CSE 142, Spring 2009, Sample Final Exam #1

Name: _____

Section: _____ **TA:** _____

Student ID #: _____

Rules:

- You have 110 minutes to complete this exam.
  You will receive a deduction if you keep working after the instructor calls for papers.
- This test is open-book/notes.
- You may not use any electronic device of any kind including calculators.
- Your code will be graded on proper behavior/output, not on style.
- You do not need to write any `import` statements in your code.
- Do not abbreviate any code that you write.
- Do not abbreviate any answer asking for output (show the complete output).
- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.
- You must show your Student ID to a TA or instructor for your submitted exam to be accepted.

Good luck!

Problem	Description	Earned	Max
1	Array Simulation		10
2	Inheritance		7
3	Parameters / References		8
4	Token-Based File Processing		8
5	Line-Based File Processing		10
6	Arrays		10
7	ArrayList		8
8	Critters		15
9	Arrays		14
10	Programming		10
TOTAL	**Total Points**		**100**

1. Array Simulation, 10 points

Consider the following method:

```
public static void arrayMystery(int[] a) {
    for (int i = a.length - 2; i > 0; i--) {
        if (a[i - 1] < a[i + 1]) {
            a[i] += a[i - 1];
        } else {
            a[i] += a[i + 1];
        }
    }
}
```

Indicate in the right-hand column what values would be stored in the array after the method `arrayMystery` executes if the array in the left-hand column is passed as its parameter.

Original Contents of Array	Final Contents of Array
`int[] a1 = {1, 2, 3};` `arrayMystery(a1);`	_____
`int[] a2 = {8, 2, 3, 1, 6};` `arrayMystery(a2);`	_____
`int[] a3 = {1, 1, 1, 1, 1, 1};` `arrayMystery(a3);`	_____
`int[] a4 = {40, 10, 25, 5, 10, 30};` `arrayMystery(a4);`	_____
`int[] a5 = {15, 6, -1, 4, 8, -2, 7, 4};` `arrayMystery(a5);`	_____

2. Inheritance, 7 points

Assume the following four classes have been defined:

```
public class McCain extends Biden {
    public void republican() {
        System.out.print("mccain-R   ");
    }
}

public class Palin {
    public void republican() {
        System.out.print("palin-R   ");
    }

    public void democrat() {
        republican();
        System.out.print("palin-D   ");
    }

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

```
public class Obama extends Palin {
    public void republican() {
        super.republican();
        System.out.print("obama-R   ");
    }
}

public class Biden extends Palin {
    public String toString() {
        return "biden";
    }

    public void democrat() {
        System.out.print("biden-D   ");
        super.democrat();
    }
}
```

Given the classes above, what output is produced by the following code?

```
Palin[] politicians = {new Biden(), new Palin(), new McCain(), new Obama()};
for (int i = 0; i < politicians.length; i++) {
    System.out.println(politicians[i]);
    politicians[i].republican();
    System.out.println();
    politicians[i].democrat();
    System.out.println();
    System.out.println();
}
```

3. Parameters and References, 8 points

The program below produces 4 lines of output. What are they?

```
public class VerySimplePoint {
  int x;
  int y;
  public VerySimplePoint() {
     x = 2;
     y = 2;
  }
}

public class Mystery {
  public static void main(String[] args) {
     int a = 7;
     int b = 9;
     VerySimplePoint p1 = new VerySimplePoint();
     VerySimplePoint p2 = new VerySimplePoint();
     addToXTwice(a,p1);
     System.out.println(a + " " + b + " " + p1.x + " " + p2.x);
     addToXTwice(b,p2);
     System.out.println(a + " " + b + " " + p1.x + " " + p2.x);
  }

  public static void addToXTwice(int a, VerySimplePoint p1) {
     a = a + a;
     p1.x = a;
     System.out.println(a + " " + p1.x);
  }
}
```

Line 1: _____

Line 2: _____

Line 3: _____

Line 4: _____

4. Token-Based File Processing, 8 points

Write a static method called `printStrings` that takes as a parameter a `Scanner` holding a sequence of integer/String pairs and that prints to `System.out` one line of output for each pair with the given String repeated the given number of times. For example if the `Scanner` contains the following data:

