CSCI 201 Fall'07: Review for the Final

Notes: (1) All exams are closed book.

(2) You should also study the material on previous reviews and exams.

(3) The exam will contain questions on strings and inheritance similar to those listed below.

Problem 1: Strings

Below are the method summaries of several Java String methods taken from the Java API documentation.

char	**charAt**(int index) Returns the char value at the specified index.
int	**indexOf**(int ch) Returns the index within this string of the first occurrence of the specified character.
int	**lastIndexOf**(int ch) Returns the index within this string of the last occurrence of the specified character.
int	**length**() Returns the length of this string.
String	**substring**(int beginIndex) Returns a new string that is a substring of this string.
String	**substring**(int beginIndex, int endIndex) Returns a new string that is a substring of this string.

Keep in mind that the first substring method returns a substring that begins at position beginIndex and continues to the end of the string while the second substring method returns a substring that begins at position beginIndex and ends at position endIndex-1. For example, “abcde”.substring(2) will be “cde and “abcde”.substring(2,4) will be “cd”. Also, remember that two String’s can be concatenated by the + operator, for example, “ab” + “cde” would be “abcde”.

Suppose S is a String object and that it contains a person’s first and last name separated by a space, as in “George Washington”. Using the above String methods plus concatenation write small sections of Java to do the following operations:

(1) Set the integer variable b to the position of the first blank within the line.

```
int b = S.indexOf(’ ’) ;
```

(2) Set the String variable lastName to contain the person’s last name.

```
String lastName = S.substring(b+1) ;
```

(3) Set the String variable firstName to contain the person’s first name.

```
String firstName = S.substring(0, b) ;
```

(4) Set the String variable reversedName so that the name is of the form “Lastname, Firstname”, as in “Washington, George”.

```
String reversedName = lastName + ”, ” + firstName ;
```

Problem 2: the toString() method

What will be output by the program below?

```
public class Bird
{     protected String name, size, colors;
      public Bird(String n, String s, String myColors)
      {     name = n;
            size = s;
            colors = myColors;
      }
      public String song()
      {     String mySong = "tweet, tweet, tweet";
            return mySong;
      }
      public String eats()
      {     String food = "worms and seeds";
            return food;
      }
      public String nameAndColors()
      {     String features = "a " + name + "'s color is " + colors;
            return features;
      }
```

```
	public String nameAndSize()
	{	String features = "a " + name + "'s overall size is " + size;
		return features;
	}
}

public class BlueBird extends Bird
{	public BlueBird()
	{	super ("Bluebird", "small", "blue");
	}
	public String toString()
	{	String myMessage = nameAndSize() + "\n";
		myMessage = myMessage + "a " + name + " song is " + song() +
"\n";
		return myMessage;
	}
}

public class BirdTalk
{	public static void main(String [] args)
	{	BlueBird happy = new BlueBird();
		System.out.print(happy);
	}
}
```

```
a Bluebird's overall size is small
a Bluebird song is tweet, tweet, tweet
```

Problem 3: Overriding and Overloading

a. Write a class called *Peguin* that is derived from the *Bird* class given in the problem above. The Penguin class must contain the methods described below. (It may include other methods as well.)

1. A constructor. The constructor must create a Penguin object with the following attributes:

   name = “Penguin”, size = “large”, colors = “black and white”

2. A *song()* method that **overrides** the Bird class *song*() method and returns the String: "bark, bark, bark"

3. A *eats()* method that **overloads** the Bird class *eats*() method. The new method *eats*() has one String parameter; it returns the value of its String parameter.

```
public class Peguin extends Bird {

	public Penguin() {
		super(”Penguin”, ”large”, ”black and white”) ;
	}

	public String song() {
		return “bark, bark, bark” ;
```

```
      }

      public String eats(String s) {
            Return s ;
      }
}
```

b. What is output by the program below?

```
public class MusicNote {

    double frequency ;

    public MusicNote() {
        frequency = 440.0 ;
    }
    public MusicNote(double f) {
        frequency = f ;
    }
    public void setFrequency(double f) {
        frequency = f ;
    }
    public double getFrequency() {
        return frequency ;
    }
    public void play() {
        System.out.printf("%8.2f ---------------> %8.2f\n",
                frequency, frequency) ;
    }
}

public class GuitarNote extends MusicNote {

    double bend ;

    public GuitarNote() {
        bend = 0.0 ;
    }
    public GuitarNote(double f) {
        super(f) ;
    }
    public GuitarNote(double f, double b) {
        super(f) ;
        bend = b ;
    }
    public void setBend(double b) {
        bend = b ;
    }
    public double getBend() {
        return bend ;
    }
    public double getFrequency(double error) {
        return (frequency * (1+error));
            }
    public void play() {
        System.out.printf("%8.2f --> %8.2f --> %8.2f\n",
```

```
                frequency, frequency*(1+bend), frequency) ;
    }
}


public class Main {

    public static void main(String[] args) {
        MusicNote m = new MusicNote(1760.0) ;
        GuitarNote g = new GuitarNote(1760.0, 1/32.0) ;
        m.play() ;
        g.play() ;
        System.out.println(g.getFrequency(0.1));
        System.out.println(g.getFrequency());

    }
}

1760.00 ---------------->  1760.00
 1760.00 -->  1815.00 -->  1760.00
1936.0000000000002
1760.0
```

