Test 2: CompSci 6

75 Minute Exam

March 4, 2009

Name (print): _____

Honor Acknowledgment (signature): _____

DO NOT SPEND MORE THAN 10 MINUTES ON ANY OF THE QUESTIONS! If you do not see the solution to a problem right away, move on to another problem and come back to it later.

Before starting, make sure your test contains 13 pages. The last page is blank and can be used as scratch paper, but **all pages must be turned in**.

None of the Java programs or program segments should have syntax errors unless stated. If you think there is a syntax error or you do not understand what a question is asking, then please ask. Import statements may not always be shown.

Do **not** discuss this test **with anyone** until the test is handed back.

	value	grade
Problem 1	16 points	
Problem 2	24 points	
Problem 3	20 points	
TOTAL:	50 points	

PROBLEM 1 : (*Cutting through the confusion:* (16 points))

Many web pages embed parameters to their web pages within the page's URL (Uniform Resource Locator). This is often done with scripted pages such as you would find at online stores. This information appears after the web page's name and is encoded such that each piece is separated from other pieces by an ampersand, &. A piece is made up of an id and a value which are separated by an equal sign, =.

Several examples are given below. The first line below has three pieces. The first piece has the id *name* and the value *harry*, the middle piece has the id *type* and the value *wizard*, and the last (or third) piece has the id *diet* and the value *potions*.

```
"name=harry&type=wizard&diet=potions"
"name=ethan&year=1999&month=june&color=pink",
"name=robert&animal=tiger",
"name=penny&type=penguin&zoo=asheboro",
"name=denny&type=duke&weight=120",
```

Part A: (5 points)

Write a method `pieceCount` that returns the number of pieces in a URL.

For example the code below sets x to 1 and y to 5.

```
int x = pieceCount("name=marco");
int y = pieceCount("name=moe&type=zebra&weight=360&height=71&zoo=bronx");
```

Complete the method, `pieceCount`, below.

```
/**
 * Return the number of pieces (as defined above) in url
 */
public int pieceCount (String url)
{









}
```

Part B: (5 points)

Write a method `getNthPiece` that returns just the specified piece from the given URL. For example, the code below sets first to "name=moe" and last to "zoo=bronx"

```
string zoostuff = "name=moe&type=zebra&weight=360&height=71&zoo=bronx";
string first = getNthPiece(zoostuff, 1);
string last = getNthPiece(zoostuff, 5);
```

Complete the method, `getNthPiece`, below.

```
/**
 * Return the n-th piece of url (where n is guaranteed to be
 * between 1 and pieceCount(url), inclusive).
 */
public String getNthPiece (String url, int n)
{









}
```

Part C: (6 points)

Write a method `splitURL` that splits a given URL into two parallel lists, one containing the ids and one the values.

For example, the following URL would fill the list *ids* with the strings "name", "year", "month", and "color" and the list *values* with the strings "ethan", "1999", "june", and "pink" respectively. In this way, the second element of the list *ids* corresponds to the second element of the list *values*

```
"name=ethan&year=1999&month=june&color=pink"
```

You will not receive full credit for this part unless you call the methods *pieceCount* and *getNthPiece* that you wrote previously and use their results in determining the result of this function. Assume each works as specified regardless of what you wrote.

Complete the method, `splitURL`, below.

```
/**
 * Splits each piece of given url such that ids contains all
 * of the id fields in url and  values contains all of the
 * value fields in url.  You may assume that both lists are
 * non-null and simply add the pieces to the end of each list.
 */
public void splitURL (String url, List<String> ids, List<String> values)
{









}
```

PROBLEM 2 : (*Above average?:* (24 pts))

Consider the following super-class declaration used for this problem.

```
/**
 * A Shape represents a generic geometric shape that fits within a rectangle.
 */
public class Shape
{
    private double myWidth, myHeight;  // dimensions

    public Shape (double width, double height) {
        myWidth = width;
        myHeight = height;
    }

    // returns specific dimension asked for
    protected double getWidth () {
        return myWidth;
    }

    protected double getHeight () {
        return myHeight;
    }

    // returns area of the geometric shape
    public double area () {
        return 0;
    }
}
```

Part A: (6 points)

Briefly explain the meaning of Java's visibility keywords: `public`, `protected`, and `private`? Use examples from the `Shape` class to illustrate your explanantions.

Part B: (6 points)

The class `Circle` below extends `Shape` and overrides the `area` method to return the area for only a circle shape. However, given the implementation below, there are two problems with the `area` method as written: one is a compilation error and the other a logic error.

```
/**
 * A Circle represents a geometric circle.
 */
public class Circle extends Shape
{
    private double myWidth;

    public Circle (double diameter)
    {
        super(diameter, diameter);
    }

    // returns area of circle
    public double area ()
    {
        return Math.PI * (myWidth / 2) * (myHeight / 2);
    }
}
```

Below, explain the reason for each error and then modify the code above to fix each one.

Part C: (6 points)

Write the class `Rectangle` that extends `Shape` and overrides the `area` method to return its $width * height$. You should provide only those constructors and methods that are required in order for the class to compile.

Part D: (6 points)

Complete the method, `averageArea`, that returns the average area of a collection of objects that all extend `Shape`. If the given collection is empty, i.e., its `size` is 0, then the method should return 0.

