# CS 112 - Exam 2 - Spring 2012

Name:

| | | |
|---|---|---|
| **Question 1** | **10** | |
| **Question 2** | **10** | |
| **Question 3** | **10** | |
| **Question 4** | **10** | |
| **Question 5** | **10** | |
| **Question 6** | **10** | |
| **Total** | **60** | |

The first two questions ask you to implement methods of a StringList class defined as in Lab 4 Question 1. As a reminder, the class was defined with the following data members.

```
public class StringList {

      //an array that maintains a list of String objects
      //the default initial size of the array is 10
      private String[] strings;

      //the number of valid String objects stored in the array
strings
      private int count;


}
```

1.For the `StringList` class, implement the following method `swap` that takes as input two indices and swaps the String objects stored at those indices. For the array
`strings -> [“a”, “b”, “c”, “d”, “e”, “f”, null, null, null, null]`
a call to `swap(1, 4)` would result in the array
`strings -> [“a”, “e”, “c”, “d”, “b”, “f”, null, null, null, null]`

```
//return false if x or y specifies an invalid location
public boolean swap(int x, int y);
```

2. For the `StringList` class, implement the following method `redundantItems`. The method takes as input an array of Strings and returns true if *any* of the Strings in the input array appear *more than once* in the `strings` data member. The method can be called as follows:

```
//example 1
StringList sl = new StringList();
sl.add("a");
sl.add("b");
sl.add("c");
String[] test = {"z", "a"};
System.out.println(sl.redundantItems(test)); //prints false

//example 2
StringList sl = new StringList();
sl.add("a");
sl.add("b");
sl.add("a");
String[] test = {"z", "a"};
System.out.println(sl.redundantItems(test)); //prints true


public boolean redundantItems(String[] input);
```

3. There are several problems with the following implementation of the `remove` method for the `StringList` class.

```
public boolean remove(String s) {

      int location = -1;

      for(int i = 0; i < this.count; i++) {
            if(this.strings[i].equals(s)) {
                  location = i;
                  break;
            }
      }
      if(location == -1) {
            return false;
      }

      for(int i = this.count-1; i > location; i--) {
            this.strings[i-1] = this.strings[i];
      }
      return true;
}
```

3(a) Given instance variables with the following values, show the resulting state of the array `strings` after a call to `remove` passing in the String `"c"`. Show your work to receive partial credit for an incorrect answer.

```
strings -> ["a", "b", "c", "d", "e", "f", null, null, null, null]
count -> 6
```

3(b) Describe the logic error in the method.

4. Consider the classes defined on the following page. What would be the output of the following code fragments? If a statement will cause an error, indicate that as well.

(a)

```
Pet p1 = new Pet("Brown", "Spot");
System.out.println(p1.getDescription());
System.out.println(p1.getColor());
System.out.println(p1.getBreed());
```

(b)

```
Dog d1 = new Dog("Black", "Fluffy", "Poodle");
System.out.println(d1.getDescription());
System.out.println(d1.getColor());
System.out.println(d1.getBreed());
```

(c)

```
Pet p2 = d1;
System.out.println(p2.getDescription());
System.out.println(p2.getColor());
System.out.println(p2.getBreed());
```

(d)

```
Dog d2 = (Dog)p2;
System.out.println(d2.getDescription());
System.out.println(d2.getColor());
System.out.println(d2.getBreed());
```

```
public abstract class Pet {

	protected String color;
	protected String name;

	public Pet(String color, String name) {
		this.color = color;
		this.name = name;
	}

	public String getDescription() {
		return "Name: " + name + " Color: " + color;
	}
	public String getColor() {
		return color;
	}
}

public class Dog extends Pet {

	protected String breed;

	public Dog(String color, String name, String breed) {
		super(color, name);
		this.breed = breed;
	}

	public String getDescription() {
		String desc = super.getDescription();
		desc += " Breed: " + breed;
		return desc;
	}

	public String getBreed() {
		return breed;
	}

	public void changeBreed(String breed) {
		this.breed = breed;
	}
}
```

5. What is the output of the following program? Show your work in order to receive partial credit for an incorrect answer.

```
public class Reverser {

	public void reverse(char[] values) {

		char[] tmp = new char[values.length];
		for(int i = values.length-1; i >= 0; i--) {
			tmp[values.length-1-i] = values[i];
			values[i] = 'X';
		}
	}

	public static void main(String[] args) {
		Reverser r = new Reverser();
		char[] word = {'c', 'h', 'a', 'i', 'r'};
		System.out.println(word);
		r.reverse(word);
		System.out.println(word);
	}
}
```

6. In a Sudoku program, a two-dimensional array of ints is used to represent the board. Recall that a valid Sudoku solution is one where each row, column, and 3x3 subsquare contains all of the numbers 1-9. For this question, implement the following method that will take as input a 9x9 array of ints and print the value stored in the upper left corner of each subsquare. Given the following array, your method would print `8, 7, 4, 1, 5, 2, 3, 6, 7`.

```
int[][] sample = {{8,6,1,7,2,9,4,5,3},
                  {4,9,2,3,6,5,0,8,8},
                  {5,7,3,1,4,8,6,2,9},
                  {1,4,9,5,8,3,2,6,7},
                  {2,3,7,9,1,6,8,4,5},
                  {6,5,8,2,7,4,3,9,1},
                  {3,1,5,6,9,2,7,8,4},
                  {9,2,4,8,3,7,5,0,6},
                  {7,8,6,4,5,1,9,3,2}};
```

```
public void printCorners(int[][] board);
```

Bonus: Did you submit the extra credit Sudoku exercise on March 8?