadth of the rectangle. The arcwidth and archeight represents the horizontal and vertical diameter of the arc at the four corners. **(ii) voiddrawPolygon( ):**void drawPolygon(int x[], int y[], int n)- draws a polygon with the arrays of x coordinates and y coordinates and the number of points specified by n **OR** voiddrawPolygon(Polygon p)- draws a polygon defined by the specified polygon object. **(iii) void drawOval( ):** void drawOval(int x, int y, int width, int height) - draws an outline of an oval. **(iv) void drawstring( ):**void drawstring(String str, int x, int y)	**8M** ***1M each for syntax***

		- draws the string specified using the coordinates specified by x and y.
		
		import java.awt.*;
		importjava.applet.*;
		/*
		<applet code = MyApplet.class height = 300 width = 300></applet>
		*/
		public class MyApplet extends Applet {
		public void paint(Graphics g) {
		g.drawRoundRect(40, 40, 40, 30, 10,10);
		int x[] = {40,80,120};
		int y[] = {90,100,95};
		g.drawPolygon(x,y,3);
		g.drawOval(40, 110,40,30);
		g.drawString("My Applet",40, 160);
		}
		}
			1M each for program OR 4M if one program with all the methods

3.	**(a)**
	
	Ans.
		Attempt any FOUR of the following:
		Describe following string class method with example:
		(i) compareTo( )
		(ii) equalsIgnoreCase( )
		(i) compareTo( ):
		Syntax: intcompareTo(Object o)
		or
		intcompareTo(String anotherString)
		There are two variants of this method. First method compares this String to another Object and second method compares two strings lexicographically.
		Eg. String str1 = "Strings are immutable";
		String str2 = "Strings are immutable";
		String str3 = "Integers are not immutable";
		int result = str1.compareTo(str2);
		System.out.println(result);
		result = str2.compareTo(str3);
		System.out.println(result);
			4x4=16
			4M
			
			Each description 1M
			
			Each syntax 1M

		(ii) equalsIgnoreCase( ):
		public boolean equalsIgnoreCase(String str)
		
		This method compares the two given strings on the basis of content of the string irrespective of case of the string.
		Example:
		String s1="javatpoint";
		String s2="javatpoint";
		String s3="JAVATPOINT";
		String s4="python";
		System.out.println(s1.equalsIgnoreCase(s2));//true because content and case both are same.
		System.out.println(s1.equalsIgnoreCase(s3));//true because case is ignored.
		System.out.println(s1.equalsIgnoreCase(s4));//false because content is not same.	

	(b)
	
	Ans.
		Write a program to copy contents of one file to another. Using byte stream classes.
		class fileCopy
		{
		public static void main(String args[]) throws IOException
		{
		FileInputStream in= new FileInputStream("input.txt");
		FileOutputStream out= new FileOutputStream("output.txt");
		int c=0;
		try
		{
		while(c!=-1)
		{
		c=in.read();
		out.write(c);
		}
		System.out.println("File copied to output.txt....");
		}
		finally
		{
		if(in!=null)
		in.close();
		if(out!=null)
			4M
			
			Correct logic 3M
			
			Correct syntax 1M

		out.close();
		}
		}
		}	

	(c)
	
	Ans.
		Explain method overriding with suitable example.
		(Note: Any other example shall be considered)
		Method Overriding in Java:
		If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java. If subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.
		Method overriding is used for runtime polymorphism.
		
		Example:
		class Vehicle{
		void run(){System.out.println("Vehicle is running");}
		}
		class Bike2 extends Vehicle{
		void run()
		{
		System.out.println("Bike is running safely");
		}
		
		public static void main(String args[]){
		Bike2 obj = new Bike2();
		obj.run();
		}
			4M
			
			Explanation 2M
			
			Example 2M

	(d) **Ans.**	**Enlist any four built in packages in java API with atleast two class name from each package.** **Inbuilt packages in java:** 1. **java.lang -** language support classes. These are classes that java compiler itself uses and therefore they are automatically imported. They include classes for primitive types, strings, math functions, threads and exceptions **classes :** Thread, String , 2. **java.util** – language utility classes such as vectors, hash tables, random numbers, date etc **classes :**Date,Collection,Vector	**4M** ***List (any 4) 2M*** ***Any 2 packages class list :1M each***

		3. **java.io** – input/output support classes. They provide facilities for the input and output of data **classes** :FileReader, FileWriter 4. **java.awt** – set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on **classes** :Button,Label 5. **java.net** – classes for networking. They include classes for communicating with local computers as well as with internet servers **classes** :Socket,URL	

	(e)
	