Problem 3: Inheritance

a. Write a class to test the *Penguin* class that you developed in the problem 3.a. above. This class should contain the method *main()*. It should create a *Penguin* object and call each method available in that object with the exception of those methods inherited from the *Object* class.

```
public class TestPenguin {
      public static void main(String[] args) {
            Penguin p = new Penguin() ;
            System.out.println(p.song()) ;
            System.out.println(p.eats()) ;
            System.out.println(p.nameAndColors()) ;
            System.out.println(p.nameAndSize()) ;
            System.out.println(p) ;       // tests toString()
      }
}
```

b. Consider the following two class definitions:

```
public class XSuper {
    int factorP ;
    public XSuper() {
        factorP = 100 ;
    }
    public XSuper(int p) {
        factorP = p ;
    }
    public void raiseFactorP(int x) {
        factorP = factorP + x ;
    }
    public void raiseFactors(int x) {
```

```
        raiseFactorP(x) ;
    }
    public int sumFactors() {
        return factorP ;
    }
}
```

```
public class XSub extends XSuper {
    int factorQ ;
    public XSub() {
        factorQ = 200 ;
    }
    public XSub(int q) {
        factorQ = q ;
    }
    public void raiseFactorQ(int x) {
        factorQ = factorQ + x ;
    }
    public void raiseFactors(int x) {
        raiseFactorP(x) ;
        raiseFactorQ(x) ;
    }
    public int sumFactors() {
        return factorQ + super.sumFactors() ;
    }
}
```

Suppose P is declared of type XSuper and B of type XSub as in the following two statements:

```
XSuper P ;

XSub B ;
```

Which of the following would be legal Java statements for invoking an XSub or XSuper constructor:

`B = new XSuper(5) ;` NOT LEGAL	`P = new XSuper(5) ;` LEGAL
`B = new XSub(5) ;` LEGAL	`P = new XSub(5) ;` LEGAL

Which of the following would be legal Java statements for invoking an XSub or XSuper method:

`int f = B.raiseFactors() ;` NOT LEGAL	`int f = P.raiseFactors() ;` NOT LEGAL
`int f = B.sumFactors() ;` LEGAL	`int f = P.sumFactors() ;` LEGAL
`B.raiseFactorQ(5) ;`	`P.raiseFactorQ(5) ;`

LEGAL	NOT LEGAL
B.raiseFactorP(5) ; LEGAL	P.raiseFactorP(5) ; LEGAL

Problem 4: Dynamic Binding

a. Recall the Bird class from Problem 2 and given the following definition of the Penguin class:

```
public class Penguin extends Bird
{
   public Penguin()
   {   super ("Penguin", "large", "black and white");
   }

   public String song()
   {   String mySong = "bark, bark, bark";
       return mySong;
   }

   public String eats()
   {   String food = "fish";
       return food;
   }

   public boolean canFly()
   {   return false;
   }

   public boolean canSwim()
   {  return true;
   }

   public String dance()
   {  String strut = "I do the strut cause I'm cold all the time";
      return strut;
   }
}
```

Explain the indicated line of code in this version of BirdTalk.

```
public class BirdTalk
{
   public static void main(String [] args)
   {
      Bird slippy = new Penguin();
      System.out.println("Here's the Penguin's dance: " + ((Penguin)
slippy).dance());  ← EXPLAIN THIS LINE
   }
}
```

The Bird object slippy is cast into a Penguin object. Because slippy is of class Peguin, which extends Bird, this cast is successful. Then the dance method of slippy is called which returns ”I do the strut cause I’m cold all the time”.

b. Using the XSuper and XSub classes from Problem 3.b. above, what is printed when the following Java main routine is executed?

```
    public static void main(String[] args) {
        XSuper A = new XSuper() ;
        XSuper B = new XSub() ;
        XSub   C = new XSub() ;
        System.out.println("A factors are " + A.sumFactors()) ;
        System.out.println("B factors are " + B.sumFactors()) ;
        System.out.println("C factors are " + C.sumFactors()) ;

        A.raiseFactors(20) ;
        B.raiseFactors(20) ;
        C.raiseFactors(20) ;
        System.out.println("A factors are " + A.sumFactors()) ;
        System.out.println("B factors are " + B.sumFactors()) ;
        System.out.println("C factors are " + C.sumFactors()) ;
    }
```

```
A factors are 100
B factors are 300
C factors are 300
A factors are 110
B factors are 310
C factors are 310
A factors are 130
B factors are 350
C factors are 350
```

c. What is printed by the program below?

```
class Shape{
  protected final static double PI = 22.0/7.0;
  protected double length;
  public double area() {
       System.out.println(”Unknown”);
       return -1.0 ;
  }
}

class Square extends Shape{
  Square(double side){
        length=side;
  }
  public double area(){
        return length*length;
  }
}

class Circle extends Shape{
```

```
  Circle(double radius){
        length=radius;
  }
  public double area(){
        return PI*length*length;
  }
}


public class Test{
  public static void main(String[] args){
        Shape sh;
        Square sq = new Square(10.0);
        Circle circ = new Circle(10.0);

        sh=sq;
        System.out.println("Area of Square = " + sh.area());

        sh=circ;
        System.out.println("Area of circle = " + sh.area());
  }
}
```

```
Area of Square = 100.0
Area of circle = 314.2857142857143
```