**Java Exam Preparation Model Test**

**Full Marks: 70 Time: 3 Hours**

**Answer all the questions.**

|  |  |
| --- | --- |
| 1. | 1. Consider the following piece of code:

Employee E1 = new Employee("Somebody", 101);Employee E2 = new Employee(E1);What are the values of the expressions E1.equals(E2) and E1 == E2? Why? .5 + .5 + 1 =21. Specify which overloaded method is called by meth(2)? Or is there any error in the following program fragment? State reason behind your answer. 2

class A {  int meth(int a) {.......} double meth(int x) {.......} void meth() {System.out.println(meth(5));} }1. Design a class named **Student** that has two private data – student id and score. The class should contain a parameterized constructor to initialize its data members and one method to display the information. Now write a Java program that will use an array of **Student** objects to represent information about ***n* numbers** of students. Your program should take input from the keyboard and display the information of the students. 6
 |
| 2. | 1. Why should you use the keyword **super** in your Java program? Explain with example. 3
2. Generate the output of the following program: 6

class Add { protected int i; Add(int a) {i = a;} protected void addIt(int amount) {i += amount;} protected int getIt() {return i;} }  class DAdd extends Add { private int i; DAdd(int a, int b) { super(a); i = b; } protected void addIt(int amount) {i = i \* super.i + amount;} protected int getIt() {return i + 1;} protected void doubleIt(int amount) {addIt(2 \* amount);} } public class TestAdder { public static void main(String args[]) { Add A = new Add(3); DAdd DA = new DAdd(1, 5); A.addIt(2); System.out.println(A.getIt()); A = DA; A.addIt(2); System.out.println(A.getIt()); DA.doubleIt(2); System.out.println(A.getIt()); } }1. What is the difference between method overriding and method overloading? 1
 |
| 3. | 1. Write a single Java statement to find the largest value of three integer variables a, b and c. 1
2. Write a program that will take some positive float type numbers from the keyboard and find their summation. If any negative number is input, then your program should handle it with a user-defined exception. 6
3. Determine errors in the following program. Correct them and generate the output. 3

class TestException { public static void main(String args[]) { try { method(); System.out.println("After method()"); } catch(RuntimeException ex) { System.out.println("Exception in main"); } System.out.println("End of main"); }  static void method() throws Exception { try { final int zero=0; int y=2/zero; System.out.println("Recovered from error");} catch(RuntimeException ex) { System.out.println("Runtime Exception in method"); throw ex;  } finally { System.out.println("Finally in method"); } System.out.println("End of method"); } } |
| 4. | 1. Write the line numbers of the following programs which will generate compile errors. 6