	Ans.
		Write a program to check whether given number is prime or not. ***(Note: Any relevant logic shall be considered)***
		
		import java.io.*;
		class PrimeNo
		{
		public static void main(String args[]) throws IOException
		{
		BufferedReader bin=new BufferedReader(new InputStreamReader(System.in));
		System.out.println("Enter number: ");
		intnum=Integer.parseInt(bin.readLine());
		
		int flag=0;
		for(inti=2;i<num;i++)
		{
		if(num%i==0)
		{
		System.out.println(num + " is not a prime number");
		flag=1;
		break;
		}
		}
		if(flag==0)
		System.out.println(num + " is a prime number");
		}}
			4M
			
			Accept No. from user 1M
			
			Prime No. logic 3M

MODEL ANSWER

WINTER - 2017 EXAMINATION

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

4.	**(A)**	**Attempt any THREE of the following:**	**3x4=12**
	(a)	**Write a program to print the following output:** **1 1 1 1 1** **2 2 2 2** **3 3 3** **4 4** **5** ***(Note: Any relevant logic shall be considered)***	**4M**

	Ans.	class Sample
		{
		public static void main(String args[]){
		inti,j,a=1;
		for(i=5;i>=1;i--)
		{
		for(j=1;j<=i;j++)
		{
		System.out.println(a+" ");
		}
		System.out.println("\n");
		a++;
		}
		}
			Correct logic 3M
			Correct syntax 1M

	(b)	**Illustrate with example the use of switch case statement.** ***(Note: Any other example shall be considered)***	**4M**
	Ans.	**Switch case statement:** 1. The switch statement is used to select among multiple alternatives 2. It uses all primitive datatypes except boolean expression to determine which alternative should be executed. **General form:** switch(expression) { case value1: block 1; break; case value2: block 2; break;	***Explanation 2M***

		.
		.
		.
		default:
		default block;
		break;
		}
		statement n;
		
		Example:
		public class SwitchExample {
		public static void main(String[] args) {
		int number=20;
		switch(number){
		case 10: System.out.println("You are in 10");break;
		case 20: System.out.println("You are in 20");break;
		case 30: System.out.println("You are in 30");break;
		default:System.out.println("Not in 10, 20 or 30");
		}
		}
		}
			Example 2M

	(c)
	
	Ans.
		Write a program to create two thread one to print odd number only and other to print even numbers.
		(Note: Any other logic shall be considered)
		class EvenThread extends Thread
		{
		EvenThread()
		{
		start();
		}
		public void run()
		{
		try
		{
		for(inti = 0;i <= 10;i+=2)
		{
		System.out.println("Even Thread : "+i);
		Thread.sleep(500);
		}
			4M
			
			Correct program 4M

```
}
catch (InterruptedExceptione){}
}
}
class OddThread implements Runnable
{
OddThread()
{
Thread t = new Thread(this);
t.start();
}
public void run()
{
try
{
for(inti = 1;i <= 10;i+=2)
{
System.out.println("Odd Thread : "+i);
Thread.sleep(1500);
}
}
catch (InterruptedExceptione){}
}
}
class Print
{
public static void main(String args[])
{
new EvenThread();
new OddThread();
}
}
```

(d) Ans. **What is the use of try catch and finally statement give example.** **4M**

i. try- 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.

Syntax:

```
try
```

		{
		// block of code to monitor for errors
		}
		For eg. try
		{
		for(inti = 1;i <= 10;i+=2)
		{
		System.out.println("Odd Thread : "+i);
		Thread.sleep(1500);
		}
		}
		
		ii.<u>catch</u>- 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 too can have one or more statements that are necessary to process the exception.
		Syntax:
		catch (*ExceptionType1 exOb*)
		{
		// exception handler for *ExceptionType1*
		}
		For eg.
		catch (InterruptedExceptione){}
		
		iii.<u>finally</u>: It can be used to handle an exception which is not caught by any of the previous catch statements. finally block can be used to handle any statement generated by try block. It may be added immediately after try or after last catch block.
		Syntax:
		finally
		{
		// block of code to be executed before try block ends
		}
		For eg.
		finally{System.out.println("finally block is always executed");}
			Each try 1½M ,catch 1 ½ M, finally 1 M

4.	**(B)**	**Attempt any ONE of the following:**	**1x6=6**
	(a)	**What is importance of super and this keyword in inheritance? Illustrate with suitable example.** ***(Note: Any appropriate example shall be considered)***	**6M**

	Ans.	Using inheritance, you can create a general class that defines traits common to a set of related items. This class can then be inherited by other, more specific classes, each adding those things that are unique to it. In the terminology of Java, a class that is inherited is called a superclass. The class that does the inheriting is called a subclass. Therefore, a subclass is a specialized version of a superclass.
		Whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword **super**.
		
