CSE 142, Winter 2007
Final Exam

1. Expressions (5 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 (e.g., `7.0` rather than `7` for a `double`, `String`s in "quotes").

Expression	**Value**
`22 / 4 / 2.0 - 1.0 / 2`	_____
`1 < 2 && (3 == 4 || 5 != 5)`	_____
`10 % 25 + 10 % (11 % 4)`	_____
`1 + 2 + "3 + 4" + 5 + "(int) 6.0"`	_____
`5 < 3 || !(2 + 2 >= 4)`	_____

2. Array Mystery (10 points)

Consider the following method:

```
public static void mystery(int[] array) {
    for (int i = 1; i < array.length - 1; i++) {
        array[i] = array[i - 1] + array[i + 1];
    }
}
```

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

Method Call	**Final Array Contents**
`int[] a1 = {42};` `mystery(a1);`	_____
`int[] a2 = {8, 2, 3};` `mystery(a2);`	_____
`int[] a3 = {14, 7, 5, 6};` `mystery(a3);`	_____
`int[] a4 = {1, 0, 1, 0, 1, 0};` `mystery(a4);`	_____
`int[] a5 = {10, 1, 4, 2, 6, 8, 1};` `mystery(a5);`	_____

3. Inheritance Mystery (10 points)

Assume that the following classes have been defined:

```
public class Superman extends Batman {
    public void methodB() {
        System.out.println("Superman B");
    }

    public String toString() {
        return "Up up and away!";
    }
}

public class Hulk {
    public String toString() {
        return "Hulk smash!";
    }

    public void methodA() {
        System.out.println("Hulk A");
    }

    public void methodB() {
        System.out.println("Hulk B");
    }
}
```

```
public class WonderWoman extends Superman {
    public void methodA() {
        System.out.println("WonderWoman A");
    }
}

public class Batman extends Hulk {
    public void methodA() {
        System.out.println("Batman A");
    }

    public String toString() {
        return "I'm Batman.";
    }
}
```

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

```
Hulk[] heroes = {new Batman(), new WonderWoman(), new Hulk(), new Superman()};
for (int i = 0; i < heroes.length; i++) {
    System.out.println(heroes[i]);
    heroes[i].methodA();
    heroes[i].methodB();
    System.out.println();
}
```

4. File Processing (15 points)

Write a method named `count` that accepts two parameters: a `Scanner` holding a sequence of words, and a `String` representing a word. The method should count how many times that word occurs in the input, case-insensitively. For the purposes of this problem, we will use whitespace to separate words. (This is the same definition that the `Scanner` uses to separate tokens.)

For example, suppose the `Scanner` contains the following words:

```
The man on the hill by  THE
   plain    with

the telescope  saw them    there.
```

For a Scanner variable named `input` referring to the input above, the call of `count(input, "the")` should return `4`. The call of `count(input, "wit")` should return `0`.

5. File Processing (10 points)

Write a method named `analyzeStocks` that accepts as a parameter a `Scanner` for an input file. Each line of input holds a person's name followed by some number of stock purchases. Each stock purchase contains the stock's symbol, the price at which it was bought, and the price at which it was sold. The method should output to the console each person's name and total profit, which is the difference between the total amount for all stocks bought and the total amount for all stocks sold. For example, consider the following input file:

```
Victoria INTC 18.8 20.8
Andrew MSFT 27.1 29.3 YHOO 44.6 40.6 GOOG 382.1 379.6
Justine
Josh NVDA 10.0 29.1 BNSN 7.11 19.79 VIA 29.0 39.5 MRTY 15.90 25.0
```

For a `Scanner` variable named `input` referring to the input above, the call of `analyzeStocks(input)` should produce the following output:

```
Victoria: total profit = $2.0
Andrew: total profit = $-4.300000000000011
Justine: total profit = $0.0
Josh: total profit = $40.88
```

The format of your output must exactly match that shown above. Notice that some people (such as Justine above) might buy no stocks, resulting in a profit of 0. Do not worry about rounding the average words per line. You may assume that each line contains valid input.

