Building Java Programs
Sample Final Exam #1

Name of Student_____

Section (e.g., AA)_____ TA_____

This is an open-book/open-note exam. Space is provided for your answers. Use the backs of pages if necessary. The exam is divided into nine questions with the following points:

#	Problem Area	Points	Score
1	Simulation	10	_____
2	Polymorphism	6	_____
3	File processing	9	_____
4	File Processing	10	_____
5	Arrays	10	_____
6	ArrayList	10	_____
7	Critters	15	_____
8	Classes	20	_____
9	Arrays	10	_____
	Total	100	_____

In general the exam is not graded on style and you do not need to include comments. However, all data fields in a class must be declared private and if a problem statement indicates that a method should be public, it must be declared as public.

You do not have to include any import statements.

You may use a calculator to do simple arithmetic, although you are not allowed to program the calculator or to run a stored program.

Do not begin work on this exam until instructed to do so. Any student who starts early or who continues to work after time is called will receive a 10 point penalty.

If you finish the exam early, please hand your exam to Stuart and exit quietly through the front door.

1. Simulation, 10 points. You are to simulate the execution of a method that manipulates an array of integers. Consider the following method:

```
public static int mystery(int[] list) {
    int x = 0;
    for (int i = 0; i < list.length - 1; i++)
        if (list[i] > list[i + 1])
            x++;
    return x;
}
```

Notice that this method takes an integer array as a parameter and returns an integer. In the left-hand column below are specific lists of integers. You are to indicate in the right-hand column what value would be returned by method mystery if the integer list in the left-hand column is passed as a parameter to mystery.

List	Value returned
(8)	_____
(14, 7)	_____
(7, 1, 3, 2, 0, 4)	_____
(10, 8, 9, 5, 6)	_____
(8, 10, 8, 6, 4, 2)	_____

2. Polymorphism, 6 points. Assume the following classes have been defined:

```
public class Pen extends Sock {
    public void method1() {
        System.out.println("pen 1");
    }
}

public class Lamp {
    public void method1() {
        System.out.println("lamp 1");
    }

    public void method2() {
        System.out.println("lamp 2");
    }

    public String toString() {
        return "lamp";
    }
}

public class Book extends Sock {
    public void method2() {
        System.out.println("book 2");
    }
}

public class Sock extends Lamp {
    public void method1() {
        System.out.println("sock 1");
    }

    public String toString() {
        return "sock";
    }
}
```

Consider the following code fragment:

```
Lamp[] elements = {new Book(), new Pen(), new Lamp(), new Sock()};
for (int i = 0; i < elements.length; i++) {
    System.out.println(elements[i]);
    elements[i].method1();
    elements[i].method2();
    System.out.println();
}
```

What output is produced by this code?

3. File Processing, 9 points. Write a static method processData that takes a Scanner holding a sequence of words and that reports the total number of words and the average word length. For example, suppose the Scanner contains the following words.

```
To be or not to be, that is the question.
```

For the purposes of this problem, we will use whitespace to separate words. That means that some words include punctuation, as in "be,". This is the same definition that the Scanner uses for tokens.

For the input above, your method should produce the following output.

```
Total words    = 10
Average length = 3.2
```

You are to exactly reproduce the format of this output.

Write your solution to processData below.

4. File Processing, 10 points. Write a static method processFile that takes a Scanner containing an input file as a parameter and that writes to System.out the same file with successive pairs of lines reversed in order. For example, if the input file contains the following text:

```
Twas brillig and the slithy toves
did gyre and gimble in the wabe.
All mimsey were the borogroves,
and the mome raths outgrabe.

"Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch."
```

The program should print the first pair of lines in reverse order, then the second pair in reverse order, then the third pair in reverse order, and so on. Thus, your method should produce the following output.

```
did gyre and gimble in the wabe.
Twas brillig and the slithy toves
and the mome raths outgrabe.
All mimsey were the borogroves,
"Beware the Jabberwock, my son,

Beware the JubJub bird and shun
the jaws that bite, the claws that catch,
the frumious bandersnatch."
```

Notice that a line can be blank, as in the third pair. Also notice that an input file can have an odd number of lines, as in the one above, in which case the last line is printed in its original position. You may not make any assumptions about how many lines are in the Scanner.

Write your solution to processFile below.

5. Arrays, 10 points. Write a static method minGap that takes an integer array as a parameter and that returns the minimum gap between adjacent values in the array. The gap between two adjacent values in a list is defined as the second value minus the first value. For example, suppose a variable called "list" is an array of integers that stores the following sequence of values.

    (1, 3, 6, 7, 12)

   The first gap is 2 (3 - 1), the second gap is 3 (6 - 3), the third gap is 1 (7 - 6) and the fourth gap is 5 (12 - 7). Thus, the call:

    minGap(list)

   Should return 1 because that is the smallest gap in the list. Notice that the minimum gap could be a negative number. For example, if list stores the following sequence of values:

    (3, 5, 11, 4, 8)

   The gaps would be computed as 2 (5 - 3), 6 (11 - 5), -7 (4 - 11), and 4 (8 - 4). Of these values, -7 is the smallest, so it would be returned.

   This gap information can be helpful for determining other properties of the list. For example, if the minimum gap is greater than or equal to 0, then you know the list is in sorted (nondecreasing) order. If the gap is greater than 0, then you know the list is both sorted and unique (strictly increasing).

   If you are passed a list with fewer than 2 elements, you should return 0.

   Write your solution to minGap below.

