**Questions/Exercises for final**

1. What are the differences between composition and inheritance?

1. What is polymorphism. Give a brief example.

Is the ability of one type, A, to appear as and be used like another type, B.

Example:

**public class Base {**

protected int **theInt = 100**;

...

public void **printTheInt**() {

System.out.println( theInt );

}

**public c}lass Doubler extends Base {**

...

public void **printTheInt**() {

System.out.println( theInt\*2 );

}

**}**

**public class Tripler extends Base {**

...

public void **printTheInt**() {

System.out.println( theInt\*3 );

}

**}**

**public class Squarer extends Tripler {**

...

public void **printTheInt**() {

System.out.println( theInt\*theInt );

}

**}**

1. What is the difference between overriding and overloading. Give brief examples.

When a method is **overridden**, the new method definition given in the derived class has the exact same number and types of parameters as in the base class. When a method in a derived class has a different signature from the method in the base class, that is **overloading**.

Example:

private class Parallelogram

{

protected int l1;

protected int l2;

private int getSurface(int height1)

{

return l1\*height1;

}

}

private class Rectangle extends Parallelogram

{

private int getSurface ()

{

return l1\*l2;

}

}

private class Square extends Rectangle

{

private int getSurface ()

{

return l1\*l1;

}

}

1. Given classes A, B, and C, where B extends A, and C extends B, and where all classes implement the instance method void doIt() . How can the doIt() method in A be called from an instance method in C? Why?

**A**:

C obj;

(A) obj.doIt();

Because C is a subclass of A and the method exits in both classes, so it must be specified from which class we want to call it.

1. What would be the result of compiling and running the following program?

// Filename: MyClass. java

public class MyClass

{

  public static void main(String[] args) {

    C c = new C();

    System.out.println(c. max(13, 29));

  }

}

class A {

  int max(int x, int y) { if (x>y) return x; else return y; }

}

class B extends A{

  int max(int x, int y) { return super.max(y, x) - 10; }

}

class C extends B {

  int max(int x, int y) { return super.max(x+10, y+10); }

}

**A**: 29

1. Which is the simplest expression that can be inserted at (1), so that the program prints the value of the text field from the Message class?

// Filename: MyClass.java

class Message {

   // The message that should be printed:

  String text = "Hello, world! ";

}

class MySuperclass {

  Message msg = new Message() ;

}

public class MyClass extends MySuperclass {

  public static void main(String[] args) {

    MyClass object = new MyClass();

    object.print();

  }

  public void print() {

    System.out.println( /\* (1) WRITE THIS COMPLETED STATEMENT \*/ ) ;

  }

}

**A:**

// Filename: MyClass.java

class Message {

   // The message that should be printed:

  String text = "Hello, world! ";

}

class MySuperclass {

  Message msg = new Message() ;

}

public class MyClass extends MySuperclass {

  public static void main(String[] args) {

    MyClass object = new MyClass();

    object.print();

  }

  public void print() {

    System.out.println(msg.text ) ;

  }

}

1. Which method declarations, when inserted at (7), will not result in a compile-time error?

class MySuperclass {

  public        Integer step1(int i)     { return 1; }     // (1)

  protected     String  step2(String str1, String str2) { return str1; }  // (2)

  public        String step2(String str1)    { return str1; }  // (3)

  public static String step2()        { return "Hi "; }  // (4)

  public MyClass      makeIt() { return new MyClass(); }           // (5)

  public MySuperclass makeIt2() { return new MyClass(); }          // (6)

}

public class MyClass extends MySuperclass {

  // (7) WRITE THIS METHOD DECLARATION

}

**A:**

1,2,3,4,5,6

1. What would be the result of compiling and running the following program?

class Vehicle

{

  static public String getModelName() { return "Volvo"; }

  public long getRegNo() { return 12345; }

}

class Car extends Vehicle {

  static public String getModelName() { return "Toyota"; }

  public long getRegNo() { return 54321; }

}

public class TakeARide {

  public static void main(String args[]) {

    Car c = new Car();

