COMP 202 Winter 2005 Assignment #3

On conditionals, loops and objects

Distributed: Feb 9th, Due: Mar 1st

1 The game of Nim

In this assignment you are asked to develop a game of Nim between human and computer. The game of Nim is a simple game with many variants. Here we will consider the following variant.

The game of Nim is played between two players. The two players alternatively take marbles from a pile. In each turn, a player decides how many marbles to take. A player must take at least one marble, but at most half of the marbles currently in the pile. Then the other player takes a turn. The player who takes the last marble loses.

Write a program in which the computer plays against a human opponent. The initial size of the pile is a random number between 10 and 100 (but your program must work with *any* initial pile size greater than 2.) A coin is tossed to decide who takes the first turn. The computer decides randomly what strategy to take. It can take two possible strategies: *stupid* or *smart*. In stupid mode, the computer simply chooses a random number of marbles to take (as long as that number is legal.) In smart mode, the computer takes off enough marbles to make the size of the pile a power of two minus 1 – that is, 1, 3, 7, 15, 31, 63, ... etc. That is always a legal move (why?) except if the size of the pile is currently one less than a power of 2 (why?) In that case, the computer makes a random legal move.

Note that the computer cannot be beaten in smart mode when it has the first move, unless the pile size happens to be 15, 31, 63, ... (why?) Of course, a human player who has the first move and knows the computer strategy can win against the computer.

You cannot assume that the human enters a legal move.

When a game finishes, the program must inform who won, and reveal the strategy used by the computer (regardless of who won.) The program must keep track of how many games were won by the human, and how many games were won by the computer using either strategy. Furthermore, it must print the ratio of games won by the stupid strategy with respect to the total computer wins, and the ratio of games won using the smart strategy with respect to total computer wins. All this information must be printed after each game.

Once a game is finished, the user is asked whether (s)he would like to play again.

A sample run is shown after part 2.

2 Object-Oriented Nim

The second part of this assignment is to transform your program into an Object-Oriented program. The idea is that we want to represent players as objects, and actions they can take as methods. In order to do that, we need to define a `Player` class, with the following attributes:

- `int type`: the type of player: 0 for human, 1 for computer
- `int strategy`: 0 for stupid, 1 for smart, -1 if the player is human
- `int wins`: the number of games won
- `int stupid_wins`: the number of games won following the stupid strategy if the player is the computer
- `int smart_wins`: the number of games won following the smart strategy if the player is the computer

The Player class should have the following methods:

- `Player(int type)`: this is the constructor. Its parameter is 0 if the player is human, or 1 if it is the computer. If the type is 1, a coin is tossed to decide whether it will follow the stupid strategy or the smart strategy.
- `int chooseMarbles(int current_pile_size)`: this method performs the action of deciding how many marbles to take. It receives as parameter the current number of marbles in the pile. It returns the number of marbles to be taken by the player from the pile. If the player is human, ask the user to enter the number of marbles (ensuring it is legal.) If it is the computer, just choose according to the strategy, as it was done in part 1.
- `void registerWin()`: this method updates the wins, stupid_wins, and/or smart_wins attributes, according to the type of player. This method should be invoked whenever a player wins a game.
- `int getType()`: returns the type of player.
- `int getStrategy()`: returns the strategy if the player is the computer, or -1 if it is human.
- `int getWins()`: return the total number of wins.
- `int getStupidWins()`: return the number of wins following the stupid strategy.
- `int getSmartWins()`: return the number of wins following the smart strategy.

- `double getStupidWinsRatio()`: returns the ratio of stupid wins with respect to total wins if the player is the computer.
- `double getSmartWinsRatio()`: returns the ratio of smart wins with respect to total wins if the player is the computer.

Aside from the `Player` class you should have the "main" class (also called the "driver" class) called `OONim` which should contain the main loop driving the game (in the `main` method.) This class is responsible for creating the appropriate player objects and executing the game itself.

Hint: for the main class you should be able to keep the general structure of your main loop from part 1.

Sample run

A typical execution for the program (both part 1 and part 2,) looks like this:

```
JavaNim 1.0
Welcome to the game of Nim!
Whomever takes the last marble loses.
Game #1
Initial pile size = 81
I take the first turn.
Move #1
        I took 12 marbles.
        The pile has now 69 marbles.
Move #2
        How many marbles will you take? 40
        That is an illegal move.        How many marbles will you take? 30
        You took 30 marbles.
        The pile has now 39 marbles.
Move #3
        I took 4 marbles.
        The pile has now 35 marbles.
Move #4
        How many marbles will you take? 0
        That is an illegal move.        How many marbles will you take? -1
        That is an illegal move.        How many marbles will you take? 1
        You took 1 marbles.
        The pile has now 34 marbles.
Move #5
        I took 7 marbles.
        The pile has now 27 marbles.
Move #6
        How many marbles will you take? 7
        You took 7 marbles.
```

```
        The pile has now 20 marbles.
Move #7
        I took 2 marbles.
        The pile has now 18 marbles.
Move #8
        How many marbles will you take? 9
        You took 9 marbles.
        The pile has now 9 marbles.
Move #9
        I took 3 marbles.
        The pile has now 6 marbles.
Move #10
        How many marbles will you take? 3
        You took 3 marbles.
        The pile has now 3 marbles.
Move #11
        I took 1 marbles.
        The pile has now 2 marbles.
Move #12
        How many marbles will you take? 1
        You took 1 marbles.
        The pile has now 1 marbles.
        You win.
        I used the stupid strategy.
Overall statistics:
        Games played:  1
        Human wins:    1
        Computer wins: 0
                Stupid strategy wins: 0
                Smart strategy wins:  0
        Stupid wins ratio: 0.0
        Smart wins ratio:  0.0
Another game? (y/n) y
Game #2
Initial pile size = 27
I take the first turn.
Move #1
        I took 12 marbles.
        The pile has now 15 marbles.
Move #2
        How many marbles will you take? 8
        That is an illegal move.        How many marbles will you take? 7
        You took 7 marbles.
        The pile has now 8 marbles.
Move #3
        I took 1 marbles.
```

```
        The pile has now 7 marbles.
Move #4
        How many marbles will you take? 3
        You took 3 marbles.
        The pile has now 4 marbles.
Move #5
        I took 1 marbles.
        The pile has now 3 marbles.
Move #6
        How many marbles will you take? 2
        That is an illegal move.        How many marbles will you take? 1
        You took 1 marbles.
        The pile has now 2 marbles.
Move #7
        I took 1 marbles.
        The pile has now 1 marbles.
        I win.
        I used the smart strategy.
Overall statistics:
        Games played:  2
        Human wins:    1
        Computer wins: 1
                Stupid strategy wins: 0
                Smart strategy wins:  1
        Stupid wins ratio: 0.0
        Smart wins ratio:  1.0
Another game? (y/n) n
Bye.
```

Submission instructions

You must submit the following source files:

- `Nim.java`: the non-object oriented program from part 1
- `Player.java`: for part 2
- `OONim.java`: for part 2.

At the beginning of each source file you must put a comment with:

- Your full name
- Your McGill student ID number
- Your section
- The following statement: "I declare that this file is entirely my own work, and that I have not copied any part from anyone else."

Marking scheme

- Part 1: 10 points
  - Syntax/compilation: 1 point
  - Correctness (correct algorithm, structure and execution): 8 points
  - Style (comments, indentation, variable names, etc.): 1 point
- Part 2: 10 points
  - Syntax/compilation: 1 point
  - Correctness (correct algorithm, structure and execution): 8 points
  - Style (comments, indentation, variable names, etc.): 1 point