CSCI 162
Dr. Stephanie Schwartz
Review Questions for Exam 1

(answers to select problems)

3. Describe the purpose of the instance variables of a class

The instance variables allow us to implement information hiding (by making them private). The values of the instance variables capture the state of an object at any given point in time.

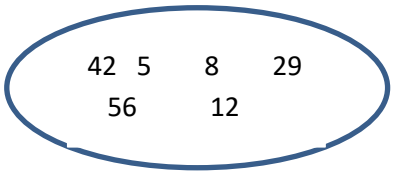
4. Arrays. Write a function called sumElements that will return as its integer return value the sum of all the elements in field (a two dimensional integer array which is the only parameter to the function; it already contains values and does not change).

```
public int sumElements(int[][] field) {
      int sum;
      for (int i=0; i<field.length; i++) {
            for (int j=0; j<field[i].length; j++) {
                  sum += field[i][j];
            }
      }
      return sum;
}
```

6. Simple Container Classes

a. Given a bag, describe its contents (abstract value) after the series of operations:

insert 42, insert 5, insert 56, insert 12, insert 29, insert 8



size: 6

b. Given a sequence, describe its contents (abstract value) after the series of operations:

addBefore 42, addAfter 5, addAfter 56, addBefore 12, addBefore 29, addBefore 8

42 5 ***8*** 29 12 56

Size: 6

c. How do the bag and sequence differ even though they contain the same values?

The order of the items is important in sequence, as is the concept of a current element

d. Draw a picture of the internal contents (instance variables) of the sequence where the series was applied.

Data:

42	5	8	29	12	56	?	?	?	?

manyItems 6
current 2

Be complete with your picture.

e. Describe the representation of the sequence as we implemented it with a partially-filled array.

Elements are in order and are stored from data[0] through data[manyItems-1]. Current keeps track of index of current element.

7. Classes (Hint: refer to the program on the back page and questions 8 to 10) This is a long description, but the problem isn't hard. You're building a game in which the players each set the goal they must meet to win. As players move through the game, they must collect 10 objects from three categories: purple, green, and navy. Before the game begins, each player records how many objects they must collect in each category to win the game. The total of objects a player must collect must total to 10, and no category may have a negative number. For example, Sandy might choose to try to get 4 purple, 2 green, and 4 navy, while Terry might choose 0 purple, 7 green, and 3 navy.

Your task is to build a class to support goal setting and recording. The Goal class has six integer instance variables to record the targets in each category (use pT, gT, and nT for the targets) and the number of objects accumulated in each category (p, g, and n). The constructor will take in the three target values for the categories as its parameters (pT0, gT0, and nT0). It will also initialize the accumulated objects counters in each category. Throw an IllegalArgumentException if the targets are not each at least zero or don't sum to ten.

The class also has methods that update the accumulated object counts. For the test, we'll have only addPurple which adds 1 to the purple count (p). The class also has a boolean hasReachedGoal method that returns true if the goal has been reached and false otherwise. The goal is reached only when all of the targets have been met (they may be exceeded).

Write the Goal class including the constructor, the two methods described above (addPurple and hasReachedGoal), and the six instance variables (pT, gT, and nT and p, g, and n)

```
public class Goal {

	private int pT, gT, nT, p, g, n;

	public Goal(int pTO, int gTO, int nT0) {
		if (pTO < 0 || gTO < 0 || nTO < 0 || (pTO + gTO + nTO != 10))
			throw new IllegalArgumentException("Goals must be greater
				than 0 and total 10");
		pT = pTO;
		gT = gTO;
		nT = nTO;
	}

	public void addPurple() {
		p++;
	}

	public boolean hasReachedGoal() {
		return (p >= pT && g >= gT && n > nT);
	}
```

8. Parts of Functions and Classes

a. What is the purpose of the new in Record papers = new Record( ) ?

Allocates a new instance of Record class. Calls constructor.

b. Why is the throw statement used in the method report?

To protect from illegal divide by 0 operation

c. How many instances of the Game class are created when this program runs?

none

d. In addGame, what would happen if you incremented us at the end of the method?

Nothing – us is an integer parameter. Incrementing this at the end of the method will have no effect because the it is a simply a copy of the value passed in from the calling method

e. Could print add one to the value of games as in r.games++? Why or why not?

No, games is a private instance variable of Record.

9. What does the program on the next page print? Show the steps of execution for partial credit. It does compile and execute cleanly.

Papers 100%
Scissors 0%
Papers 20%
Scissors 60%

10. Expanding the Class

a. Write a concise and precise class invariant for the games instance variable in terms of the other instance variables.
games will be the sum of wins, losses and ties.

b. Write the prototype (method header) for a numWins method that returns the number of wins as its integer return value. It has no parameters. Write prototype only.

public int numWins()

c. You want to print a win-loss-tie record (such as 3-1-1) in Game.java. Describe in precise words (not code) the best way to go about doing this. What would change where?

Two possibilities:

1) Create accessor methods for numWins, numLosses, numTies on Record class. Call these from Game.java
2) Create a String method on Record class that returns formatted record. Call this from Game.java

```
////////////////////////////////////////////////
// Game.java
////////////////////////////////////////////////

public class Game {
    public static void main(String[] args) {
        Record papers = new Record( );
        Record scissors = new Record( );
        papers.addGame(4, 3);
        scissors.addGame(0, 4);
        print("Papers", papers);
        print("Scissors", scissors);
        papers.addGame(5, 7);
        scissors.addGame(4, 6);
        scissors.addGame(9, 1);
        papers.addGame(3, 7);
        scissors.addGame(7, 3);
        papers.addGame(0, 5);
        papers.addGame(1, 4);
        scissors.addGame(3, 1);
        print("Papers", papers);
        print("Scissors", scissors);
    }

  public static void print(String name, Record r)
  {
    System.out.println(name + " "
    + r.report( ) + "%");
    }
}
```

```
////////////////////////////////////////
// Record.java
////////////////////////////////////////
public class Record {
    private int wins;
    private int losses;
    private int ties;
    private int games;

public Record( ) {
    wins = 0;
    losses = 0;
    ties = 0;
    games = 0;
}

public void addGame (int us, int them) {
    if (us > them) {
        wins++;
    } else if (us < them) {
        losses++;
    } else {
        ties++;
    }
    games++;
}

public int report ( ) {
    if (games > 0) {
        return (wins * 100) / games;
    } else {
        throw new ArithmeticException
          ("divide by zero");
    }
}
}
```

11. Write method "clone" for a GroceryCart class that has an int numItems and an array of doubles called prices.

```
public GroceryCart clone() {
   GroceryCart answer;
   try {
         answer = (GroceryCart)super.clone();
   }
   catch (CloneNotSupportedException e) {
         throw new RuntimeException("clone not supported");
   }
   answer.prices = prices.clone();
   return answer;
}
```

12. Using the GroceryCart class described in the previous question, write a **static** "union" method that takes two GroceryCart objects "c1" and "c2" and returns a new GroceryCart object representing the union of the two GroceryCarts. You can assume a constructor that takes an initial number of items already exists for GroceryCart.

```
public static GroceryCart union(GroceryCart c1, GroceryCart c2) {
   if (c1 == null || c2 == null)
         throw new IllegalArgumentException("Null parameter");
   GroceryCart c3 = new GroceryCart(c1.numItems + c2.numItems);
   System.arraycopy(c1.prices, 0, c3.prices, 0, c1.numItems);
   System.arraycopy(c2.prices, 0, c3.prices, c1.numItems, c2.numItems);
   c3.numItems = c1.numItems + c2.numItems;
   return c3;
}
```