    Vehicle v = c;

    System.out.println("| " + v.getModelName() + "| " + c. getModel Name() +

  "| " + v.getRegNo() + "| " + c. getRegNo() + "| ");

  }

}

**A**: | Volvo| Toyota| 54321| 54321|

1. What would be the result of compiling and running the following program?

final class Item {

  Integer size;

  Item(Integer size) { this. size = size; }

  public boolean equals(Item item2) {

    if (this == item2) return true;

    return this.size.equals(item2.size);

  }

}

public class SkepticRide {

  public static void main(String[] args) {

    Item itemA = new Item(10);

    Item itemB = new Item(10);

    Object itemC = itemA;

    System.out.println("| " + itemA.equals(itemB) +

      "| " + itemC.equals(itemB) + "| ") ;

  }

}

 A: | true| false|

1. Which constructors can be inserted at (1) in MySub without causing a compile-time error?

class MySuper {

  int number;

  MySuper(int i) { number = i ; }

}

class MySub extends MySuper {

  int count;

  MySub(int count, int num) {

    super(num);

    this.count = count;

  }

  // (1) WRITE CONSTRUCTOR NEEDED AT THIS POINT

}

1. What will the following program print when run?

// Filename: MyClass.java

public class MyClass {

  public static void main(String[] args) {

    B b = new B("Test");

  }

}

class A {

  A() { this("1", "2"); }

  A(String s, String t) { this(s + t); }

  A(String s) { System.out.printl n(s); }

}

class B extends A {

  B(String s) { System.out.printl n(s); }

  B(String s, String t) { this(t + s + "3"); }

  B() { super("4"); };

}

**12.** Consider the following two class definitions.

class X {

  public double g(double x) {

    return f(x) \* f(x);

  }

  public double f(double x) {

    return x + 1.0;

  }

}

class Y extends X {

  public double f(double x) {

    return x + 2.0;

  }

}

What will the following sequence of statements print?

Y y = new Y();

X x = y;

System.out.println(y.f(2.0));

System.out.println(x.f(2.0));

System.out.println(x.g(2.0));

**13.** Suppose a user executes the following run() method and types the numbers 5 and 22.

public static void run() {

    int a = IO.readInt();

    int b = IO.readInt();

    int m = 0;

    int n = b;

    while(m < n) {

        m += a;

        n -= a;

    }

    System.out.println(m - n);

}

Show all the values taken on by the variables of the program.

a

b

m

n

What does the method print?

1. Complete the below class method so that it reads two strings from the user and displays "yes"' if the second string includes only characters from the first, and "no"' if it contains any characters that the first does not include. For example, I should be able to execute your method and see the following. (Boldface indicates what the user types.)

**brillig glib**

yes

Or I might see the following. In this example, it prints "no" because broil includes the letter o, which does not occur in brillig.

**brillig broil**

no

public static void run() {

}

To accomplish this, you may find the following String instance methods useful.

int length()

Returns the number of characters in the target string.

char charAt(int i)

Returns the character at index i. For example, if str holds the string brillig, str.charAt(2) would return the character 'i'.

int indexOf(String s)

Returns the index where s occurs first within the string on which the method is called. If s occurs nowhere within the target string, the method returns -1. For example, if str holds the string brillig, str.indexOf("il") would return 2, since this is the index where il occurs first.

1. At right, write a definition of a new type, called IntRange, representing a range of integers. This class should support the following instance methods.

IntRange(int start, int end)

(Constructor method) Sets up a new IntRange object representing the set of integers between start and end, including these two endpoints. The method may assume that the first parameter is below or equal to the second parameter.

int getSize()

Returns the number of integers in the range.

boolean contains(int i)

Returns true if i lies within the range.