		Superhas two general forms. The first calls the super class constructor. The second is used to access a member of the superclass that has been hidden by a member of a subclass.
		super() is used to call base class constructer in derived class.
		Super is used to call overridden method of base class or overridden data or evoked the overridden data in derived class.
		
		E.g.: use of super()
		class BoxWeightextends Box
		{
		BowWeight(int a ,intb,int c ,int d)
		{
		super(a,b,c) // will call base class constructer Box(int a, int b, int c)
		weight=d // will assign value to derived class member weight.
		}
		
		E.g.: use of super.
		Class Box
		{
		Box()
		{
		}
		void show()
		{
		//definition of show
		}
			Inheritance definition 1M
			
			Super 2M

		} //end of Box class
		
		Class BoxWeight extends Box
		{
		BoxWeight()
		{
		}
		void show() // method is overridden in derived
		{
		Super.show() // will call base class method
		}
		}
		
		The this Keyword
		Sometimes a method will need to refer to the object that invoked it. To allow this, Java defines the this keyword. this can be used inside any method to refer to the current object. That is, this is always a reference to the object on which the method was invoked. You can use this anywhere a reference to an object of the current class' type is permitted. To better understand what this refers to, consider the following version of Box( ): // A redundant use of this. Box(double w, double h, double d) { this.width = w; this.height = h; this.depth = d; }
		
		Instance Variable Hiding
		when a local variable has the same name as an instance variable, the local variable hides the instance variable. This is why width, height, and depth were not used as the names of the parameters to the Box( ) constructor inside the Box class. If they had been, then width would have referred to the formal parameter, hiding the instance variable width.
		// Use this to resolve name-space collisions.
		Box(double width, double height, double depth)
		{
		this.width = width;
		this.height = height;
		this.depth = depth;
		}
			this 1M
			
			Example 2M

	(b)
	
	Ans.
		Write a single program to implement inheritance and polymorphism in java.
		class Employee {
		String name;
		String address;
		int number;
		
		Employee(String name, String address, int number) {
		System.out.println("Constructing an Employee");
		this.name = name;
		this.address = address;
		this.number = number;
		}
		
		public void mailCheck() {
		System.out.println("Mailing a check to " + this.name + " " +
		this.address);
		}
		
		public String toString() {
		return name + " " + address + " " + number;
		}
		
		public String getName() {
		return name;
		}
		
		public String getAddress() {
		return address;
		}
		
		public void setAddress(String newAddress) {
		address = newAddress;
		}
		
		public intgetNumber() {
		return number;
		}
		}
			6M
			
			Correct Logic 4M
			
			Correct Syntax 2M

```
class Salary extends Employee {
private double salary;
Salary(String name, String address, int number, double salary) {
super(name, address, number);
setSalary(salary);
 }

public void mailCheck() {
System.out.println("Within mailCheck of Salary class ");
System.out.println("Mailing check to " + getName()
     + " with salary " + salary);
 }

  public double getSalary() {
  return salary;
 }

public void setSalary(double newSalary) {
if(newSalary>= 0.0) {
 salary = newSalary;
}
 }

public double computePay() {
System.out.println("Computing salary pay for " + getName());
return salary/52;
}
}

public class Demo {
public static void main(String [] args) {
Salary s = new Salary("RAM", "Dadar", 3, 3600.00);
Employee e = new Salary("John ", "Thane", 2, 2400.00);
System.out.println("Call mailCheck using Salary reference --");
s.mailCheck();
System.out.println("\n Call mailCheck using Employee reference--");
e.mailCheck();
}
}
```

MODEL ANSWER

WINTER - 2017 EXAMINATION

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

5.	**(a)**	**Attempt any TWO of the following:** **What is exception? Why the exception occurred in program? Explain with suitable example.**	**2x8=16** **8M**

	Ans.	An exception is an event, which occurs during the execution of a program on an existence of an error, generally a run time error.
		If there are some syntactical errors in the program, those can be caught and debugged by compiler, but if there exist any logical errors, the program may get terminated at run time.
		Exception handling mechanism helps no to terminate the program at runtime because of logical error, but it allows the program to take some proper action and execute further.
		It is achieved by 5 keywords as try, catch, throw, throws and finally.
		1) try :
		The code which is to be monitored is contained in try block
		
		2) catch :
		If there exists any error in try block it is caught in catch block and action is taken. It works like a method and accepts an argument in the form, of Exception object.
		Syntax :
		try
		{
		… // statements to be monitored for errors.
		…
		}
		catch(Exception e)
		{
		… // actions to be taken if exception is caught in try block
		…
		}
		