Complete the method `averageArea` below:

```
/**
 * @return the average of the areas of all shapes in the list shapes,
 *         or 0 if the list is empty
 */
public double averageArea (ArrayList shapes)
{









}
```

PROBLEM 3 : (*Working Together (20 pts)*)

Assume that every player on a team is modelled using the class `Player` declared as shown below.

```
public class Player
{
    private String myName;
    private ArrayList<Integer> myMinutes; // minutes played in each game

    public Player (String name, ArrayList<Integer> minutes)
    {
        myName = name;
        myMinutes = minutes;
    }

    public String getName ()
    {
        return myName;
    }

    public int getMinutesForGame (int whichGame)
    {
        // assume 0 <= whichGame < myMinutes.size()
        return myMinutes.get(whichGame).intValue();
    }

    public int getNumberGamesPlayed ()
    {
        return myMinutes.size();
    }
}
```

Part A (10 points)

Write the function `readTeam` below that reads data from a file and stores it in an initially empty list of `Player`'s. Assume that the text file with team information being read is in the format shown below. There are two lines for each player in the team. The first line contains first and last name, and the second line contains a list of the number of minutes played in each game during the season. Note that the number of games played varies from player to player.

A sample data file is shown below with the first player with 4 games and the second play with 7 games:

```
Fred Smith
1 13 5 1
Chris Jones
8 2 5 6 4 7 16
```

Complete `readTeam` below.

```
/**
 * Reads all player information from the file represented by
 * the parameter Scanner and returns it in a list
 */
public ArrayList<Player> readTeam (Scanner input)
{









}
```

Part B (4 points)

Write the function `totalNumberOfMinutes` below that, given a `Player`, returns the total number of minutes that player played during the season.

Given the example players in the previous part, your function should return the value `20` minutes when called with the `Player` object representing Fred Smith and `48` minutes when called with the `Player` object representing Chris Jones.

Complete `totalNumberOfMinutes` below. Note, this method is not part of the `Player` class and so does not have access to its private instance variables.

```
/**
 * Return the sum of all the minutes player played.
 */
public int totalNumberOfMinutes (Player player)
{









}
```

Part C (6 points)

Write the function `mostMinutes` below that, given a list of players representing a team, returns the name of the `Player` who has played the most total minutes during the season.

Given the example team from the previous parts, your function should return the name "Chris Jones" because he played for 48 total minutes, while his only other team member played for only 20 minutes.

You will not receive full credit for this part unless you call the function `totalNumberOfMinutes` that you wrote in Part B at least once and use its result in determining the result of this function. Assume `totalNumberOfMinutes` works as specified regardless of what you wrote in Part B.

Complete `mostMinutes` below.

```
/**
 * Return the player that has played the most minutes regardless
 * of the number of games played.
 */
public String mostMinutes (ArrayList<Player> team)
{









}
```

Throughout this test, assume that the following classes and methods are available. These classes are taken directly from the material used in class. There should be no methods you have never seen before here.

Point

```
public class Point {
    // coordinates
    public int x;
    public int y;

    // Constructs and initializes a point at the
    // specified (x,y) location.
    public Point (int x, int y)

    // Returns distance from this point to given one
    public int distance (Point other)

    // Adds dx and dy to this point's coordinates
    public void translate (int dx, int dy)
}
```

Math

```
public class Math {
  // Returns the greater of the two given values
  public int max (int a, int b)
  // Returns the lesser of the two given values
  public int min (int a, int b)
}
```

ArrayList

```
public class ArrayList<Item> {
    // Constructs an empty list
    public ArrayList ()

    // Returns the number of items in this list.
    public int size ()

    // Searches for the first occurence of the given
    // item, returns  -1 if not found
    public int indexOf (Item i)

    // Returns item at index in this list.
    public Object get (int index)

    // Appends given item to end of this list
    public boolean add (Item i)
}


// To create an ArrayList of String objects:
ArrayList<String> things = new ArrayList<String>();
```

String

```
public class String {
    // Returns string's length
    int length ()

    // Returns true if the given string contains
    // the exact same characters as this string
    public boolean equals (String other)

    // Returns true if the given string contains
    // the same characters as this string regardless
    // of whether they are upper- or lower-case
    public boolean equalsIgnoreCase (String other)

    // Returns a substring of this string that
    // begins at the given beginIndex and extends
    // to the character at index endIndex - 1
    String substring (int beginIndex, int endIndex)

    // Returns a substring of this string that
    // begins at the given beginIndex and extends
    // to the end of the string
    String substring (int beginIndex)

    // Returns position of the first occurrence
    // of str, -1 if not found
    int indexOf (String str)

    // Returns position of the first occurrence
    // of str after index start, -1 if not found
    int indexOf (String str, int start)

    // returns true if this string starts with
    // the characters in str
    boolean startsWith (String str)

    // returns true if this string ends with
    // the characters in str
    boolean endsWith (String str)
}
```

Random

```
public class Random {
    // Creates a new random number generator.
    public Random ()

    // Returns pseudorandom, uniformly distributed,
    // int value between 0 (inclusive) and the
    // specified value (exclusive)
    public int nextInt (int n)
}
```

Scanner

```
public class Scanner {
    // Create Scanner for that reads data from a string.
```

```
    public Scanner (String str)

    // Use the given pattern to delimit items when scanning
    public void useDelimiter (String pattern)

    // Check if more items are available
    public boolean hasNext ()

    // Get next delimited item as a string
    public String next ()

    // Get next delimited item as an integer value
    public int nextInt ()
}

// typically used in a while loop:
Scanner input = new Scanner("a b c");
while (input.hasNext())
{
    String letter = input.next();
    // do something with letter
}
```