The following example method, which uses the IntRange class you define, illustrates how the class should work.

public static void run() {

    IntRange range = new IntRange(2, 4);

    System.out.println(range.getSize());         // this prints ``3''

    System.out.println(range.contains(1));       // this prints ``false''

    System.out.println(range.contains(3));       // this prints ``true''

    System.out.println(range.contains(4));       // this prints ``true''

    System.out.println(range.contains(5));       // this prints ``false''

}

public class IntRange {

}

1. Suppose we have the following two class definitions.

|  |  |
| --- | --- |
| class A {    public int f(int x) {      return 2 \* x;    }    public int g(int x) {      return f(x \* 3);    }  } | class B extends A {    public int f(int x) {      return 5 \* x;    }    public int h(int x) {      return f(7 \* x);    }  } |

And suppose we execute the following run method.

public static void run() {

  B b = new B();

  System.out.println(b.f(1) + " " + b.g(1) + " " + b.h(1));

  A a = b;

  System.out.println(a.f(1) + " " + a.g(1));

}

What would the method print?

1. Define a class PassCount to track whether all the students of a class have passed a test. It should support the following methods.

PassCount()

(Constructor method) Constructs an object representing an empty class.

void addGrade(double grade) Adds grade into the class.

boolean isAnyFailing() Returns true only if there is somebody in the class who received a grade of less than 60.

For example, if you defined this class properly, I should be able to write the following class to test it.

public class PassCountTest {

public static void run() {

PassCount a = new PassCount();

addGrade(45.0);

addGrade(76.0);

System.out.println(a.isAnyFailing()); // should print "true"

PassCount b = new PassCount();

addGrade(60.0);

System.out.println(b.isAnyFailing()); // should print "false"

}

}

Note that, to accomplish this, a PassCount object need not remember every single grade --- that is, you do not need to use arrays: It just needs to remember whether any of the ones it has seen were below 60.

1. Define a class NumberIterator for iterating through a sequence of numbers. It should support the following methods.

NumberIterator(int start, int stop)

(Constructor method) Constructs an object for iterating through the integers beginning at start and going up to stop. The constructor assumes that start is less than stop.

boolean hasMoreNumbers()

Returns true if there are more numbers remaining in the sequence.

int nextNumber()

Returns the current number in the sequence and steps the iterator forward, so that the next call to this method returns the following number in the sequence. This method initiates a NoSuchElementException if the sequence has no more elements remaining.

For example, if you defined this class properly, I should be able to write the following class to test it. When executed, its run() method would print "5 6 7 8".

public class NumberIteratorTest {

    public static void run() {

        NumberIterator it = new NumberIterator(5, 8);

        System.out.print(it.nextNumber());

        while(it.hasMoreNumbers()) {

            int i = it.nextNumber();

            System.out.print(" " + i);

        }

    }

}

1. Suppose we have the following two class definitions.

|  |  |
| --- | --- |
| class P {    public int f(int x) {      return x + 1;    }    public int g(int x) {      return f(x + 2);    }  } | class Q extends P {    public int f(int x) {      return x + 4;    }    public int h(int x) {      return f(x + 8);    }  } |

And suppose we execute the following run method.

public static void run() {

  P a = new P();

  Q b = new Q();

  P c = b;

  System.out.println(a.f(0) + " " + a.g(0));

  System.out.println(b.f(0) + " " + b.g(0) + " " + b.h(0));

  System.out.println(c.f(0) + " " + c.g(0));

}

What would the method print?

1. Suppose we have the class defined as below.

|  |  |
| --- | --- |
| class C {      static int y = 0;      int z;        C() {          z = 0;      }      void incrX() {          int x = 0;          x++;          IO.println(x);      }      void incrY() {          y++;          IO.println(y);      }      void incrZ() {          z++;          IO.println(z);      }  } | public class Example {      public static void run() {          C a = new C();          C b = new C();          a.incrX();          a.incrX();          b.incrX();          a.incrY();          a.incrY();          b.incrY();          a.incrZ();          a.incrZ();          b.incrZ();      }  } |

What would the computer print when it executes the Example class's run() method?

1. A matrix is symmetric if, for each i and j, ai,j = aj,i. The following is an example of a symmetric matrix.

|  |  |  |
| --- | --- | --- |
| 0 | 23 | 45 |
| 23 | 10 | 36 |
| 46 | 36 | 20 |

Another way of defining it: A symmetric matrix can be reflected across its main diagonal (top left corner to bottom right corner) to obtain the same matrix.

