QUEEN'S UNIVERSITY
SCHOOL OF COMPUTING

CISC212, FALL TERM, 2004
FINAL EXAMINATION
9am to 12noon, 22 DECEMBER 2004

Instructors: Alan McLeod and Nidal Nasser

If the instructor is unavailable in the examination room and if doubt exists as to the interpretation of any problem, the candidate is urged to submit with the answer paper a clear statement of any assumptions made.

Proctors are unable to respond to queries about the interpretation of exam questions. Do your best to answer exam questions as written.

Please write your answers in the boxes provided. Extra space is available on page 16 of the exam. The back of any page can be used for rough work. This exam refers exclusively to the use of the Java language. Comments are not required in the code you write.

This is a closed book exam. No computers or calculators are allowed.

Student Number:

Problem 1: / 12
Problem 2: / 10
Problem 3: / 16
Problem 4: / 20
Problem 5: / 16
Problem 6: / 6

TOTAL: / 80
Problem 1) [12 marks]
Write the output of the following program in the box beside (or under) each System.out.println() statement. The program compiles and runs without any errors.

```java
public class WhatIsOutput {

    public static void main (String[] args) {

        //Integer.MAX_VALUE = 2147483647
        //Integer.MIN_VALUE = -2147483648
        //Double.MAX_VALUE = 1.7976931348623157E308
        //Double.MIN_VALUE = 4.9E-324

        System.out.println((Integer.MAX_VALUE + 1) == Integer.MIN_VALUE); //

        System.out.println((Integer.MAX_VALUE + 1) == Integer.MAX_VALUE);

        System.out.println((Double.MAX_VALUE + 1) == Double.MAX_VALUE);

        System.out.println(Double.MIN_VALUE / 2);

        System.out.println(1.0 / 0);

        System.out.println(Math.sqrt(-1.0));

        System.out.println((1E10 + 1E-5) == 1E10);

        System.out.println((1E10 + 1E-8) == 1E10);

        System.out.println((1E-10 + 1E-30 - 1E-10) == 1E-30);

        System.out.println((1E-10 - 1E-10 + 1E-30) == 1E-30);
    }
}
```
Problem 1 Cont.

double sum = 0;
for (int i = 1; i <= 100; i++) sum += 0.1;
System.out.println(sum == 10.0);

System.out.println(Math.abs(sum - 10) < 1e-6);

} // end main

} // end WhatIsOutput

Problem 2) [10 marks]
For this problem you must examine the following four classes, ClassA, ClassB, ClassC and TestClass. You must provide the output of the main method in TestClass. These classes compile and run without any errors.

public abstract class ClassA {

    private int num1;
    private int num2;

    public ClassA (int n1, int n2) {
        num1 = n1;
        num2 = n2;
    }

    public abstract int sumNums();

    public abstract int subNums();

    public abstract int subNums(char c);

    public int timesNums() {
        return num1 * num2;
    }

    public int getNum1 () { return num1; }

    public int getNum2 () { return num2; }

} // end ClassA
Problem 2 Cont.)

public class ClassB extends ClassA {

    private int num1;

    public ClassB (int n1, int n2, int n3) {
        super(n1, n2);
        num1 = n3;
    }

    public ClassB (int n) {
        super(n, n + 1);
        num1 = n;
    }

    public int sumNums () {
        return num1 + getNum1() + getNum2();
    }

    public int timesNums() {
        return super.timesNums() * num1;
    }

    public int subNums () {
        return num1 - super.getNum2() - super.getNum1();
    }

    public int subNums (char dir) {
        if (dir == 'f')
            return super.getNum1() - super.getNum2() - num1;
        else
            return subNums();
    }

    public int getNum1 () { return num1; }
}

// end ClassB

public class ClassC extends ClassB {

    private int num1;