6. Arrays (15 points)

Write a method named `isReverse` that accepts two arrays of integers as parameters and returns `true` if they contain the same elements in the opposite order, and `false` otherwise. For example, if two variables named `list1` and `list2` store the following values:

```
int[] list1 = {10, 15, 24, 32, 19};
int[] list2 = {19, 32, 24, 15, 10};
```

Then the call of `isReverse(list1, list2)` should return `true`.

For your method to return `true`, the arrays must be the same length. You may assume that each array's length is at least 1.

7. Arrays (10 points)

Write a method named `wrapHalf` that accepts an array of integers as a parameter and rearranges its elements so that the first half and second half of the array are switched. For example, if an array named `list1` stores the values `{10, 20, 30, 40, 50, 60}`, after the call of `wrapHalf(list1)` the array should store the values `{40, 50, 60, 10, 20, 30}`.

If the length of the array is odd, consider the first half to be the larger half and the second half to be the smaller. For example, if an array named `list2` stores the values `{1, 2, 3, 4, 5}`, after the call of `wrapHalf(list2)` the array should store the values `{4, 5, 1, 2, 3}`.

You may use an auxiliary data structure or array to help you solve this problem if you like. It may also help you to think of the task of wrapping as performing many rotations on the array. You may assume that the array passed to your method will have a length of at least 2.

8. Critters (15 points)

Write the `getMove` method for the class `Baboon` that implements the `Critter` interface from Homework 8. Instances of the `Baboon` class should move in a clockwise "spiral" pattern as follows: `NORTH` 1 time, `EAST` 2 times, `SOUTH` 3 times, `WEST` 4 times, `NORTH` 5 times, `EAST` 6 times, `SOUTH` 7 times, `WEST` 8 times, then repeats.

Use the constants for directions defined in the `Critter` interface. You may add anything needed (fields, other methods, constructors, etc.) to implement `getMove` appropriately.

```
import java.awt.*;   // for Color

public class Baboon implements Critter {
    // declare any necessary fields here




    // fight, getColor, toString methods omitted (you do not need to write them)

    public int getMove(CritterInfo info) {
        // complete the getMove method here
```

9. Classes (10 points)

This question uses the `Date` class as specified at right (this is the same Date class as seen on past sample final exams).

Write a method named `subtractWeeks` that will be placed inside the `Date` class. The `subtractWeeks` method accepts an integer as a parameter and shifts the date represented by the `Date` object backward by that many weeks. A week is considered to be exactly 7 days. You may assume the value passed is non-negative. Note that subtracting weeks might cause the date to wrap into previous months or years.

For example, if the following `Date` is declared in client code:

```
Date d = new Date(9, 19);
```

The following calls to the `subtractWeeks` method would modify the `Date` object's state as indicated in the comments. Remember that `Date` objects do not store the year; the date before January 1st is December 31st. `Date` objects also ignore leap years.

```
Date d = new Date(9, 19);
d.subtractWeeks(1);    // d is now 9/12
d.subtractWeeks(2);    // d is now 8/29
d.subtractWeeks(5);    // d is now 7/25
d.subtractWeeks(20);   // d is now 3/7
d.subtractWeeks(110);  // d is now 1/26
                       //  (2 years prior)
```

```
public class Date {
    private int month;
    private int day;

    public Date(int m, int d) {
        month = m;
        day = d;
    }

    public int getDay() {
        return day;
    }

    public int getMonth() {
        return month;
    }

    // returns the number of days
    // in the given month
    public int numDays(int m) {
        if (m == 2) {
            return 28;
        } else if (m == 4 ||
                   m == 6 ||
                   m == 9 ||
                   m == 11) {
            return 30;
        } else {
            return 31;
        }
    }

    // your method would go here
}
```