Complete the following method so that it returns true only if the two-dimensional array parameter mat is symmetric. Your solution may assume that the matrix is square (as many columns as rows).

public static boolean isSymmetric(int[][] mat) {

}

(Your solution shouldn't really be this long!)

1. Define a class PigPen for tracking the number of pigs in in a pen. It should support the following methods.

PigPen(int pigs)

(Constructor method) Constructs an object representing a pig pen containing pigs pigs.

boolean isEmpty()

Returns true if there are no pigs in the pen.

void pigEnters()

Adds one to the number of pigs in the pen.

void pigExits()

Subtracts one from the number of pigs in the pen.

For example, if you defined this class properly, I should be able to write the following class to test it.

public class PigPenTest {

    public static void run() {

        PigPen pen = new PigPen(2);

        pen.pigExits();

        System.out.println(pen.isEmpty()); // prints "false"

        pen.pigExits();

        System.out.println(pen.isEmpty()); // prints "true"

        pen.pigEnters();

        System.out.println(pen.isEmpty()); // prints "false"

    }

}

1. Write a class method named mode that takes an array of ints as a parameter and returns the integer that occurs in the array most frequently. For example, the following code fragment that uses your mode method should print 23.

int[] a = { 23, 34, 45, 23, 0, 23 };

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

Your method should not call any other methods to accomplish this. It will need more than one loop to count the number of occurrences of each number in the array.

1. Consider the following Java program.

class A {

    int f() { return 1; }

}

class B extends A {

    int f() { return 0; }

}

class Main {

    public static void main(String[] args) {

        A a = new B();

        System.out.println(a.f());

    }

}

What does this program print?

1. Consider the following program.

class Ident {

    int f(int x) { return x; }

    int g(int x) { return f(f(x)); }

}

class Square extends Ident {

    int f(int x) { return x \* x; }

}

class Main {

    public static void main(String[] args) {

        Ident a = new Ident();

        Ident b = new Square();

        Square c = new Square();

        System.out.println(a.g(3) + " " + b.g(3) + " " + c.g(3));

    }

}

What does this program print?

1. Write an expression will extract the substring "kap", given the following declaration:

String str = "kakapo";

1. What will be the result of attempting to compile and run the following code?

class MyClass {

  public static void main(String[] args) {

    String str1 = "str1";

    String str2 = "str2";

    String str3 = "str3";

    str1.concat(str2) ;

    System.out.println(str3.concat(str1));

  }

}

1. What will be the result of attempting to compile and run the following program?

public class RefEq {

  public static void mai n(String[] args) {

    String s = "ab" + "12";

    String t = "ab" + 12;

    String u = new String("ab12");

    System.out.println((s==t) + " " + (s==u));