 1 package p1; 2  3 public class P1 { 4 public int p1\_pub; 5 private int p1\_pri; 6 protected int p1\_pro; 7 int p1\_def; 8 } 9 10 class P1\_Sub extends P1 { 11 P1\_Sub() {12 super.p1\_pub = 5;13 super.p1\_pro = 5;14 super.p1\_def = 5;15 super.p1\_pri = 5; 16 }17 } 1 package p2; 2 import p1.\*; 3 class P2 { 4 P2() { 5 P1 p1 = new P1(); 6 p1.p1\_pub = 5;  7 p1.p1\_pro = 5; 8 p1.p1\_def = 5; 9 p1.p1\_pri = 5;10 }11 }12 class P2\_SubP1 extends P1 {13 P2\_SubP1() { 14 super.p1\_pub = 5;15 super.p1\_pro = 5;16 super.p1\_def = 5;17 super.p1\_pri = 5;18 }19 }20 class P2\_SubP2 extends P2\_SubP1 {21 P2\_SubP2() {22 p1\_pub = 5;23 p1\_pro = 5;24 p1\_def = 5;25 p1\_pri = 5;26 }27 }**P1.java****P2.java** 1 package p1; 2 3 public class P3 extends P1{ 4 P3() { 5 p1\_pub = 5; 6 p1\_pro = 5; 7 p1\_def = 5; 8 p1\_pri = 5;  9 }10 }**P3.java**1. How can multiple inheritance be achieved in Java? 4
 |
| 5. | 1. Write a Java program that will write a list of **double** numbers into a file. Your program will then read the content of the file and find the **summation** of the numbers. 2.5 + 2.5 + 1 = 6
2. Write a Java code segment that will display the contents of a directory. 2
3. State the advantages of autoboxing & auto-unboxing. 2
 |
| 6. | 1. Write a Java program that will perform the following operations: 5
2. Create an object of type ***ArrayList*** that will contain a list of floating-point numbers.
3. Now insert the following data: 12.34, 34.5, 5.6, 7.89, 10.12, 3.45
4. Show the number of elements in the object.
5. Remove 5.6 and 10.12
6. Display the content of the object.
7. How can you create a thread in your Java program? 4
8. Why should you use abstract class in your program? 1
 |
| 7. | 1. Write a program that will create two threads named ***one*** and ***two*** from the main thread. Each of the thread will display the message “**Thread *name* Starting**”, where ***name*** is the name of the thread. Each thread will then print a message “**Hello from thread *name***” 3 times on the screen. Here, ***nam*e** is the name of the child thread. After each write on the screen it will sleep for 500 milliseconds. Main thread should wait for the termination of the child threads. 7
2. What is the Collections Framework? State the advantages of using collection classes. 3
 |

**Solutions**

1. (c) import java.util.Scanner;

class Student {

 private int student\_ID;

 private int score;

 Student(int std\_ID, int s) {

 student\_ID = std\_ID;

 score = s;

 }

 void display() {

 System.out.println("ID: " + student\_ID + ", score: " + score);

 }

}

public class Main {

 public static void main(String[] args) {

 Scanner in = new Scanner(System.in);

 int n = in.nextInt();

 Student students[] = new Student[n];

 int stdID, stdScore;

 for (int i = 0; i < n; i++) {

 System.out.print("Enter student ID: ");

 stdID = in.nextInt();

 System.out.print("Enter score: ");

 stdScore = in.nextInt();

 students[i] = new Student(stdID, stdScore);

 }

 //Display student information

 for (int i = 0; i < n; i++) {

 students[i].display();

 }

 }

}

2. (b) 5

8

123. (a) int largest = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);

3. (b) import java.util.Scanner;

class NegativeNumberException extends Exception {

 String a;

 NegativeNumberException(String x) {

 a = x;

 }

 public String toString() {

 return "Error! Negative number found: " + a;

 }

}

public class UserDefinedException {

 static float check(float x) throws NegativeNumberException {

 if (x <0) {

 throw new NegativeNumberException(Integer.toString(x));

 } else {

 return x;

 }

 }

 public static void main(String[] args) {

 Scanner in = new Scanner(System.in);

 float sum = 0;

 try {

 while (in.hasNextFloat()) {

 sum += check(in.nextFloat());

 }

 } catch (NegativeNumberException e) {

 System.out.println(e);

 }

 System.out.println(sum);

 }

}

3. (c) **Error: Line 4:** Unreported exception java.lang.Exception, must be caught or declared to be thrown.

**Correction:** Line 2 should be:

public static void main(String args[]) throws Exception {

**Output:**

Runtime Exception in method

Finally in method

Exception in main

End of main

4. (a) P1.java: 15

 P2.java: 7, 8, 9, 16, 17, 24, 25

 P3.java: 8

4. (b) Multiple inheritance can be achieved in Java by allowing a class to inherit from one other class and an unlimited number of interfaces. Below is a program demonstrating multiple inheritance, which inherits a class and an interface:

interface Engine{

 void setHorsePower(int hrsPwr);

 int getHorsePower();

 void setMaker(String mk);

 String getMaker();

}

class Body{

 private String color;

 void setColor(String clr) {

 color = clr;

 }

 String getColor() {

 return color;

