CSE 142, Summer 2008
Final Exam
Friday, August 22, 2008

Name: ____

Section: ____________________ **TA:** ____________________

Student ID #: ____________________

Rules:

- You have 60 minutes to complete this exam.
  You may receive a deduction if you keep working after the instructor calls for papers.
- This test is open-book/notes.
- You may not use any computing devices of any kind including calculators.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- You do not need to write any `import` statements in your code.
- Please do not abbreviate code, such as writing ditto marks ("") or dot-dot-dot marks (...).
- 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	Expressions		15
2	Array Mystery		15
3	Inheritance Mystery		15
4	File Processing		15
5	Array Programming		15
6	Critters		15
7	Array Programming		10
X	Extra Credit		+1
TOTAL	**Total Points**		**100**

1. Expressions (15 points)

For each expression in the left-hand column, indicate its value in the right-hand column.
Be sure to list a constant of appropriate type and capitalization.
e.g., `7` for an `int`, `7.0` for a `double`, `"hello"` for a `String`, `true` or `false` for a `boolean`.

Expression — Value

`1 + 9 / 2 * 2.0` ________________________________

`5.0 / (3125 % 2) + 2 * (5 / 3)` ________________________________

`6 % 17 + 9 % 3 + 22 / 4 / 2.0` ________________________________

`"[" + 2 + 4 * 2.0 + "]" + 3` ________________________________

`!(3 < 2) && (4.3 > 3 || 3 < 2)` ________________________________

2. Array Mystery (15 points)

Consider the following method:

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

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 a parameter to it.

Original Contents of Array — Final Contents of Array

```
int[] a1 = {"a", "b", "c"};
arrayMystery(a1);
```
____

```
int[] a2 = {"a", "bb", "c", "dd"};
arrayMystery(a2);
```
____

```
int[] a3 = {"z", "y", "142", "w", "xx"};
arrayMystery(a3);
```
____

3. Inheritance Mystery (15 points)

Assume that the following three classes have been defined:

```
public class Dog extends Cat {
   public void m1() {
      m2();
      System.out.print("dog 1   ");
   }
}

public class Lion extends Dog {
   public void m2() {
      System.out.print("lion 2   ");
      super.m2();
   }

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

```
public class Cat {
   public void m1() {
      System.out.print("cat 1   ");
   }

   public void m2() {
      System.out.print("cat 2   ");
   }

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

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

```
Cat[] elements = {new Dog(), new Cat(), new Lion()};
for (int i = 0; i < elements.length; i++) {
    elements[i].m1();
    System.out.println();
    elements[i].m2();
    System.out.println();
    System.out.println(elements[i]);
    System.out.println();
}
```

4. File Processing (15 points)

Write a static method `blackjack` that accepts as its parameter a `Scanner` for an input file containing a hand of playing cards, and returns the point value of the hand in the card game Blackjack.

A card has a rank and a suit. There are 13 ranks: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, and King. There are 4 suits: Clubs, Diamonds, Hearts, and Spades. A Blackjack hand's point value is the sum of its cards' point values. A card's point value comes from its rank; the suit is irrelevant. In this problem, cards are worth the following points:

Rank	**Point Value**
2-10	The card's rank (for example, a 7 is worth 7 points)
Jack (`J`), Queen (`Q`), King (`K`)	10 points each
Ace (`A`)	11 points *(for this problem; simplified compared to real Blackjack)*

The input file contains a single hand of cards, each represented by a pair of "*<rank> <suit>*" tokens. For example:

```
5 Diamonds
Q Spades
2 Spades  3 Hearts
```

Given the above input, your method should return 20, since the cards' point values are 5 + 10 + 2 + 3 = 20.

The input can be in mixed casing, have odd spacing between tokens, and can be split across lines. For example:

```
2 Hearts
  j SPADES   a  Diamonds
2 ClUbS
   A
hearts
```

Given the above input, your method should return 36, since the cards' point values are 2 + 10 + 11 + 2 + 11 = 36.

You may assume that the `Scanner` contains at least 1 card (two tokens) of input, and that no line will contain any tokens other than valid card data. The real game of Blackjack has many other rules that you should ignore for this problem, such as the notion of going "bust" once you exceed a score of 21.

5. Array Programming (15 points)

Write a static method named `allPlural` that accepts an array of strings as a parameter and returns `true` only if every string in the array is a plural word, and `false` otherwise. For this problem a plural word is defined as any string that ends with the letter S, case-insensitively. The empty string `""` is *not* considered a plural word, but the single-letter string "s" or "S" is. Your method should return `true` if passed an empty array (one with 0 elements).