		3) 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
		
		4) **throws:** It can be used with the method’s declaration which may have some run time errors.
		Example :
		public static void main(String args[]) throws IOException
			Explanation 4M

5) **finally**: it includes the code which executes irrespective of errors in try block. A try should have at least one catch or a finally block associated to complete the syntax.

```
Syntax :
    try
    {
    ...                 // statements to be monitored for errors.
    ...
    }
    finally
    {
    .....               // statements to be executed even if there is no error.
    .....
    }
```

Example (Program to raise an exception if the passwords do not match)

```
import java.io.*;
classPasswordException extends Exception
 {
PasswordException(String msg)
 {
super(msg);
 }
 }

class  PassCheck
 {
public static void main(String args[]) throws IOException
 {
BufferedReader bin=new BufferedReader(new
InputStreamReader(System.in));
try
 {
System.out.println("Enter Password : ");
if(bin.readLine().equals("abc123"))
{
System.out.println("Authenticated ");
 }
else
```

Example 4M

		{
		throw new PasswordException("Authentication failure");
		}
		}
		catch(PasswordException e)
		{
		System.out.println(e);
		}
		}
		}	

	(b)
	
	Ans.
		Write a program to define two thread one to print from 1 to 100 and other to print from 100 to 1. First thread transfer control to second thread after delay of 500 ms.
		class thread1 extends Thread
		{
		public void run()
		{
		int flag=0;
		for(inti=1; i<=10;i++)
		{
		System.out.println("thread1:"+i);
		try
		{
		Thread.sleep(500);
		flag=1;
		}
		catch(InterruptedException e)
		{}
		if (flag==1)
		yield();
		}
		}
		}
		class thread2 extends Thread
		{
		public void run()
		{
		int flag=0;
		for(inti=10; i>=1;i--)
			8M
			
			Correct logic 4M
			
			Correct syntaxes 4M

		{
		System.out.println("thread2:"+i);
		try
		{
		Thread.sleep(500);
		flag=1;
		}
		catch(InterruptedException e)
		{}
		if (flag==1)
		yield();
		}
		}
		}
		class test
		{
		public static void main(String args[])
		{
		thread1 t1= new thread1();
		thread2 t2= new thread2();
		t1.start();
		t2.start();
		}
		}	

	(c) **Ans.**	**How to pass parameter to an applet? Write an applet to accept Account No and balance in form of parameter and print message "low balance" if the balance is less than 500.** **Passing parameters to an applet :** For passing parameters in an applet class <param> tag can be used within <applet> tag. <param> has two attributes as name and value. *For example :* <applet code=applet1 width=200 height=200> <param name="uname" value="abc"> </applet> Attribute name specifies name of the parameter as "uname" in example and value specifies the value inside uname as "abc".	**8M** ***Explanation 4M***

		The values of the parameter can be fetched in applet with the help of getParameter() method as
		String username=getParameter("uname");
		
		Program :
		importjava.awt.*;
		importjava.applet.*;
		public class applet1 extends Applet
		{
		String accno="";
		int balance=0;
		public void init()
		{
		accno=getParameter("acno");
		balance=Integer.parseInt(getParameter("bal"));
		}
		public void paint(Graphics g)
		{
		if(balance<500)
		g.drawString(accno+": Low balance...",100,100);
		else
		g.drawString(accno+":sufficient balance...",100,100);
		}
		}
		/*<applet code=applet1 width=200 height=200>
		<param name="acno" value="1001">
		<param name="bal" value="200">
		</applet>*/
			Program with correct logic and syntax 4M

6.	**(a)** **Ans.**	**Attempt any FOUR of the following:** **What is the use of wrapper classes in Java? Explain float wrapper with its methods.** **Use :** Java provides several primitive data types. These include int (integer values), char (character), double (doubles/decimal values), and byte (single-byte values). Sometimes the primitive data type isn't enough and we may have to work with an integer object. Wrapper class in java provides the mechanism to convert primitive into object and object into primitive.	**4x4=16** **4M** ***Use 2M***

