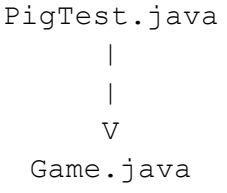
Lecture 13 - 2/27/2024

You didn’t learn any new Java today, but you did learn how to more effectively write the Java you already know. You primarily learned about Top-Down Design or as Professor Cannon put it, “Wishful Programming”. So in this review, I will go over Top-Down design and also talk about Bottom-Up design as well so you will have a better understanding of both of these methods and choose which you think is the best for you when you write your programs.

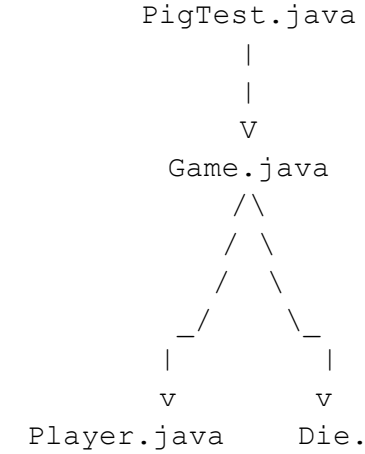
With Top-Down Design we start with the main product that we want to run; From lecture this is the PigTest class, we can visualize it like a tree:

PigTest.java

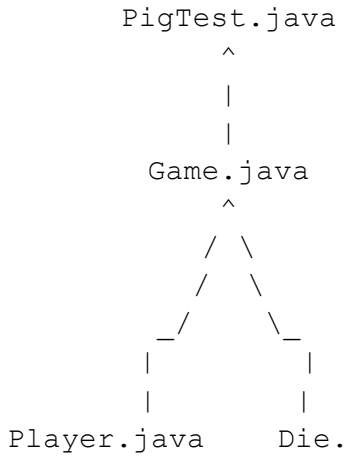
From PigTest we derive another crucial element, we need a Game class so our tree begins to grow:



Well then we start to think about the components of the Game class, well to play a Game you definitely need Players, and in the case of lecture we also needed a die. Our tree now looks as follows:



This could obviously keep going for a while, but in the case of lecture this is the final result. This process is the same for both Top-Down as well as Bottom-Up except the Class Tree is reversed in direction.



(alternatively you could have wrote the arrows the same direction but have the bottom class on the top)

The direction of the arrow should key into the difference between these design techniques. With Top-Down design, you implement the class at the top and work your way down. With Bottom-Up design, you implement the classes on the bottom and work your way up to the main product. My preference is Bottom-Up design, I am not a "Wishful Programmer" like Professor Cannon.

That is pretty much it, you can learn more about the effectiveness of these techniques as well as when it may be better to use one or the other by googling these methods. Best of luck on the current homework!

Lecture 14 - 2/29/2024

There was not much new content covered in this lecture, the main discussion focused on the idea of using simulations to determine optimums for a particular strategy. You saw this explicitly with the PigSim.java file. This is also the crux of the second part of Homework 6. To visualize the simulation needed for HW 6, you have two computers playing the game with each other, trying to play at their best, you'll need to test each effective combination of strategies (this should sound familiar) given this structure we are essentially trying to find the maximum of a bunch of minimums. Consider the following table:

```
1 2 3 -4 6 10  | min is -4
2 4 1 -2 10 9  | min is -2
9 6 2 1 0 20   | min is 0
2 4 9 1 10 -1  | min is -1
```

From this we can compute the maximum which is 0, this is the general gist of how you should work through the simulation.