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

Array	Call and Value Returned
`String[] a1 = {"snails", "DOGS", "Cats"};`	`allPlural(a1)` returns `true`
`String[] a2 = {"builds", "Is", "S", "THRILLs", "CS"};`	`allPlural(a2)` returns `true`
`String[] a3 = {};`	`allPlural(a3)` returns `true`
`String[] a4 = {"She", "sells", "sea", "SHELLS"};`	`allPlural(a4)` returns `false`
`String[] a5 = {"HANDS", "feet", "toes", "OxEn"};`	`allPlural(a5)` returns `false`
`String[] a6 = {"shoes", "", "socks"};`	`allPlural(a6)` returns `false`

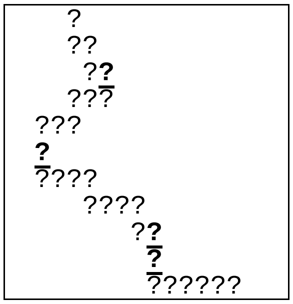
For full credit, your method should not modify the array's elements.

6. Critters (15 points)

Write a class `Minnow` that extends `Critter` from HW8, along with its movement and eating behavior. All other aspects of `Minnow` use the defaults. Add fields, constructors, etc. as necessary to your class.

`Minnow` objects initially move in a S/E/S/E/... pattern. However, when a `Minnow` encounters food (when its `eat` method is called), it should do all of the following:

- **Do *not* eat the food.**
- **Start the movement cycle over.** In other words, the next move after `eat` is called should always be South.
- **Lengthen and reverse the horizontal portion of the movement cycle pattern.**
  The `Minnow` should reverse its horizontal direction and increase its horizontal movement distance by 1 for subsequent cycles. For example, if the `Minnow` had been moving S/E/S/E, it will now move S/**W/W**/S/**W/W**. If it hits a second piece of food, it will move S/**E/E/E**/S/**E/E/E**, and a third, S/**W/W/W/W**/S/**W/W/W/W**, and so on.



The following is an example timeline of a particular `Minnow` object's movement. The timeline below is also drawn in the diagram at right. Underlined occurrences mark squares where the `Minnow` found food.

- S, E, S, E (hits food)
- S, W, W, S, W, W, S (hits food)
- S, E, E, E, S, E, E, E, S, E (hits food)
- S (hits food)
- S, E, E, E, E, E, S, E, E, E, E, E, ...

7. Array Programming (10 points)

Write a static method named `reverseChunks` that accepts two parameters, an array of integers *a* and an integer "chunk" size *s*, and reverses every *s* elements of *a*. For example, if *s* is 2 and array *a* stores `{1, 2, 3, 4, 5, 6}`, *a* is rearranged to store `{2, 1, 4, 3, 6, 5}`. With an *s* of 3 and the same elements `{1, 2, 3, 4, 5, 6}`, array *a* is rearranged to store `{3, 2, 1, 6, 5, 4}`. The chunks on this page are underlined for convenience.

If *a*'s length is not evenly divisible by *s*, the remaining elements are untouched. For example, if *s* is 4 and array *a* stores `{5, 4, 9, 2, 1, 7, 8, 6, 2, 10}`, *a* is rearranged to store `{2, 9, 4, 5, 6, 8, 7, 1, 2, 10}`. It is also possible that *s* is larger than *a*'s entire length, in which case the array is not modified at all. You may assume that *s* is 1 or greater (an *s* of 1 would not modify the array). If array *a* is empty, its contents should remain unchanged.

The following table shows some calls to your method and their expected results:

Array and Call	Array Contents After Call

`int[] a1 = {20, 10, 30, 60, 50, 40};`
`reverseChunks(a1, 2);`
	`{10, 20, 60, 30, 40, 50}`

`int[] a2 = {2, 4, 6, 8, 10, 12, 14, 16};`
`reverseChunks(a2, 3);`
	`{6, 4, 2, 12, 10, 8, 14, 16}`

`int[] a3 = {7, 1, 3, 5, 9, 8, 2, 6, 4, 10, 0, 12};`
`reverseChunks(a3, 5);`
	`{9, 5, 3, 1, 7, 10, 4, 6, 2, 8, 0, 12}`

`int[] a4 = {1, 2, 3, 4, 5, 6};`
`reverseChunks(a4, 8);`
	`{1, 2, 3, 4, 5, 6}`

`int[] a5 = {};`
`reverseChunks(a5, 2);`
	`{}`

X. Extra Credit (+1 point)

Describe your TA in 5 words or less.

(Any words that appear to reflect a nontrivial effort will receive the bonus point.)