PAL Consolidation 14 *University of Hertfordshire – Week 16 – March 2016 –*

Aim & Objectives

- Discuss previous session
- Look forward to next session

Introduction

Using the sections previously used and some new ideas, we will design a simple game that users can play over the terminal. We will talk about how the game is played, the architecture and how it will be implemented.

The Game

This will be a two player game of guessing, where two random numbers, A & B are given to one of the players (named the keeper) and the other player (named the guesser) must guess which is greater than the other. Before making the guess, the guessing player may say one number and find out whether A or B is higher or lower than that number. To make things easy, we'll just say they can find out information about A.

A typical game would look like this:

- The '`keeper`' generated two random numbers that are hidden to the guesser, where '`A = 7`' and '`B = 3`'.
- The '`guesser`' may then choose a random number, in this case '`K = 2`'.
- The '`keeper`' then answers the question of whether '`A > K`'. In this case, we ask whether '`7 > 2`', which is '`true`'.
- Using this information, the '`guesser`' can then answer the question whether '`A > B`'. In this case the '`guesser`' agrees with the statement that '`A > B`'.
- '`keeper`' checks this, '`A > B == 7 > 3`', which is in fact true.
- This result is given as a win to '`guesser`' for correctly guessing.
- A winner is decided after '`N`' games by who has the most correct guesses. If truly random, the score would be 50% wins each.

Exercise - Try it out

Get some paper and pen or a text editor on the computer to record the score. Play the game with somebody sitting close to you. You'll need to randomly produce a number within a given range. The following URL will produce two very random numbers between 1 and 100:

'*https://www.random.org/integers/?num=2&min=1&max=100&col=2&base=10&format=html&rnd=new*'

NOTE: You'll need to make sure that these numbers remain hidden with your partner.

Exercise - Architecture

Now we will look at designing our software architecture. We'll want something that will allow two humans, a human and a computer or a computer and computer to play. Ideally the human and computer should be swappable. Consider making them as classes with a interface class, for example:

```
public interface Player{
   /**
      * setNumbers()
      *
      * This method allows a player to
      * receive the current randomly
      * generated numbers.
      *
      * @param a The first number.
      * @param b The second number.
   **/
   public void setNumbers(int a, int b);

   /**
      * getNumber()
      *
      * Gets the number to compare with
      * A.
      *
      * @return K to compare with A.
   **/
   public int getNumber();

   /**
      * getGuess()
      *
      * Gets guess for A greater than
      * B.
      *
      * @return True or false for A
      * greater than B.
   **/
   public boolean getGuess();
}
```

You are then able to implement these methods with the following code:

```
public class Computer implements Player{
   public void setNumbers(int a, int b){
      /* Write code here */
   }

   public int getNumber(){
      /* Write code here */
   }

   public boolean getGuess(){
      /* Write code here */
   }
}
```

You might not have seen this before, but the idea is simple. One class defines how other implementing classes are defined - the others just write the code for those methods. What this means is we can do the following:

```
/* Initialise the players */
Player p1 = new Computer();
Player p2 = new Human();
/* Get the K number from player 1 */
int k = p1.getNumber();
```

We don't have to worry about which one we have! We treat them both the same. We can't on the other hand, have:

```
Player p3 = new Player();
```

The reason this throws an error is because it doesn't have any code, it simply doesn't make sense. It's just describing the methods, not how those methods will be implemented.

Don't worry for now how this will work, it's enough to know they exist and we can use them to make our code more understandable.

TASK: Roughly plan out how you would write this code. You need to think about:

- Turns, how will you know who's turn is next?
- Who are the players?
- How will the game play out?
- How will it be displayed?

Exercise - Implement

Now it's time to implement the code, you'll need to research the following on your favourite web browser:

- How will random numbers be produced between 1 and 100?
- How will I get input from the users?

TASK: Implement the code.

Exercise - Advanced

So, you think you're a wise crack eh? Okay, get your head around this then:

'*https://www.youtube.com/watch?v=ud_frfkt1t0*'

If you need help understanding this, please ask. Essentially we can improve our chances of winning above 50% by producing a random number, hence the part that didn't make all that much sense at first.

TASK: Implement this as a smart AI to play the game.

Resources & Further Reading

'*http://homepages.herts.ac.uk/~db12aba/*' – All content from these sessions updated weekly.

'*http://code.org/*' – A good resource testing your programming skills.

'*http://stackoverflow.com/*' – Highly recommended online help for programmers (NOTE: Employers are interested to know whether you're an active member of this site!).

'*http://draw.io*' – A very good, free online drawing tool that exports to many formats, including '`XML`' and '`JPG`'.