CSE1030 Lab 01

Thursday, July 3, 2014
Due: Monday, July 7, 2014, before 23:59

Introduction

Recall that a utility class is a class containing only `public static` constant attributes (fields) and methods. The purpose of a utility class is to group together related constants and methods so that a client can access these features from a single class.

In this lab, you will create a utility class for a tic-tac-toe game.

The goals of this lab include:

- reviewing the style rules that your code is expected to adhere to
- implementing the utility class `TicTacToeUtil`; this will require a good understanding of Java strings, lists, and sets.
- learning how to perform unit tests on your code

Style Rules

The style rules are not overly restrictive in CSE1030.

1. Your programs should use the normal Java conventions (class names begin with an uppercase letter, variable names begin with a lowercase letter, etc.).

2. In general, use descriptive variable names. There are exceptions to this rule; for example, traditional loop variables are often called i, j, k, etc.

3. Use a consistent indentation size. Beware of the TAB vs SPACE problem: Tabs have no fixed size; one editor might interpret a tab to be 4 spaces and another might use 8 spaces. If you mix tabs and spaces, you will have indenting errors when your code is viewed in different editors.

4. Use a consistent brace style:

```
// left aligned braces

class X
{
  public void someMethod()
  {
    // ...
  }

  public void anotherMethod()
  {
```

```
    for (int i = 0; i < 1; i++)
    {
      // ...
    }
  }
}
```

or

```
// ragged braces

class X {
  public void someMethod() {
    // ...
  }

  public void anotherMethod() {
    for (int i = 0; i < 1; i++) {
      // ...
    }
  }
}
```

5. This one always causes problems for students: Insert a space around operators (except the period ".").

The following is

```
    // some code somewhere
    boolean isBetween = (x > MIN_VALUE) && (x > MAX_VALUE);
    int someValue = x + y * z;
```

much easier to read than this

```
    // AVOID DOING THIS

    // some code somewhere
    boolean isBetween=(x>MIN_VALUE)&&(x>MAX_VALUE);
    int someValue=x+y*z;
```

A utility class for Tic-tac-toe

Utility classes are used to group a set of constants and methods that are used in related tasks. For example, `java.lang.Math` groups together mathematical constants and methods, `java.util.Arrays` groups together methods that operate on arrays, and `java.util.Collections` groups together methods that operate on collections.

Question: Implement the utility class `TicTacToeUtil`

In CSE1030, you are typically required to implement an API that has been designed by someone else. Typically, to implement an API you must complete the following steps:

- create one or more packages; **the package names must exactly match the package names given in the API**
- in each package, create one or more required class; **the class names must exactly match the class names given in the API**
- in each class provide all of the `public` attributes (fields) detailed in the API; **the modifiers, names, and types of the attributes must exactly match what is specified in the API**
- in each class provide all of the `public` methods detailed in the API; **the modifiers, return types, method names, and list of parameters must exactly match what is specified in the API**

Getting started

To get started, you should do the following in eclipse:

1. Create a new Java Project (perhaps called `lab1`)
2. In your project, create a new package named `cse1030.games.tictactoe`
3. In the package `cse1030.games.tictactoe` create a new Java class named TicTacToeUtil
4. Complete the class TicTacToeUtil so that it implements the API:

   Fields:

   ```
   1. static int NUMBER_OF_SPACES //# of cells in the game (usually 9)
   ```

   Methods:

   Methods 1–3 return the number of rows, columns and diagonals respectively with consist of symbols of one kind (0 or X). Possible return values are 0, 1, and 2 (for 3 or more you need more than 5 symbols of the same kind).

   Methods 4 and 5 return true if only naughts or only crosses have horizontal, vertical, or diagonal rows consisting of only naught symbols, or only crosse symbols, respectively.

   Let the game be a draw (Method 6) when either 1) neither naughts, nor crosses have complete rows; or 2) both have complete rows – horizontal, vertical, or diagonal rows consisting of symbols of one corresponding kind.

   ```
   1. public static int horRows (String s);
   2. public static int vertRows (String s);
   3. public static int diagRows (String s);
   ```

4. `public static Boolean naughtsWin (String s);`
5. `public static Boolean crossesWin (String s);`
6. `public static Boolean isDraw (String s);`
7. `public static void main` (see next Step)

   NOTE: The state of the board is represented with strings like "XXX 0X0 000", which corresponds to the following board configuration (in the string above the three groups of spaces are separated with spaces):

   ```
   XXX
   0X0
   000
   ```

   Configuration that there could be not more than 5 symbols of the same kind and that in this case for simplicity having multiple rows consisting of identical symbols each (e.g., top and bottom rows above).

5. Your main method should compute the odds of winning for naughts and the crosses (**assuming one always starts with naughts**). To do so, you will simulate filling the board with naughts and crosses randomly 1,000,000 times and count the number of times that each of the cases above appear (number of horizontal|vertical|diagonal rows, cross wins, naught wins, draws).

   The odds in percent of can be computed using the formula

   ```
   100.0 * n / 1000000
   ```

   where `n` is the number of times the case appeared in the 1,000,000 games. Your method should print the odds of getting each case as shown below:

```
Horizontal rows: x
Vertical rows: x
Diagonal rows: x
Naughts win: x.xxxx %
Crosses win: x.xxxx %
Draw: x.xxxx %
```

Submit

Submit your solution using the `submit` command. Remember that you first need to find your workspace directory, then you need to find your project directory. In your project directory, your files will be located in the directory `src/cse1030/games`

```
submit 1030 L1 TicTacToeUtil.java
```

Alternatively, you may use the web form at
`https://webapp.eecs.yorku.ca/submit/index.php`

Some things to think about

- Do you get the same odds every time you run your program?
- What happens to the odds if you reduce the number of games (to say 1,000)? Do the odds change more or less than your original program when you run your program several times?
- What happens to the odds if you increase the number of rolls (to say 1,000,000,000)? Do the odds change more or less than your original program when you run your program several times?
- How can you test your methods for correctness?