CSE 142
Sample Final Exam #1

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
`4.5 / 3 / 2 + 1`	_____
`57 % 10 / 3 + 1.25 * 4`	_____
`5 * 6 / 4 % 3 – 23 / (14 % 6)`	_____
`(6.7 > 25) && 9 > 2 && !(8.2 < 5)`	_____
`8 - 4 + "17" + 7.5 * 2`	_____

2. Array Mystery (10 points)

Consider the following method:

```
public static void mystery(int[] array) {
    for (int i = 0; i < array.length; i++) {
        array[i] = i * array[i];
    }
}
```

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`.

Array	Final Contents of Array
`int[] a1 = {};` `mystery(a1);`	_____
`int[] a2 = {7};` `mystery(a2);`	_____
`int[] a3 = {3, 2};` `mystery(a3);`	_____
`int[] a4 = {5, 4, 3};` `mystery(a4);`	_____
`int[] a5 = {2, 4, 6, 8};` `mystery(a5);`	_____

3. Inheritance Mystery (10 points)

Assume that the following classes have been defined:

```
public class A extends B {
    public void method2() {
        System.out.println("a 2");
    }
}

public class B extends C {
    public String toString() {
        return "b";
    }

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

public class C {
    public String toString() {
        return "c";
    }

    public void method1() {
        System.out.println("c 1");
    }

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

public class D extends B {
    public void method1() {
        System.out.println("d 1");
    }
}
```

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

```
C[] elements = {new A(), new B(), new C(), new D()};
for (int i = 0; i < elements.length; i++) {
    System.out.println(elements[i]);
    elements[i].method1();
    elements[i].method2();
    System.out.println();
}
```

4. File Processing (15 points)

Write a static method named `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 <> 2 25 4 wow!
```

your method should produce the following output:

```
fun.fun.fun.fun.fun.fun.
hellohellohello
<><><><><><><><><><>
2525
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 `"<>"`, the fourth line has 2 occurrences of `"25"` the fifth 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` (which itself could be numeric, such as `"25"` above). If the `Scanner` is empty (no integer/string pairs), your method should produce no output.

5. File Processing (10 points)

Write a static method named `reverseLines` that accepts a `Scanner` containing an input file as a parameter and that echoes the input file to `System.out` with each line of text reversed. For example, given the following input file:

```
If this method works properly,
the lines of text in this file
will be reversed.

Remember that some lines might be blank.
```

Your method should print the following output:

```
,ylreporp skrow dohtem siht fI
elif siht ni txet fo senil eht
.desrever eb lliw

.knalb eb thgim senil emos taht rebmemeR
```

Notice that some of the input lines can be blank lines.

6. Array Programming (15 points)

Write a static method `isAllEven` that takes an array of integers as a parameter and that returns a boolean value indicating whether or not all of the values are even numbers (true for yes, false for no). For example, if a variable called `list` stores the following values:

```
int[] list = {18, 0, 4, 204, 8, 4, 2, 18, 206, 1492, 42};
```

Then the call of `isAllEven(list)` should return `true` because each of these integers is an even number. If instead the list had stored these values:

```
int[] list = {2, 4, 6, 8, 10, 208, 16, 7, 92, 14};
```

Then the call should return `false` because, although most of these values are even, the value 7 is an odd number.

7. Array Programming (10 points)

Write a static method named `isUnique` that takes an array of integers as a parameter and that returns a boolean value indicating whether or not the values in the array are unique (`true` for yes, `false` for no). The values in the list are considered unique if there is no pair of values that are equal. For example, if a variable called `list` stores the following values:

```
int[] list = {3, 8, 12, 2, 9, 17, 43, -8, 46, 203, 14, 97, 10, 4};
```

Then the call of `isUnique(list)` should return `true` because there are no duplicated values in this list. If instead the list stored these values:

```
int[] list = {4, 7, 2, 3, 9, 12, -47, -19, 308, 3, 74};
```

Then the call should return `false` because the value 3 appears twice in this list. Notice that given this definition, a list of 0 or 1 elements would be considered unique.

8. Critters (15 points)

Write the `getMove` and `getColor` methods for the class `Bunny` that implements the `Critter` interface from Homework 8. Instances of the `Bunny` class should move and change color as follows: first pick a random direction and go three in that direction, then stay in the same place for three steps, then pick a random direction and go three in that direction, then stay in the same place for three steps, and so on, alternating between moving in a random direction and staying still. Whenever a bunny is moving, it should return `Color.YELLOW`. As soon as a bunny stops moving it should try to hide by blending in with its surroundings and returning the `Color.CYAN`. When picking a random direction, use the constants for directions defined in the `Critter` interface, namely `NORTH`, `SOUTH`, `EAST`, and `WEST`. Each direction should be equally likely. To stay in one place, a bunny should return `CENTER`.

