



**1. True/False [ 20 pts ]**

Answer the following questions True or False. Be sure to write out True and False, NOT just a T or F.

- (a) The `BoxLayout` manager positions elements in a rectangular grid or box in which you specify the number of rows and columns in the grid.
- (b) Overloading is the process when a subclass provides its own version of a method which also exists in its parent class.
- (c) A `NullPointerException` is an unchecked exception.
- (d) The *finally* clause of a try-catch block is only executed if none of the catch blocks execute.
- (e) The *transient* reserved word is used in java to designate fields that should not be written to disk during object serialization.
- (f) Using polymorphism with references can be achieved in Java by using either a parent class or an interface name as the declaration type.
- (g) An abstract data type (ADT) exposes the details of implementation so that code using it can be optimized for speed.
- (h) A mouse move event and a mouse click event will be sent to different listener methods.
- (i) A linked list uses object references to establish links between nodes in the list.
- (j) A bag would be a good data structure for modeling people standing in line at a bank, where the order of customer arrival was important to know.

## 2. Multiple Choice, Basic Inheritance [15 pts]

Answer the following questions based upon this partial class information:

```
public class A1 {
    public int x;
    private int y;
    protected int z;
    ...
}

public class A2 extends A1 {
    protected int a;
    private int b;
    ...
}

public class A3 extends A2 {
    private int q;
    public A3() { super(); }
    ...
}
```

- 3 (a) Which of the following is true with respect to A1, A2 and A3?
1. A1 is a subclass of A2 and A2 is a subclass of A3
  2. A3 is a subclass of A2 and A2 is a subclass of A1
  3. A1 and A2 are both subclasses of A3
  4. A2 and A3 are both subclasses of A1
  5. A1, A2 and A3 are all subclasses of the class A
- 3 (b) Which of the following lists of instance data are accessible in class A2?
1. x, y, z, a, b
  2. x, y, z, a
  3. x, z, a, b
  4. z, a, b
  5. a, b
- 3 (c) Which of the following lists of instance data are accessible in A3?
1. x, y, z, a, b, q
  2. a, b, q
  3. a, q
  4. x, z, a, q
  5. x, a, q

6. x, q

- 3 (d) The instruction `super( )`; does which of the following?
1. calls the method `super` as defined in the current class
  2. calls the method `super` as defined in the current class' parent class
  3. calls the method `super` as defined in `java.lang`
  4. calls the constructor as defined in the current class
  5. calls the constructor as defined in the current class' parent class
- 3 (e) The relationship between a parent class and a child class is referred to as a(n) ---- relationship.
1. has-a
  2. is-a
  3. was-a
  4. instance-of
  5. alias

### 3. Constructor Tracing[ 10 pts ]

What is printed by the following code.

```
public class Car {
    private int year;
    private String maker;

    public Car(String m, int y) {
        year=y;
        maker=m;
        System.out.println("Car");
    }
    public Car() { this("Ford",1997); }

    public String toString() { return "IMA Car";}
} //end car

public class RaceCar extends Car {
    private int horsepower;

    public RaceCar(String m, int y, int h) {
        super(m,y);
        horsepower=h;
        System.out.println("RaceCar");
    }

    public RaceCar() { this("Chevy",1957,350);}

    public String toString() {
        return "Vroom, Vroom: "+horsepower;
    }
    public static void main(String[] args) {
        RaceCar rc = new RaceCar();
        System.out.println(rc);
        Car c = new Car();
        System.out.println(c);
    }
}
```

---

Output:

#### 4. Polymorphism [ 12 pts ]

Consider the following code:

```
public abstract class RecordingDevice {
    private int runTime;
    public abstract void record();
    public abstract void play();
}

public class CD extends RecordingDevice {
    public void record() { }
    public void play() { }
}

public interface Portable {
    public void carry();
}

public class MP3 extends RecordingDevice implements Portable {
    public void record() { }
    public void carry(){ }
    public void play(){ }
}

public class MP3Player extends MP3 {
    public void play(){ }
    public void zap(){ }
}
```

For each of the following lines of code, decide whether they will compile and run without error. If the line(s) are OK, then simply write OK. If they will fail to compile, write COMPILE ERROR or if they will fail at runtime write RUNTIME ERROR. For those items that are not OK, give a BRIEF explanation of what is wrong.

