Problem 1) [This problem is longer than what you will see on the quiz. You would not be asked to write as many methods, just one or two of them. This is still good practice, however!]

For this problem you need to write an encapsulated class along with its exception class. The class will be called “Votes” and the exception class “IllegalVotes”. The purpose of Votes is to store the voting records for a single polling station in a particular district. The single constructor of Votes takes four pieces of information:

- The polling station ID as a String. The string must consist of three digit characters, such as “007”.
- The district name as a String. It must consist of at least 3 characters.
- A String array of candidate names. Each name must consist of at least 3 characters and there must be at least 2 candidates in the array.
- An integer array containing the vote counts and two extra array positions which contain the number of blank ballots and the number of spoiled ballots. All values in the array must be greater than or equal to zero and the array cannot contain any empty positions.

For example, here is a code snippet that supplies a legal data set to the constructor:

```java
String district = "Sydenham";
String station = "010";
String[] candidates = {"Downes", "Foster", "Hector", "Owen", "Paterson", "Slomka"};
int[] votes = {120, 34, 110, 56, 142, 23, 12, 7};

Votes test = null;
try {
    test = new Votes(station, district, candidates, votes);
} catch (IllegalVotes e) {
    System.out.println(e.getMessage());
}
```

You can tell from these values that in Sydenham district, polling station 010, Rick Downes got 120 votes, Bryan Paterson 142 votes, and there were 12 blank ballots and 7 spoiled ballots.

The Votes object must be completely immutable. Write simple accessors for the district name, the polling station ID, the number of blank ballots and the number of spoiled ballots. Write another accessor called getVotes that accepts the name of a candidate as a String which then returns the number of votes for that candidate. If the candidate name is not in the array of candidates, then this accessor should throw an IllegalVotes exception.

The only other two methods you need are an equals method and a toString method. The equals method must override the equals method inherited from the Object class. Equality is defined as both the District name and the polling station ID being the same. The output of toString for the object test instantiated above would be:

```
Voting results for district: Sydenham, station: 010.
```
Problem 2)

Here are a bunch of classes and an interface from the same Java project:

```java
public abstract class Base {
    public int sum(int a, int b) {
        return a + b;
    }

    public int multiply(int a, int b) {
        return a * b;
    }

    public abstract int subtract(int a, int b);
}
```

```java
interface Dividing {
    int divide(int a, int b) throws DivideByZero;
}
```

```java
public class DivideByZero extends Exception {
    public DivideByZero(String message) {
        super(message);
    }
}
```

```java
public class Concrete extends Missing {
    public int multiply(int a, int b, int c) {
        return a * b * c;
    }
}
```

There are two more classes in this project. The one on the next page contains a main method that tests the other classes:
problem 2, cont.)

public class Demonstration {

    public static int remainder(int a, int b) {
        return a % b;
    }

    public static void main(String[] args) {

        Concrete test = new Concrete();    // Prints:
        System.out.println(test.sum(4, 5));  // 9
        System.out.println(test.sum(4, 5, 6));  // 15
        System.out.println(remainder(12, 5));  // 2
        System.out.println(test.multiply(4, 5));  // 20
        System.out.println(test.multiply(3, 4, 5));  // 60
        System.out.println(test.subtract(4, 5));  // -1

        try {
            System.out.println(test.divide(10, 0));    // Attempt to divide by 0!
        } catch (DivideByZero e) {
            System.out.println(e.getMessage());
        }

        try {
            System.out.println(test.divide(10, 2));  // 5
        } catch (DivideByZero e) {
            System.out.println(e.getMessage());
        }

        try {
            Dividing dTest = new Missing();
            System.out.println(dTest.divide(15, 2));  // 7
        } catch (DivideByZero e) {
            System.out.println(e.getMessage());
        }

        } // end main

    } // end Demonstration

as you can see the listing shown above contains the output of the program when run as in-line comments. One class is Missing. The Missing class extends one class and implements an interface. Write the Missing class.
Problem 3) [Attempt this problem without running the code first.]

Here are four classes contained in the same package. Write the output of each println() statement from the main method in the box beside each statement. If you think that the statement would generate a compile or runtime error, write “error” in the box instead.

```java
public class ClassRed {
    public int method1(int a) {
        return 2 * a;
    }
    public int method2(int a, int b) {
        return a * b;
    }
} // end ClassRed

public class ClassBlue extends ClassRed {
    public int method1 (int a, int b, int c) {
        return a + b + c;
    }
    public int method2 (int a, int b) {
        return 2 * super.method2(a, b);
    }
    public int method3 (int a) {
        return 2 * super.method1(a);
    }
} // end ClassBlue

public class ClassGreen extends ClassBlue {
    public int method1 (int a) {
        return 3 * a;
    }
    public int method3 (int a, int b) {
        return a + b;
    }
} // end ClassGreen
```

(Continued on the next page...)

```java
```
Problem 3, Cont.)

public class ClassMain {

    public static int method2(int a, int b) {
        return a * b;
    }

    public static void main(String[] args) {

        ClassGreen green = new ClassGreen();
        System.out.println(green.method1(1, 2, 3));

        System.out.println(green.method2(4, 5));

        System.out.println(green.method1(10));

        System.out.println(green.method3(6));

        ClassRed redGreen = new ClassGreen();
        System.out.println(redGreen.method1(7));

        System.out.println(redGreen.method2(2, 3));

        System.out.println(redGreen.method3(4, 5));

        ClassRed blue = new ClassBlue();
        System.out.println(blue.method1(5) * method2(2, 3));

        System.out.println(blue.method3(6));

        System.out.println(blue.method2(2, 3));

    }

} // end ClassMain
Problem 4)
Here is an incomplete set of classes and one interface:

```java
public interface Capitalizer {
    // Returns capitalized version of supplied string. Uses
    // the String method toUpperCase().
    String capitalizeAll(String s);
}
```

```java
public abstract class RootClass {
    // The String returned is equivalent to " " + s.
    public abstract String addLeftSpace(String s);
    // The String returned is equivalent to s + " ".
    public abstract String addRightSpace(String s);
}
```

```java
public abstract class SecondClass extends RootClass {
    public String join(String s1, String s2) {
        return s1 + s2;
    }
}
```

```java
public class MainClass {
    /*
    Generates a quote by Sam Ewing (a baseball player).
    */
    public static void main(String[] args) {
        ThirdClass third = new ThirdClass();
        String quoteL = "bikinis";
        quoteL = third.capitalizeAll(quoteL);
        quoteL = third.join(third.addRightSpace("like"), quoteL);
        quoteL = third.punctuate(quoteL);
        quoteL = third.join(third.addRightSpace("Computers are"), quoteL);
        String quoteR = third.addLeftSpace("lot of guesswork");
        quoteR = third.punctuate(quoteR);
        quoteR = third.join("They save people a", quoteR);
        quoteR = third.addLeftSpace(quoteR);
        String quote = third.join(quoteL, quoteR);
        System.out.println(quote);
    }
}
```
Problem 4, Cont.)

The missing class, ThirdClass, must extend SecondClass and implement the interface. The method punctuate() returns a String with a period added to the end of the supplied String. Write this class below:

```
What is the quotation? Enter the output of the main method below:
```

```