You may add anything needed (fields, other methods) to implement `getMove` and `getColor` appropriately.

```
public class Bunny implements Critter {









    public int fight(String opponent) {
        return POUNCE;
    }

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

    // Please complete the getColor method
    public Color getColor() {







    }

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

















    }
}
```

9. Classes (10 points)

Consider the following `Date` class. Each `Date` object represents a calendar date such as September 19th.:

```
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;
    }

    public int numDays(int month) {
        if (month == 2) {
            return 28;    // ignore leap years
        } else if (month == 4 || month == 6 || month == 9 || month == 11) {
            return 30;
        } else {
            return 31;
        }
    }

    // your method would go here
}
```

Write a method named `compareTo` that will be placed inside the `Date` class. The `compareTo` method accepts another `Date` as a parameter and compares them to see which comes first in chronological order. It returns an integer with the following value:

- a negative integer (such as -1) if the date represented by this `Date` comes before that of the parameter
- 0 if the two `Date` objects represent the same month and day
- a positive integer (such as 1) if the date represented by this `Date` comes after that of the parameter

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

```
// client code
Date sep19 = new Date(9, 19);
Date dec15 = new Date(12, 15);
Date temp = new Date(9, 19);
Date sep11 = new Date(9, 11);
```

The following `boolean` expressions should all produce a `true` result.

```
sep19.compareTo(sep11) > 0
sep11.compareTo(sep19) < 0
temp.compareTo(sep19) == 0
dec15.compareTo(sep11) > 0
```

Solutions

1.

```
Expression                                  Value
---------------------------------------------------
4.5 / 3 / 2 + 1                             1.75
57 % 10 / 3 + 1.25 * 4                      7.0
5 * 6 / 4 % 3 - 23 / (14 % 6)               -10
(6.7 > 25) && 9 > 2 && !(8.2 < 5)           false
8 - 4 + "17" + 7.5 * 2                      "41715.0"
```

2.

```
Array                                   Value returned
-----------------------------------------------------
{}                                      {}
{7}                                     {0}
{3, 2}                                  {0, 2}
{5, 4, 3}                               {0, 4, 6}
{2, 4, 6, 8}                            {0, 4, 12, 24}
```

3.

```
b
c 1
a 2

b
c 1
b 2

c
c 1
c 2

b
d 1
b 2
```

4.

```
public static void printStrings(Scanner input) {
    while (input.hasNextInt()) {
        int times = input.nextInt();
        String word = input.next();
        for (int i = 0; i < times; i++) {
            System.out.print(word);
        }
        System.out.println();
    }
}
```

5.

```
public static void reverseLines(Scanner input) {
    while (input.hasNextLine()) {
        String text = input.nextLine();
        for (int i = text.length() - 1; i >= 0; i--) {
            System.out.print(text.charAt(i));
        }
        System.out.println();
    }
}
```

6.

```
public static boolean isAllEven(int[] list) {
    for (int i = 0; i < list.length; i++) {
        if (list[i] % 2 != 0) {
            return false;
        }
    }
    return true;
}
```

7.

```
public static boolean isUnique(int[] list) {
    for (int i = 0; i < list.length; i++) {
        for (int j = i + 1; j < list.length; j++) {
            if (list[i] == list[j]) {
                return false;
            }
        }
    }
    return true;
}
```

8.

```
public class Bunny implements Critter {
    private Random rand;
    private int steps;
    private int direction;
    private boolean hiding;

    public Bunny() {
        rand = new Random();
        hiding = false;
        direction = rand.nextInt(4);
    }

    ...

    public Color getColor() {
        if (hiding) {
            return Color.CYAN;
        } else {
            return Color.YELLOW;
        }
    }

    public int getMove(CritterInfo info) {
        // Pick a new direction and re-set the steps counter
        if (steps == 3) {
            steps = 0;
            hiding = !hiding;
            direction = rand.nextInt(4);
        }

        steps++;
        if (hiding) {
            return CENTER;
        } else if (direction == 0) {
            return NORTH;
        } else if (direction == 1) {
            return SOUTH;
        } else if (direction == 2) {
            return EAST;
        } else {  // direction == 3
            return WEST;
        }
    }
}
```

9. Two solutions are shown.

```
public int compareTo(Date other) {
    if (month < other.month || (month == other.month && day < other.day)) {
        return -1;
    } else if (month == other.month && day == other.day) {
        return 0;
    } else {
        return 1;
    }
}

public int compareTo(Date other) {
    if (month == other.month) {
        return day - other.day;
    } else {
        return month - other.month;
    }
}
```