    public ClassC (int n1, int n2, int n3, int n4) {
        super(n1, n2, n3);
        num1 = n4;
    }
}
Problem 2 Cont.)

```java
public ClassC (int n) {
    super(n);
    num1 = n;
}

    public int timesNums() {
        return super.timesNums() * num1;
    }

} // end ClassC

public class TestClass {

    public static void main (String[] args) {

        ClassA b = new ClassB(2, 3, 4);
        System.out.println(b.getNum1());
        System.out.println(b.sumNums());
        System.out.println(b.timesNums());
        System.out.println(b.subNums('f'));
        System.out.println(b.subNums());
        ClassA c = new ClassC(3);
        System.out.println(c.timesNums());
        System.out.println(c.sumNums());
        System.out.println(c.getNum1());
        System.out.println(c.getNum2());
        ClassA cc = new ClassC(5, 6, 7, 8);
        System.out.println(cc.sumNums());

    } // end main

} // end TestClass
```
Problem 3) [16 marks]
The SolidSphere class must extend the Sphere class and implement the Shape interface, given below:

```
public interface Shape {

    double PI = Math.PI;

    // The circumference is pi * diameter
double circumference ();

    // The surface area of a sphere is 4*pi*r*r
double area ();

    // The volume of a sphere is 4*pi*r*r*r/3
double volume ();

    // Returns true if a rectangle of size length by width fits within
    // the sphere
    boolean fits (double length, double width);

} // end Shape interface
```

```
public class Sphere {

    private double r;

    public Sphere (double radius) {
        setRadius(radius);
    }

    public double getRadius () { return r; }

    private void setRadius (double radius) { r = radius; }

} // end Sphere
```

These two classes compile without problems and cannot be changed. The following page contains the code for a SolidSphere class that does not compile. Below the incorrect code, list all the compilation errors that you can find. On the next page, write your own version of SolidSphere that will compile.
problem 3 cont.)
// Does not compile!
public class SolidSphere extends Sphere implements Shape {

    private String m;

    public SolidSphere (double radius, String material) {
        m = material;
        PI = Math.PI;
    }

    public SolidSphere (double radius) {
        m = "wood";
        PI = Math.PI;
    }

    private double volume () {
        return 4.0 * PI * Math.pow(r, 3) / 3.0;
    }

    private boolean fits (double l, double w, double h) {
        double diagonal = Math.sqrt(Math.pow(l, 2) + Math.pow(w, 2));
        return diagonal <= 2 * r;
    }

} // end SolidSphere

list errors here:
Problem 3 Cont.)
Write a correct version of SolidSphere here. You can continue onto the next page, if necessary.

```java
public class SolidSphere extends Sphere implements Shape {
```
Problem 4) [20 marks]
The Java API for the `Arrays.sort(Object[] a)` method in the `java.util` package states:

“All elements in the array must implement the Comparable interface. Furthermore, all elements in the array must be mutually comparable (that is, `e1.compareTo(e2)` must not throw a `ClassCastException` for any elements `e1` and `e2` in the array).”

Here is the code for the Comparable interface:

```java
package java.lang;

public interface Comparable {
    public int compareTo(Object o);
}
```

You must write the code for the `Point` class on the next page. You do not need to write code for any other classes. An array of `Point` objects must be “sortable” by the `Arrays.sort()` method. You must also override the `equals()` method of the `Object` class, so that search techniques can be used. Here are some more design details:

- The `Point` class holds two private double attributes, `x` and `y`, that represent the Cartesian coordinates of a point on a plane.
- You do not need to carry out any error checking on these two values – any values are acceptable.
- Equality is defined as the `x` and `y` values agreeing with the values of another `Point` object to within 1 part per million (`1x10^-6`).
- The signature for the `equals` method is “public boolean equals (Object o)”. 
- `Point` objects will be ordered by the distance of a `Point` from the origin, where this distance is defined as `Math.sqrt(x*x + y*y)`. Points closer to the origin must come before Points that are further away, when an array of Points is sorted. So, `compareTo` will return –1 if the current `Point` is closer to the origin than the `Point` supplied to the method.
- If the `Object` supplied to the `equals()` method is not a `Point`, return `false`.
- If the `Object` supplied to the `compareTo` method is not a `Point`, return –999.
- Write a single, full parameter constructor only.
- Do not write any mutator methods.
- None of the other standard ADT methods (such as `toString()`) are required.
- Do not write any other classes or a main method.
Problem 4 Cont.)
Write your Point class here and continue onto the next page, if necessary: