Maryfrances Umeora

mumeora

Project 2

Lab Times: TR 11:05-12:20

I did not collaborate with anyone on this assignment... unfortunately

Note to Grader: For some reason, some of my comments keeps disappearing. They may or may not show up when you load my code, so I made this README very detailed.

**Flow of Events**

There are two courses a user can play, and each course has 18 holes.

Each hole is pretty much the same: there’s a physical hole you must get the ball into, a green, and a bunch of clubs you can use at different power levels. The only difference is the mean and standard deviation for each choice you make, which is easily done with methods. Thus, to understand the whole game, you only need to understand what goes on in one hole.

* Each hole has a predetermined yardage and hole distance.
* The user chooses a club. Depending on the club the user chooses, the mean and standard deviation gets set to certain values. Then the user chooses a power.
* The mean, stdDev and power variables are sent to a method called shoot. Once shoot is called, the user’s score goes up by one point.
* All the shoot method does is call the nextDistance() method, which calculates how far the user’s shot went. This value is returned to the user, so the user knows.
* After this is a whole bunch of if statements.
  + If the ball doesn’t make it within 20 yards from the hole but doesn’t pass the hole, the user can decide to shoot again. This keeps happening until the ball makes it within 20 yards of the hole.
  + If the ball glides past the hole, it’s called a flyer, and the user gets to shoot again. This keeps happening until the ball makes it within 20 yards of the hole.
  + If the ball makes it within 20 yards from the hole, the user can now putt. It’s basically the same as shooting, except the club is predetermined as a putt. Once the user putts, the score goes up by one. If the ball doesn’t make it in after the first putt, the user continues to putt. If not, the computer prints something along the lines of “It’s in the hole!” as well as calls the scoreReport() method. Then the game goes on to the next tee unless the user decides to quit.

That’s basically what happens for each tee. And there are 36 tees in the program, all with the same order of events, but different variables.

**The Code**

I used three different classes in my code.

**The Golf Rules Class**

This class is very simple. Once called, it prints out the rules to the game. I did this as an extra-credit feature to make the game more realistic.

**The Tee Class**

This class creates a Tee object with which I use to run a bunch of different methods to do different things. The methods are enumerated later.

**The Main Method**

The code starts by printing out the rules by calling the Golf Rules Class.

Then the user can either choose Ye Olde Course of La Nouveau Course. Both courses are the same save for difference in yardage and par-age. La Nouveau Course is pretty much a copy of Ye Olde Course with a few numbers changed. Let’s say you chose Ye Olde Course. You would automatically be playing the first hole at the first tee. What I’m describing next will be what occurs in the tee methods (either firstTee, otherTees or finalTee).

All the tee methods are pretty much the same. There are two differences:

1. The strings in each one. firstTee prints “Welcome to the first hole!” otherTees prints “Welcome to hole number” + whatever number of tee it is. finalTee prints “You made it to the end!” as well as a “thanks for playing!” message at the end.
2. The first tee is a do-while loop while the second tee and onwards are while loops. The condition is guess != 0. Once the user types in zero for guess, they have quit the game. If they quit after the first tee, the second tee won’t run because it checks the condition first.

At the beginning of the tee, I set the how far away the hole is from the starting point (holeDistance) depending on what tee it is. This value is different for every tee. The user then sets the club and power they want. Then I call the shoot method.

Depending on how far away the user’s ball lands from the ball, I output “the ball is in!” “not even close” or “welcome to the dance floor!”

I consider a ball *in* if it lands within 5 yards of the hole *going forward*. For example, if the hole is 301 yards away and the ball stops at 304, I will say that the ball is in. I decided to this because, in reality, if a ball stopped at the 304-yard mark, it probably hit the ground at like 298 and bounced and kept rolling. As it rolled, it would fall into the hole.

If the user sees “Welcome to the dance floor,” they are now on the green and can only putt. Putting is represented by a do-while loop much like the one for regular shooting. The differences are that the user can’t choose a different club and the if-statements following it aren’t quite as complex. The statement stops once the ball is in i.e. the hole is 0 yards away*.*

If the user sees “Not even close,” they get to shoot again if they so wish to, or quit.

If the ball goes in, the code asks you if you want to hit again, go to the next tee or quit completely. It’d be ridiculous to hit again if the ball is already in, so the user is advised to go on to the next tee or quit the game.

The thing with the code is that it doesn’t specify what direction you’re shooting in. For example, let’s say the hole is 30 yards away but you accidentally hit it 194 yards away. The code will tell you that the hole is now 164 yards away. Of course, in real life, you would be hitting in the opposite direction you were hitting before, but the code doesn’t specify. You have to figure it out for yourself. I could get extra credit for saying so, but that’s too much work. Just note that if the hole suddenly goes from 30 to 164 yards away, it’s not an error on my part. You probably just shot the ball too far.

**Extra Methods Created/Used**

I was able to keep my Main method very neat by using these extra methods.

* Class and Method to Print Rules

This one is quite simple. All it does is print out the rules for the user. I would suggest reading it.

* Tee Class

This is the class that contains all the other methods used in the program. Since it does so many things, I thought Tee would be a clever name. There are a bunch of instance variables and setters and getters for all of them, but the setters and getters don’t really get used. They’re just there by convention.

* Method for Club

This method is called chooseClub(). Based on what club the user chooses, I set the mean standard deviation at default power 10. Of course, later in the program, the value of mean and stdDev actually change depending on the power level the user chooses.

* Method to Shoot

All this really does is call the nextDistance() method, which calculates how far the ball went.

* Method to Putt

This method sets the standard deviation in *feet* depending on what power level the user chose. The values were provided by the teacher.

The thing with putting is that it might tell you that you hit the ball 0 yards! This doesn’t necessarily mean the ball didn’t move. It just means that the distance the ball travelled when you putted isn’t up to a yard. The next sentence will tell you how much distance is left in feet.

* Method to Guess Putt

This method calls Putt(). It basically keeps calling putt in a loop until the ball gets in. Remember, the ball in considered *in* while putting if it lands within one yard of the hole. So if you get “The hole is only 3 feet away!” and then “The ball is in!”, it’s not wrong because 3 feet = 1 yard.

Note that I don’t always call GuessPutt(). I usually only call it if the ball is a flyer, yet lands in the green. If the user’s pure shot goes straight to the green, I just typed the code out instead of calling the GuessPutt() method.

* Method to Calculate Distance

Using the code the teacher provided, this method returns how far the ball went.

* Method to Return Score

Depending on how many hits it took for the user’s ball to get into the hole, there’s a different output. It could be an “albatross” or a “birdie.” Pawlicki said we’d get extra credit for using golfing slang, so that’s why this method is as complex as it is.

* Methods for the Holes

This includes firstTee, otherTees and finalTee. The differences have already been explained above.

**My Courses**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ye Olde Course | | | | | |
| Hole | Yards | Par | Hole | Yards | Par |
| 1 | 530 | 5 | 10 | 433 | 4 |
| 2 | 305 | 4 | 11 | 363 | 4 |
| 3 | 331 | 4 | 12 | 174 | 3 |
| 4 | 201 | 3 | 13 | 545 | 5 |
| 5 | 500 | 5 | 14 | 419 | 4 |
| 6 | 226 | 3 | 15 | 512 | 5 |
| 7 | 409 | 4 | 16 | 410 | 4 |
| 8 | 410 | 4 | 17 | 320 | 4 |
| 9 | 229 | 3 | 18 | 170 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| La Nouveau Course | | | | | |
| Hole | Yards | Par | Hole | Yards | Par |
| 1 | 376 | 4 | 10 | 386 | 4 |
| 2 | 453 | 4 | 11 | 174 | 3 |
| 3 | 397 | 4 | 12 | 348 | 4 |
| 4 | 480 | 4 | 13 | 465 | 4 |
| 5 | 568 | 5 | 14 | 618 | 5 |
| 6 | 412 | 4 | 15 | 455 | 4 |
| 7 | 371 | 4 | 16 | 423 | 4 |
| 8 | 175 | 3 | 17 | 495 | 4 |
| 9 | 352 | 4 | 18 | 357 | 4 |

**Golfing Slang Used**

* Tee: where the players start
* Dance floor: slang for the green
* Flyer: a ball that goes farther than intended
* Hole in one: when you make the hole in one shot
* Albatross: when your score is 3 under par
* Eagle: when your score is 2 under par
* Birdie: when your score is 1 under par
* Bogie: when your score is 1 over par
* Double bogie: when your score is 2 over par
* Triple bogie: when your score is 3 over par

**Requirements**

* user can play one or more courses ✓
* one or more rounds (play again optional) ✓
* keep the user informed of the game situation and their score ✓
* properly compute the length of shots and putts ✓
* correctly determine when the ball is in the hole and inform the user ✓
* must use object-oriented design (classes and objects) ✓
* use methods ✓
* properly commented ✓
* submission must include a README text file or PDF ✓

**Things Done For Extra Credit**

* Use golf slang (worth up to 5% extra credit)
* Allow player to quit whenever (worth up to 5% extra credit)
* Document clearly (up to grader)