  }

}

1. What is a Java exception?

1. How can done deal with exceptional conditions in Java. Give a short example.

1. Describe the thry-throw-catch mechanism.

1. State the catch-or-declare rule.

1. How do exceptions propagate? Give a brief example.

1. What is the minimum a user-defined exception class must contain? Give a short example.

1. What is an *inner* class?

1. What is an *anonymous inner* class? Give a short example.

1. What is a local class? Give a short example.

1. What is a Java *component*? Give a short example.

1. What is a Java *container*? Give a short example.

1. What are the Basic steps in displaying Java Graphics?

1. What is a listener in Java?

1. What is the purpose of call-backs?

1. What is the role of the *model* in the MVC architecture?

1. What is the role of the *controller* in the MVC architecture?

1. What is the role of the *view* in the MVC architecture?

1. Briefly describe the Swing separable model architecture.

1. What is a Java Enumeration? Give a small example.

1. What is a *generic type* in Java? Give a small example.

1. Write a generic method which prints the values stored in a collection.

1. What is an *iterator* in Java? Give a small example.

1. How can one include primitive values in a Java Collection? Give a short example.

1. What is a Java Collection?

1. What are the main differences between classes Vector and ArrayList?

1. What is the purpose of software testing?

1. What is functional testing?

1. What defines a test case?

1. What must be considered when developing a test plan?

1. What is a unit test?

1. What is a test harness?

1. What is regression testing?

1. What is test coverage?

1. How can one get a program trace? (Hint: there are at least 2 ways)

1. What are the benefits of logging?

1. What are the shortcomings of logging?

1. What will the following program print when run?

public class UppTurn {

  public static void mai n(String[] args) {

    String str1 = "l ower", str2 = "LOWER", str3 = "UPPER";

    str1.toUpperCase();

    str1.replace("LOWER", "UPPER");

    System.out.printl n((str1. equals(str2)) + " " + (str1. equals(str3))) ;

  }

}

1. Describe the steps needed to read strings from a formatted sequential file.

1. What will the method length() in the class File return?

1. If write(0x01234567) is called on an instance of OutputStream, what will be written to the destination of the stream?

1. Given the following program:

import java.io.DataInputStream;

import java.io.EOFException;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class Endings {

  public static void main(String[] args) {

    try {

      FileInputStream fos = new FileInputStream("info.dat") ;

      DataInputStream dis = new DataInputStream(fos);

      int i= dis.readByte();

      while (i != -1) {

        System.out.print((byte)i + "| ");

        i= dis.readByte();

      }

    } catch (FileNotFoundException fnf) {

      System.out.println("File not found");

    } catch (EOFException eofe) {

      System.out.println("End of stream");

    } catch (IOException ioe) {

      System.out.println("Input error");

    }

  }

}

What will happen when one attempts to compile and run it?

1. How many methods are defined in the Serializable interface?

1. Given the following code:

public class Person  {

  protected String name;

  Person() { }

  Person(String name) { this. name = name; }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import java.io. Serializable;

public class Student extends Person implements Serializable {

  private long studNum;

  Student(String name, long studNum) {

    super(name);

    this. studNum = studNum;

  }

  public String toString() { return "(" + name + ", " + studNum + ") "; }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import java.io.\*;

public class RQ800\_10 {

public static void main(String args[])

    throws IOException, ClassNotFoundException {

    FileOutputStream outputFile = new FileOutputStream("storage. dat");

    ObjectOutputStream outputStream = new ObjectOutputStream(outputFile);

    Student stud1 = new Student("Aesop", 100);

    System.out.print(stud1);

    outputStream. writeObject(stud1);

    outputStream. flush();

    outputStream. close();

    FileInputStream inputFile = new FileInputStream("storage. dat");

    ObjectInputStream inputStream = new ObjectInputStream(inputFile);

    Student stud2 = (Student) inputStream.readObject();

    System.out.println(stud2);

    inputStream. close();

  }

}

What will happen when one attempts to compile and run it?

1. What will happen when one attempts to compile and run the following code:

import java.util. ArrayList;

import java.util. Collections;

import java.util. List;

public class WhatIsThis {

  public static void main(String[] args) {

    List<StringBuilder> list = new ArrayList<StringBuilder>();

    list.add("B");

    list.add("A");

    list.add("C");

    Collections.sort(list, Collections.reverseOrder());

    System.out.println(list.subList(1, 2));

  }

}

1. How can one achieve object persistence in Java?

1. What is a Java Buffer?

1. What are the benefits of Buffer *views*?

1. What is the purpose of a direct Buffer? Give a brief example.

1. What is a Java Channel?

1. When does the execution of a thread end?

1. How can the priority of a thread be set?

1. Describe the life cycle of a thread.

1. How should one correctly terminate a Java thread?

1. What is a Java Collection?

1. What are the main restrictions which apply to applets?

1. What are the differences between a standalone application and an applet?

1. Describe the life cycle of an applet.

1. What should be written in each of an applet's predefined methods?

1. What are the top level containers for GUIs in Java?

1. What are Java threads?

1. What are the differences between threads and operating system tasks?

1. What is a factory method? Give a brief example.