		Float wrapper Class: Float wrapper class is used to wrap primitive data type float value in an object. **Methods :** 1) **floatValue( ) method:** It is used to return value of calling object as float. 2) **isInfinite( ) method:** True if, value of calling object is infinite, otherwise false. 3) **isNaN( ) method:** True if, value of calling object is not a number, otherwise false. 4) **floatToIntBits( ) method:** It is used to return IEEE compatible single precision bit pattern for n. 5) **hashCode(float value) method:** It is used to find hash code of calling object. 6) **intBitsToFloat(int bits) method:** It is used to return float of IEEE compatible single precision bit pattern for n. 7) **parseFloat( ) method:** It is used to return float of a number in a string in radix 10. 8) **toString(float f) method:** It is used to find the string equivalent of a calling object. 9) **valueOf(String s) method:** It is used to return Float object that has value specified by str. 10) **compare(float f1, float f2) method:** It is used to compare values of two numbers. If it returns a negative value, then, n1<n2. If it returns a positive value, then, n1>n2. If it returns 0, then both the numbers are equal. 11) **compareTo (float f1) method:** It is used to check whether two numbers are equal, or less than or greater than each other. If the value returned is less than 0 then, calling number is less than x. If the value returned is greater than 0 then, calling number is greater than x. If value returned is 0, then both numbers are equal. 12) **equals(Object obj) method:** It is used to check whether two objects are equal. It returns true if objects are equal, otherwise false.	***Float Wrapper class with 2 methods 2M***
	(b) **Ans.**	**Write a program to accept number from command line and print square root of the number.**	**4M**

```
class test1
{
public static void main(String args[])
{
intnum;
num= Integer.parseInt(args[0]);
doublesq=Math.sqrt(num);
System.out.println("square root of "+ num +" is +sq);
}}
```

Correct logic 2M

Correct syntax 2M

(c) **Write any four methods of File Input stream class give their syntax.** **4M**

Ans. **File Input Stream class methods:**

Method	Description
int available()	It is used to return the estimated number of bytes that can be read from the input stream.
int read()	It is used to read the byte of data from the input stream.
int read(byte[] b)	It is used to read up to **b.length** bytes of data from the input stream.
int read(byte[] b, int off, intlen)	It is used to read up to **len** bytes of data from the input stream.
long skip(long x)	It is used to skip over and discards x bytes of data from the input stream.
FileChannelgetChannel()	It is used to return the unique FileChannel object associated with the file input stream.
FileDescriptorgetFD()	It is used to return the FileDescriptor object.
protected void finalize()	It is used to ensure that the close method is call when there is no more reference to the file input stream.
void close()	It is used to closes the stream.

Any four methods 1M each

(d) **Write a applet program to set background with red colour and fore ground with blue colour.** **4M**

Ans.

```
import java.awt.*;
import java.applet.*;
public class applet2 extends Applet
{
```

Correct logic 2M

		String str="java programming";
		public void init()
		{
		setBackground(Color.red);
		setForeground(Color.blue);
		}
		public void paint(Graphics g)
		{
		g.drawString("welcome",100,100);
		}
		}
		/*<applet code=applet2 width=200 height=200>
		</applet>*/
			Correct syntax 2M

	(e) Ans.	**Describe access control specifiers with example.**
		There are 4 types of java access modifiers:
		1. private
		2. default
		3. protected
		4. public
		
		1) private access modifier:
		The private access modifier is accessible only within class.
		Example:
		class test
		{
		private int data=40;
		private void show()
		{
		System.out.println("Hello java");
		}
		}
		
		public class test1{
		public static void main(String args[]){
		testobj=new test();
		System.out.println(obj.data);//Compile Time Error
		obj.show();//Compile Time Error
		}
			4M
			
			4 access control specifiers 1M each

		}.
		In this example, we have created two classes test and test1. test class contains private data member and private method. We are accessing these private members from outside the class, so there is compile time error
		
		2) default access specifier:
		If you don't specify any access control specifier, it is default, i.e. it becomes implicit public and it is accessible within the program anywhere.\
		Example :
		class test
		{
		int data=40; //default access
		void show() // default access
		{
		System.out.println("Hello java");
		}
		}
		
		public class test1{
		public static void main(String args[]){
		testobj=new test();
		System.out.println(obj.data);
		obj.show();
		}
		}
		
		3) protected access specifier:
		The protected access specifier is accessible within package and outside the package but through inheritance only.
		Example :
		test.java
		packagemypack;
		public class test
		{
		protected void show()
		{
		System.out.println("Hello java");	

		}
		}
		test1.java
		importmypack.test;
		class test1 extends test
		{
		public static void main(String args[])
		{
		test1obj=new test1();
		obj.show();
		}
		}
		
		4) public access specifier:
		The public access specifier is accessible everywhere. It has the widest scope among all other modifiers.
		Example :
		test.java
		packagemypack;
		public class test
		{
		public void show()
		{
		System.out.println("Hello java");
		}
		}
		test1.java
		importmypack.test;
		class test1 ///inheritance not required
		{
		public static void main(String args[])
		{
		test1obj=new test1();
		obj.show();
		}
		}	