6. ArrayList, 10 points. Write a static method markLength4 that takes an ArrayList of Strings as a parameter and that places a String of four asterisks ("****") in front of every String of length 4. For example, suppose that an ArrayList called "list" contains the following values:

```
(this, is, lots, of, fun, for, every, Java, programmer)
```

And you make the following call:

```
markLength4(list);
```

Then list should store the following values after the call:

```
(****, this, is, ****, lots, of, fun, for, every, ****, Java, programmer)
```

Notice that you leave the original Strings in the list (this, lots, Java) but include the four-asterisk String in front of each to mark it. You may assume that the ArrayList contains only String values, but it might be empty. Recall that the primary methods for manipulating an ArrayList are:

add(Object value)	appends value at end of list
add(int index, Object value)	inserts given value at given index, shifting subsequent values right
get(int index)	returns the value at given index
remove(int index)	removes value at given index, shifting subsequent values left
set(int index, Object value)	replaces value at given index with given value
size()	returns the number of elements in list

Write your solution to markLength4 below.

7. Critters, 15 points. Write a class Quail that implements the Critter interface. The instances of the Quail class go through cycles where they move north a fixed number of times, then west the same number of times, then east the same number of times, then south the same number of times, which brings them back to their starting point. They either move 10 in each of these directions or 20 in each of these directions. Each time they are about to begin a cycle, they randomly choose between using 10 or 20 for the next cycle (each choice having an equal probability). Different Quail objects will make different choices. So one Quail object might initially decide to move 10 in each of the four directions while another might decide to move 20 in each of the four directions. Each Quail object should return the letter "Q" for display purposes.

   Remember that the Critter interface is defined as follows:

```
public interface Critter {
    public char getChar();

    public int getMove();

    <definitions for constants NORTH, SOUTH, EAST and WEST>
}
```

   Write your solution to the Quail class below.

8. Classes, 20 points. Write a class PhoneCard that keeps track of a prepaid card used to make long distance calls. Each card will have an initial amount of credit which goes down as calls are made. The basic charge for phone calls is 10 cents per minute, but every call has a 25 cent surcharge for initiating the call and some calls are charged a higher rate per minute (e.g., if calling overseas). It should have the following public methods.

   * a constructor that takes an initial credit to place on the card as a parameter (a value of type double).

   * a method "call" that takes an integer number of minutes as a parameter and that charges the card for that number of minutes.

   * a method "call" that takes an integer number of minutes and a multiplier of type double as parameters and that charges the card for that number of minutes at the standard rate multiplied by the multiplier.

   * a method getCredit() that returns the remaining credit on the card.

   * a method getAverage() that returns the average cost per minute for the calls that have been made.

A typical manipulation would look like this:

```
PhoneCard card = new PhoneCard(20.00);  // new card with $20 credit

card.call(10);  // standard 10-minute call
card.call(5);   // standard 5-minute call
System.out.println("money left    = $" + card.getCredit());
System.out.println("average spent = $" + card.getAverage());

card.call(6, 2);    // 6-minute call at 2 times the normal rate
card.call(10, 2.5); // 10-minute call at 2.5 times the normal rate
System.out.println("money left    = $" + card.getCredit());
System.out.println("average spent = $" + card.getAverage());
```

The card is constructed with $20 credit. The first two calls that are made use the standard calling rate of 10 cents per minute. A total of $2 is spent because each call has the standard 25 cent surcharge for initiating the call. Averaging the $2 cost over these 15 minutes of calls you get an average per-minute rate of 13 and one third cents. Thus, the first two lines of output from the code above are as follows.

```
money left    = $18.0
average spent = $0.13333333333333333
```

Then two calls are made that involve multipliers. The first uses a multiplier of 2, which increases the per-minute rate to 20 cents. The second uses a multiplier of 2.5, which increases the per-minute rate to 25 cents. Each uses the standard 25 cent surcharge for initiating the call. Thus, these calls cost an additional $4.20, which leaves the card with $13.80 in credit. At this point the card has been used to make 31 minutes worth of calls, which gives an average per-minute rate of 20 cents. Thus, the next two lines of output are as follows.

```
money left    = $13.8
average spent = $0.2
```

Notice that the 25 cent surcharge is the same for all calls, whether or not a multiplier is being used. If the getAverage method is called before any calls have been made on the card, it should return 0.1 to indicate the standard ten cent rate. Assume that no call would cost more than the credit available on the card.

Space is provided on the next page for your definition of class PhoneCard.

Provide your definition for class PhoneCard below.

9. Arrays, 10 points. Write a static method numSame that compares two sorted arrays of integers, returning the number of values that are the same between the two. For example, if two sorted arrays store the following values:

```
list1:  (2, 4, 6, 20)
list2:  (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Then the method call:

```
numSame(list1, list2)
```

would return 3, because three of the numbers are the same in each list (i.e., 2, 4 and 6). You should get the same result from this call:

```
numSame(list2, list1)
```

There may be duplicates in either list. Any given number in a list can match at most once, so a duplicate in one list can match a duplicate in another, but if one list has fewer duplicates than the other, then only that smaller number of duplicates can match. For example, if the lists store:

```
list1:  (1, 1, 1, 1, 2, 2, 3, 4, 5, 8, 12)
list2:  (1, 1, 3, 3, 3, 3, 6, 6, 7, 7, 8, 9, 10, 11)
```

Then the method call:

```
numSame(list1, list2)
```

would return 4, because 4 of the values match (two of the 1's, one 3 and 8). Notice that only two of the 1's in list1 match because list2 has only two 1's. Also notice that even though list2 has four 3's, only one ends up matching because list1 has only one 3. As before, the method should return the same result if the order of the lists is reversed in the call.

Remember that the numbers in the two arrays will be in sorted (nondecreasing) order. You must solve this problem using the two arrays that are passed as parameters. You are not allowed to use additional arrays or another structured object like an ArrayList. You also may not modify the arrays that are passed as parameters. You can, however, receive up to 4 of the 10 points if your solution works for lists that have no duplicates.

Write your solution to numSame below.