CS1011: Introduction to Programming:
Practical 4: Teaching Week 5

Instructions:

BlueJ is available on the ITS machines in the laboratories.

Level 1: Modelling a Class

Develop on paper a class diagram that represents the problem below. Your model should consider the following points:

What data do you want to store?
What operations do you want to perform?

HiLo Game In this game, the player has to guess a number. For each guess, they are told whether the number is higher or lower than their guess. The number of guesses may be limited, and the range of numbers to guess from could be altered. While the model should lead to a representation of the game that a user can play, you should also consider including a computer player. Think about the different styles of computer player, you could have a good player, or a poor player.

You should propose the following in your model of your game, making sure you understand what each of the parts within the declarations and signatures mean:

Field declarations
Method signatures
Constructor signature(s)

Level 2: Implementing Your Model

Now try to implement the class that you modelled in above. You should do this in a new BlueJ project, so on the BlueJ menu, select *Project -> New Project* and save it in your file space.

Create a new class by selecting *New Class* on the left hand side. This will give you a choice of different types of class types that can be implemented in Java, but for the time being we are just concerned with the first type. Give your class a name and click *Ok*.

When you open the editor for the new class you will see a lot of default code that BlueJ automatically includes to illustrate a field, constructor and method, along with the matching *JavaDoc* - comments describing what the class and methods do. While you may want to use these as a guide to get started, the best option is to clear out the contents of the class to start with a clean sheet.

Writing the class: Add in the class declaration with the curly brackets that will contain all the fields and methods for the class Add in the fields with their data types that you defined in your model. Pick the simplest method/constructor first to gradually build up the functionality in your class

Hint: To produce a random number, the following expression can be added into a method in your class:

```
int randomNumber = (int)(Math.random() * upperLimit);
```

This expression will give you a random number from 0 to upperLimit - 1. If you want the range to be from 1 to upperLimit, simply +1 onto the end of the expression.

Note: The method `Math.random()` returns a value of type double between 0 and 1. This is multiplied by the range we want to select from, and then **cast** into the int data type using `(int)`. This removes any numbers after the decimal place so will always round down.

Remember, when comparing two Strings use the boolean expression:

```
string1.equals(string2);
```

Level 3: Extending Your Implementation

Add in a GameManager class that records outcomes from the game you implemented above and can play repeated games (using the computer player). The two coins project used earlier in the course may help here. Think about the different statistics you can produce, e.g. average number of guesses in HiLo (can you beat the computer).

For your implementation, draw an object diagram that depicts the object structure that is created when part way through a game.

While we have not yet covered looping over code in the lectures, you may find it useful to be able to repeat segments of code using a loop. An example of a for loop that can be included in a method is shown below. Before using it, make sure you understand each part:

```
for(int i=0; i < upperLimit; i++) {
// body of for loop
}
```