 }

}

class Car extends Body implements Engine {

 int hp;

 String maker;

 Car(int hrsPwr, String clr, String mkr) {

 hp = hrsPwr;

 setColor(clr);

 maker = mkr;

 }

 public void setHorsePower(int hrsPwr){

 hp = hrsPwr;

 }

 public int getHorsePower(){

 return hp;

 }

 public void setMaker(String mk){

 maker = mk;

 }

 public String getMaker(){

 return maker;

 }

}

public class Multi\_Inherit {

 public static void main(String[] args) {

 Car toyota\_Corolla = new Car(100, "White", "Toyota");

 Car pajero = new Car(800, "Black", "Mitsubishi");

 }

}

5. (a) import java.io.\*;

public class E19\_1 {

 public static void main(String[] args) {

 DataOutputStream dout = null;

 DataInputStream din = null;

 try {

 dout=new DataOutputStream(new FileOutputStream("C:/a.data"));

 din = new DataInputStream(new FileInputStream("C:/a.data"));

 for (int i = 0; i < 5; i++) {

 dout.writeDouble(i);

 }

 double sum = 0;

 for (int i = 0; i < 5; i++) {

 sum += din.readDouble();

 }

 System.out.println(sum);

 } catch (IOException e) {

 System.out.println("Error: " + e);

 } finally {

 try {

 dout.close();

 din.close();

 } catch (IOException ex) {}

 }

 }

}

5. (b) File dir = new File("c:/windows");

 String[] contents = dir.list();

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

 System.out.println(contents[i]);

 }

5. (c) 1. Removes the tedium of manually boxing and unboxing values.

 2. Helps prevent errors.

 3. Makes working with the Collections Framework much easier.

6. (a) import java.util.ArrayList;

public class E35\_1 {

 public static void main(String[] args) {

 ArrayList<Float> a = new ArrayList<Float>();

 a.add(12.34f);

 a.add(34.5f);

 a.add(5.6f);

 a.add(7.89f);

 a.add(10.12f);

 a.add(3.45f);

 System.out.println(a.size());

 a.remove(5.6f);

 a.remove(10.12f);

 System.out.println(a);

 }

}

6. (b) In a Java program, threads can be created in two ways:

1. By implementing the **Runnable** interface. For example:

class MultiThread implements Runnable {

 Thread t;

 MultiThread() {

 t = new Thread(this);

 t.start();

 }

 public void run() {

 //Some code here

 }

}

1. By extending the **Thread** class. For example:

class MultiThread extends Thread {

 MultiThread() {

 start();

 }

 public void run() {

 //Some code here

 }

}

6. (c) Sometimes there may be a need to create a superclass that only defines a generalized form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details. Such a class determines the nature of the methods that the subclasses must implement. Abstract class provides a way to solve this type of situation.

7. (a) 1 class NewThread implements Runnable {

 2 Thread t;

 3 String threadName;

 4

 5 NewThread(String name) {

 6 threadName = name;

 7 System.out.println("Thread " + threadName + " Starting");

 8 t = new Thread(this, threadName);

 9 t.start();

10 }

11

12 public void run() {

13 try {

14 for (int i = 0; i < 3; i++) {

15 System.out.println("Hello from therad " + threadName);

16 Thread.sleep(500);

17 }

18 } catch (InterruptedException e) {}

19 }

20 }

21

22 public class E11\_1 {

23 public static void main(String[] args) throws InterruptedException{

24 NewThread t1 = new NewThread("one");

25 NewThread t2 = new NewThread("two");

26 t1.t.join();

27 t2.t.join();

28 }

29 }

7. (b) The Collections Framework is a sophisticated hierarchy of interfaces and classes that provide state-of-the-art technology for managing groups of objects.

The advantages of using collection classes are:

1. They are high-performance.
2. They allow different types of collections to work in a similar manner and with a high degree of interoperability.
3. Extending and/or adapting a collection is easy.