Note: for this question, you will be penalized for guessing. On each part

- For a right answer: you will receive the points for that part
- For a wrong answer: you will lose the points for that part
- For no answer (left blank): you will neither receive nor lose points for that part.

Your score for the question will not, however, be below zero.

- 2 (a) MP3 mp3 = new MP3Player();

- 2 (b) Portable p = new RecordingDevice();
- 2 (c) MP3Player mp = new MP3();
- 2 (d) RecordingDevice[] rda= {new MP3Player(), new MP3(), new CD()};  
rda[0].record();
- 2 (e) MP3 mp1 = new MP3Player();  
mp1.zap();
- 2 (f) RecordingDevice rd = new CD();  
((MP3Player)rd).zap();

## 5. Dynamic Binding [ 8 pts ]

Assume you are given the following classes:

- Plane is abstract, and has the abstract method `public int fuelRate()`, an method which returns the pounds of fuel per hour that a plane consumes.
- Commuter extends Plane and implements `fuelRate` so that it returns 20.
- Fighter extends Plane and implements `fuelRate` so that it returns 200.
- Airliner extends Plane and implements `fuelRate` so that it returns 100.

Write the method `public int totalFuelNeeded(Plane[] airport)` which computes the total pounds of fuel needed for all the planes at an airport. This method should be useable with other subclasses of Plane that might be created later. For this method you may not use:

1. The `instanceof` operator
2. Casting of any sort
3. The `getClass()` method

If you violate the above restrictions, you will receive no credit for this question. For example, if an airport were created with

```
Plane[] airport = {new Commuter(), new Fighter(), new Airliner()};
```

Then the call: `totalFuelNeeded(airport)` would return 320.

```
public int totalFuelNeeded(Plane[] airport) {
```

```
}
```

**6. Exceptions [ 10 pts ]**

Given the following code fragment (with line numbers on the left that do not affect the code's functionality):

```
1: int x = getId();
   try {
2:   x= validate(x);
3:   checkSum();
4:   y=10;
   } catch (IOException ioe) {
5:   System.err.println(ioe);
   } catch (NoSuchElementException nse) {
6:   System.err.println(nse);
   } catch (Exception e) {
7:   System.err.println(e);
   } finally {
8:   System.out.println("Out of try");
   }
9: System.out.println("end of method");
```

Write in the correct order the line numbers of that execute in the code above if:

- (a) There are no exceptions thrown by any statement.
- (b) There is an unchecked Exception thrown in the call to getId in statement 1.
- (c) There is a NoSuchElementException thrown in the call to checkSum at statement 3.

## 7. File IO Coding [15 pts ]

Implement a method called *type* which reads in a series of text lines from a file, and echoes them to System.out. *type* does not throw any exceptions [note it is implemented without requiring a throws clause in method header](HINT: This requires you to catch the exceptions in the method). The following excerpts from the API may be useful:

```
Class FileReader
```

```
FileReader(String name)throws FileNotFoundException; //Creates a new FileReader.
```

```
int read() throws IOException; //Read a single character.
```

```
void close() throws IOException; //Close the stream.
```

```
Class BufferedReader
```

```
BufferedReader(Reader in); //Create a buffering character-input stream.
```

```
String readLine() throws IOException; //Read a line of text
```

```
void close() throws IOException; //Close the stream.
```

```
public static void type(String filename) {
```

```
}//end type
```

**8. Bag Coding [ 10 pts ]**

Implement the method `addAll()` which takes a `Bag` as a parameter and adds its contents to the current bag. Assume that this method is part of a `BagImpl` class which uses an `ArrayList` as its underlying storage medium. The following API excerpt may be helpful:

```
ArrayList
```

```
boolean add(Object o); //Appends object o to end of list.
void clear(); //Removes all elements from the list.
Object get(int index); //Returns the element at the specified index.
Iterator iterator(); //Returns an iterator for the ArrayList;
int size(); //Returns the number of elements in the list.
```

```
Iterator
```

```
boolean hasNext(); //Returns true while there are more elements
Object next(); //Returns the next object in the collection.
```

```
Bag
```

```
Iterator iterator(); //Returns an iterator over the Bag
Object remove(); //Removes a random object from the Bag
void add(Object o); //Adds the object to the Bag
```

```
public class BagImpl implements Bag {
    ArrayList data;
```

```
    public void addAll(Bag aBag) {
```

```
        } //end add method
    } //end class def
```