COMP202 Winter 2005 Assignment #5

Distributed: **March 19th**, Due: **April 1st at 11:55 p.m.**
For the sake of the environment, please don't print this assignment – just view it online.

1 Golf Course Game

Recall assignment 2. You were asked to create a program to simulate a golf practice range. In this assignment you will extend on the idea of hitting golf balls.

You will create a multi-player object-oriented two-dimensional golf course game. Do not be afraid! First understand the problem, then break it up into manageable chunks.

For information about golf, please read Wikipedia's article about Golf:

`http://en.wikipedia.org/wiki/Golf`

First of all, your game will ask how many players want to play, what will be the name and golfing ability of each player, and for how many holes the group wants to play. Immediately after this information is entered, you will summarize in a "welcome" message: who is playing and for how many holes. After each hole is played, you will print a "score sheet" of each player's name and score for all the holes, with 0 (zero) written for yet unplayed holes.

At the end of the game, there should be an option to play more holes. The question should be posed just once, to the group of players, not to each individual player. If the answer is to play more holes, you should ask how many. Your implementation of the game must respond transparently, as if that is how many holes were initally meant to be played. The score cards, etc, must all be updated on the spot.

Your game should be able to omit asking the initial questions (regarding how many players want to play, and for how many holes) if the appropriate arguments are given to your program on the "command line". That is, if the program is executed as follows, the game should begin with the welcome-message-summary of who is playing and for how many holes, and then should proceed to the first hole.

```
java golfgame --holes 7 --players marc 0.9 ernesto 0.8 alex 0.75
```

This will require you to work the String array commonly named "args". For information about the command line and command line arguments, please refer to the world's longest URL:

`http://moncs.cs.mcgill.ca/people/eposse/teaching/comp202-winter-2005/resources/commandline.pdf`

Alternatively, you may use this tiny one: `http://tinyurl.com/4lytg`

Since you are asked to create a multi-player version of the game, you must keep score. There are several approaches to scoring in golf, though in this assignment, you are to implement strokeplay - which means that you are to keep track of the number of shots taken for each hole and then sum them at the end of the entire round to determine the winner (the person with lowest score: the fewest shots taken).

At the end of each hole you must print the score "card". The last column of the score "card" should have a column showing the sum of each player's individual hole scores. For example, the score card may look like what follows, though how you make it look is your choice:

```
Player  |  1  |  2  |  3  |  4  |  5  |  6  |  7  | total
--------+-----+-----+-----+-----+-----+-----+-----+------
ernesto |  4  |  3  |  4  |  5  |  3  |  -  |  -  |  19
marc    |  5  |  4  |  5  |  4  |  4  |  -  |  -  |  22
alex    |  7  |  4  |  4  |  3  |  4  |  -  |  -  |  22
```

The order of play (in terms of the hitting order) for the first hole is determined randomly. The order of play on all subsequent holes must be ordered: the player with the lowest score on the most recently completed hole drives first, and the player with the worst score on that hole goes last. Ties can be broken by random selection. You might want to have a class called "Group" which, by aggregation, has the "Player" representations within it and keeps them sorted by the order of play. Following this line of thought, the Group class may have a method in it called "printScoreCard".

In this implementation of the game, you are not required to implement putting (even though this would be a natural, and not too difficult extension, since putting is like hitting any other club, just softer and with no loft). You may try to implement putting for bonus points if you wish. If you don't implement putting, then for each given player, you should consider the hole to be complete when the player's ball reaches, and stops on the putting green. If the player overshoots, he/she will have to choose a more appropriate club and hitting strength and hit "back" toward the putting green.

A golf "course" consists of several "holes". Each hole consists of several "special areas", such as a teeing ground, hazards, fairway, and a putting green. Your program must generate the **structure** of each hole, by deciding which special areas are included, and the size of each special area.

Consider the following depiction of a hole. The teeing ground is on the far left, and the putting green is on the far right. In between is a hazard, in this case a water hazard. There are several ways to build a hole. One way may be to determine the length of the hole and then randomly decide where the hazard should be, and how long it should be. You are required to implement only one hazard per hole. You may try to implement multiple hazards per hole for bonus points.

```
t----------------------wwwwwwwwwwwwwwwwww----------ppppppppppp---------

teeing                         water                      putting
ground                         hazard                     green
```

The regular stretches of a golf course are called fairways because they have grass that is cut so low that most balls can be easily played.

On each player's turn you will tell the player whether he/she is hitting from the teeing ground, a sand bunker, rough grass, or the fairway. You will also provide the distance to the putting green, and you will ask the player for their club choice and club swing strength. If a given player has finished the hole, and other players have yet to complete it, you simply skip the player's turn, and maybe in passing, say something like "Billy naps...".

When you generate the course, you will generate a given number of holes (the areas between the teeing ground and the putting green). The teeing ground is nothing more than the point from which you will first hit the ball on each hole. The putting green is the area around the actual hole in the ground; its grass is cut very short so that a ball can roll over long distances.

The putting green will be 5 to 15 yards long, determined at random for each hole. In this description, the putting green, like everything else is thought of in terms of length because the game is two-dimensional. One

dimension is the "land" (the x-axis) and the other dimension is the altitude (the y-axis) since a ball follows a trajectory in the "air". If you decide to implement putting, you can choose to place the hole (the actual hole in the "ground") anywhere on the putting green, including the dead centre of every putting green.

In this implementation of the game, a hole will be 130 to 630 yards long. As in the second assignment, you should do your calculations in meters and whenever you display a horizontal distance, you must cite the distance in both units: yards first, and meters second (in round brackets).

Unlike in the second assignment, where the ball always landed at target height, in this assignment, you will pretend that there is perfect gradient between the teeing ground and the putting green. Use the following formulas to determine the time of flight:

- Start with these three equations:

$$y = V_y * t + 1/2 * a * t^2 \quad (1)$$
$$x = V_x * t \quad (2)$$
$$y = mx \text{ (where } m \text{ is the slope)} \quad (3)$$

- Write $x = y/m$ from (3), and plug in (2): $y/m = V_x * t \Longrightarrow t = y/(mV_x)$
- Plug t into (1), and solve for t (time of flight): $t = (2 * m * V_x - V_y)/a$
- To get the new coordinates, plug this t into (1) and (2).

Please use the trigonometric equations and golf contants from assignment 2 for your remaining Mathematical needs.

Remember, your implementation must include one "hazard" on each hole. You can randomly choose to make it either a sand bunker, rough grass, or a water hazard on each hole.

Hitting out of a sand bunker is quite difficult because the golf club tends to wedge itself in the sand, causing a lot of drag. Should the ball land in a sand bunker in your implementation of the game, the player will hit on his/her next turn with significantly decreased club swing speed say, 50% to 75% less than the intended speed. You can play around with this factor and incorporate what you determine to be appropriate.

Rough grass is generally uncut grass that makes it difficult to hit the ball because the grass catches the club, causing some drag. The ball is often held back by the grass as well. You can simulate this dual effect by simply reducing the club swing speed by say, 25% of the intended speed.

If the ball lands in a water hazard, you must count the stroke that got it there. On the player's next turn he/she must hit from the same position from which they hit the ball into the water in the first place.

If you consider the putting green to be a "special area" with the property that the "target" has been reached and play for the hole can stop, then you can extend this idea to create other special areas which are hazards like the sand bunkers and ponds or water streams. Note that creating a parent class called "SpecialArea" and then creating child classes called "PuttingGreen", "SandBunker", "RoughGrass", and "WaterHazard" is not a bad approach. Maybe the subclasses could be "PuttingGreen" and "Hazard" and the subsubclasses of the "Hazard" class could be "SandBunker", "RoughGrass", and "WaterHazard". This is being left open-ended on purpose. You must employ the notions of "class hierarchies" and "designing for inheritance" on your own.

The SpecialArea class might have instance data to keep track of how long the area is, and where the area is relative to the teeing ground. If that is how you decide to tackle this issue then you'll need to have methods that get and set these values. A method that might be useful to have in the SpecialArea class is one that takes the position of the ball in meters away from the teeing ground and returns true or false indicating whether the ball is in the special area or not.

Class suggestions

- Player has-a ScoreCard
- Group has-a set of Player(s)
- SpecialArea
  - PuttingGreen
  - SandBunker
  - RoughGrass
  - WaterHazard
- Hole
- Course has-a set of Hole(s)

Marking scheme

You must submit your source code (.java file) and your bytecode (.class file) to "Assignment 5 box" on WebCT. Clearly indicate, at the top of the source code, your name, student ID number, course number, section number, and the assignment number.

- 3/20 for correct syntax/compilation
- 14/20 for solving the problem (including correct algorithm)
- 3/20 for style (comments, indentation, understandable user interface)

Appendix 1: Converting strings to numbers

To convert the string version of numbers, you can use the Scanner class; however rather than scanning the System input stream, you can scan a string. Consider the following example:

```
String s1 = "123", s2 = "45.67";

Scanner scan = new Scanner(s1);
int i = scan.nextInt();

scan = new Scanner(s2);
float f = scan.nextFloat();

scan = new Scanner(s2);
double d = scan.nextDouble();
```

Alternatively, you may use wrapper classes, as in the following example:

```
int x = Integer.parseInt("123");
double y = Double.parseInt("45.67");
```