```
6 fun. 3 hello 10 <> 4 wow!
```

your method should produce the following output:

```
fun.fun.fun.fun.fun.fun.
hellohellohello
<><><><><><><><><><>
wow!wow!wow!wow!
```

Notice that there is one line of output for each integer/String pair. The first line has 6 occurrences of "fun.", the second line has 3 occurrences of "hello", the third line has 10 occurrences of "<>" and the fourth line has 4 occurrences of "wow!". Notice that there are no extra spaces included in the output. You are to exactly reproduce the format of this sample output. You may assume that the input values always come in pairs with an integer followed by a String. If the Scanner is empty (no integer/String pairs), your method should produce no output.

5. Line-Based File Processing, 10 points

Write a static method named `printDuplicates` that accepts as its parameter a `Scanner` for an input file containing a series of lines. Your method should examine each line looking for consecutive occurrences of the same token on the same line, and print each duplicated token along how many times it appears consecutively. Non-repeated tokens are not printed. Repetition across multiple lines (such as if a line ends with a given token and the next line starts with the same token) is not considered in this problem.

For example, if the input file contains the following text (sequences of duplicated tokens are underlined for emphasis):

```
hello how how are you you you you
I I I am Jack's Jack's smirking smirking smirking smirking smirking revenge
   bow  wow wow yippee yippee   yo yippee   yippee yay  yay yay
one fish two fish red fish blue fish
It's the Muppet Show, wakka wakka wakka
```

Your method would produce the following output for the preceding input file:

```
how*2 you*4
I*3 Jack's*2 smirking*5
wow*2 yippee*2 yippee*2 yay*3

wakka*3
```

Your code prints only the repeated tokens; the ones that appear only once in a row are not shown. Your code should place a single space between each reported duplicate token and should respect the line breaks in the original file. This is why a blank line appears in the expected output, corresponding to the fourth line of the file that did not contain any consecutively duplicated tokens. You may assume that each line of the file contains at least 1 token of input.

6. Arrays, 10 points

Write a static method named `arraySum` that accepts two arrays of real numbers *a1* and *a2* as parameters and returns a new array *a3* such that each element of *a3* at each index *i* is the sum of the elements at that same index *i* in *a1* and *a2*. For example, if *a1* stores `{4.5, 5.0, 6.6}` and *a2* stores `{1.1, 3.4, 0.5}`, your method should return `{5.6, 8.4, 7.1}`, which is obtained by adding 4.5 + 1.1, 5.0 + 3.4, and 6.6 + 0.5.

If the arrays *a1* and *a2* are not the same length, the result returned by your method should have as many elements as the larger of the two arrays. If a given index *i* is in bounds of *a1* but not *a2* (or vice versa), your result array's element at index *i* should be equal to the value of the element at index *i* in the longer of *a1* or *a2*. For example, if *a1* stores `{1.8, 2.9, 9.4, 5.5}` and *a2* stores `{2.4, 5.0}`, your method should return `{4.2, 7.9, 9.4, 5.5}`.

The table below shows some additional calls to your method and the expected values returned:

Arrays	Call and Value Returned
`double[] a1 = {4.5, 2.8, 3.4, 0.8};` `double[] a2 = {1.4, 8.9, -1.0, 2.3};`	`arraySum(a1, a2)` returns `{5.9, 11.7, 2.4, 3.1}`
`double[] ax = {2.4, 3.8};` `double[] ay = {0.2, 9.2, 4.3, 2.8, 1.4};`	`arraySum(ax, ay)` returns `{2.6, 13.0, 4.3, 2.8, 1.4}`
`double[] aa = {1.0, 2.0, 3.0};` `double[] ab = {4.0, 5.0};`	`arraySum(aa, ab)` returns `{5.0, 7.0, 3.0}`
`double[] ai = {};` `double[] aj = {42.0};`	`arraySum(ai, aj)` returns `{42.0}`

For full credit, you should not modify the elements of *a1* or *a2*. You may not use a `String` to solve this problem.

7. ArrayList, 8 points

Write a static method called `reverse3` that takes an `ArrayList` of integer values as a parameter and that reverses each successive sequence of three values in the list. For example, suppose that a variable called `list` stores the following sequence of values:

```
[3, 8, 19, 42, 7, 26, 19, -8, 193, 204, 6, -4]
```

and we make the following call:

```
reverse3(list);
```

Afterwards the list should store the following sequence of values:

```
[19, 8, 3, 26, 7, 42, 193, -8, 19, -4, 6, 204]
```

The first sequence of three values (3, 8, 19) has been reversed to be (19, 8, 3). The second sequence of three values (42, 7, 26) has been reversed to be (26, 7, 42). And so on. If the list has extra values that are not part of a sequence of three, those values are unchanged. For example, if the list had instead stored:

```
[3, 8, 19, 42, 7, 26, 19, -8, 193, 204, 6, -4, 99, 2]
```

The result would have been:

```
[19, 8, 3, 26, 7, 42, 193, -8, 19, -4, 6, 204, 99, 2]
```

Notice that the values (99, 2) are unchanged in position because they were not part of a sequence of three values.

8. Critters, 15 points

Write a class called `Orca` that extends the `Critter` class. The instances of the Orca class follow a pattern of moving forward four times, then turning around, then moving back four times and turning around again so that they return to their original position and direction. Each Orca is always either in moving-mode or in turning-mode. They start out in moving-mode. While in moving-mode, they try to hop forward if possible until they have hopped four times, at which point they switch into turning-mode. If it is not possible to hop while in moving-mode, an Orca instead infects whatever is in front of it. When in turning-mode, the Orca turns left twice and then switches back to moving-mode. Don't worry about the fact that if the Orca encounters a wall while in moving-mode, it gets stuck trying to infect the wall indefinitely. The Orca displays itself as "M" while in moving-mode and as "T" while in turning-mode. Its color should be the default color for critters.

9. Arrays, 14 points

Write a static method named `partition` that accepts an array of integers *a* and an integer element value *v* as its parameters, and rearranges ("partitions") the array's elements so that all its elements of *a* that are less than *v* occur before all elements that are greater than *v*. The exact order of the elements is unimportant so long as all elements less than v appear before all elements greater than *v*. For example, if your method were passed the following array:

```
int[] a = {15, 1, 6, 12, -3, 4, 8, -7, 21, 2, 30, -1, 9};
partition(a, 5);
```

One acceptable ordering of the elements after the call would be: (elements < 5 and > 5 are underlined for emphasis)

{-1, 1, 2, -7, -3, 4, 8, 12, 21, 6, 30, 15, 9}

Hint: Your method will need to rearrange the elements of the array, which will involve swapping various elements from less desirable indexes to more desirable ones.

You may assume that the array contains no duplicates and does not contain the element value *v* itself. You may not use `Arrays.sort`, `Collections.sort`, or any other pre-defined sorting algorithm from the Java Class Libraries or textbook to solve this problem. You also may not use a `String` to solve this problem.

10. Programming, 10 points

Write a static method called `acronym` that takes as a parameter a `String` containing a phrase and that returns an acronym for the phrase. For example, the following call:

```
acronym("self-contained underwater breathing apparatus")
```

should return `"SCUBA"`. The acronym is formed by combining the capitalized first letters of each word in the phrase. Words in the phrase will be separated by some combination of dashes and spaces. There might be extra spaces or dashes at the beginning or end of the phrase. The String will not contain any characters other than dashes, spaces, and letters, and is guaranteed to contain at least one word. Below are several sample calls.

Method Call	Value Returned
`acronym("  automatic  teller  machine  ")`	`"ATM"`
`acronym("personal identification number")`	`"PIN"`
`acronym("computer science")`	`"CS"`
`acronym("merry-go-round")`	`"MGR"`
`acronym("All my Children")`	`"AMC"`
`acronym("Troubled Assets Relief Program")`	`"TARP"`
`acronym("--quite-- confusing - punctuation-")`	`"QCP"`
`acronym("  loner  ")`	`"L"`