COMP 30070 Practical Exam 2014

Attempt parts 1-4 in sequence. **Submit only one Ruby program finally**. Aim to get each part working correctly before moving on to the next. The bulk of the marks will be awarded for how much you get working, so be sure to submit a working solution, commenting out sections of code if necessary. Marks are also awarded for clear design and coding, indentation and (light) commenting. You may write your program in one .rb file if you wish (the simplest approach), or use several .rb files.

Test cases: It suffices to fully test the `GuessIt` class.

Turn off your wireless network and bluetooth before the exam commences.

1. The *GuessIt* game works as follows. An oracle chooses a secret integer number randomly in some range (say 1..100). The role of the player is then to try to guess what the number is. When a player makes a guess, the oracle responds by informing the player if they are correct or not. If they are not correct, the game continues, otherwise it ends.

   Implement the `Oracle` class. It comprises a suitable initializer, and an `is_it?` method that informs the caller if the argument is the same as the secret number.

   Implement the `RandomPlayer` class. It has a `guess` method that makes a random guess at the secret number, in the same range as was used by the oracle, and asks the oracle if it is correct using the `is_it?` method. The `RandomPlayer` initializer takes as arguments the player name, a reference to the oracle and the range in which the secret number lies.

   Create also a `GuessIt` class that is responsible for setting up and starting the game. It will create the oracle and a player, and repeatedly invoke `guess` on the player until it guesses correctly.

   Add `puts` statements to the `RandomPlayer` class so that the program produces an output like this (for a player called ‘John’):

   ```
   John guessed 45
   John guessed 57
   …
   John guessed 83 and won!
   ```

   (40 marks)

2. Extend the `is_it?` method in `Oracle` so that it informs the caller if the argument is greater than, less than or equal to the secret number. The existing `RandomPlayer` class should work as before. Create a new type of player, `SmartPlayer`, that is just like a `RandomPlayer` but whose `guess` method

uses the enhanced output of the `is_it?` method to perform a binary search[1] for the secret number.

Add `puts` statements to the `SmartPlayer` class so that the program produces an output like this (for a player called ‘Aoife’):

```
Range is: 1..100
Aoife guessed 50
Range is: 1..49
Aoife guessed 25
Range is: 1..24
…
Aoife guessed 15 and won!
```

Extend the `GuessIt` class so that it creates a `RandomPlayer`, invokes `guess` on it repeatedly until it guesses correctly, creates a `SmartPlayer`, invokes `guess` on it repeatedly until it guesses correctly.

(25 marks)

3. The `SmartPlayer` class and the `RandomPlayer` class have an amount in common. Create a new superclass for these classes called `Player` and move the common parts of `SmartPlayer` and `RandomPlayer` to this superclass. Your program should produce the same as it did in Part (2).

   (15 marks)

4. Add a new player class called `SequentialPlayer` whose guessing strategy is to start at the lowest number in the range and move one-by-one to the highest. Extend the player classes to keep track of the number of guesses the player has made so far. Update the `GuessIt` class so that it creates an array of length 3, initialised with one of each type of player object. It iterates over this array invoking `guess` on each object in turn until all the players have guessed correctly. Finally the players are printed in descending order of number of guesses made.

   (20 marks)

Submission

Create a zip file called <student_id>.zip that includes the following:

1. All the .rb files you wrote, including one called `main.rb` that runs the main script for the program and a file that holds the test script for the `GuessIt` class.
2. A file called `statement.txt` that contains your name, student ID and a statement of what you achieved, e.g., “parts 1-3 completed correctly, part 4 attempted but unfinished.

[1] For example, if guessing in the range (1..19) it guesses the midpoint of the range, in this case 10. Assuming 10 isn’t correct, the `is_it?` method will inform the player if the secret number is greater or less than 10. The next range to guess is in either (1..9) or (11..19).