Fixing Both Errors

Marcar esta página

The error in go()

Hopefully, while going through the code, you wondered why in the world the go() method was set to private. Instead, it should be public since it represents a behavior that a client of Craps would need to start a game. The only reason I was able to demo the game by instantiating a game object and calling go() in the prior video is because Craps is its own client. That is, the main method that created the Craps object and called go() was inside (same) Craps.java. With main and go being members of the Craps class, there was no access issue.

Now, what if we wanted to create a separate client class (*e.g.*, called CrapsLauncher) for starting games? Take a look at the below, which is simply the same main method from Craps.java copied into CrapsLauncher.java.

```
public class CrapsLauncher {
    public static void main (String[] args) {
        // Create an instance of the game of craps
        Craps game = new Craps();
        // Start the game!
        game.go();
    }
}
```

Assuming that you save CrapsLauncher.java in the same folder as Craps.class, compiling it would result in an error like the following:

```
macbook-pro:craps omojokun$ javac CrapsLauncher.java

CrapsLauncher.java:8: error: go() has private access in Craps

        game.go();

            ^
1 error
```

As shown, we can instantiate a Craps object outside of Craps; however, with go() set to private in Craps, there's no way to start a game from that instance.

You might ask why it matters when we can already run our games with Craps. Further, why do we even need a separate client class?

Well, for this simple example, we clearly didn't since we're able to run the game on the command line any time we want by typing: java Craps. However, what if we wanted to take things up a notch and create a suite of card games (Craps, Poker, Blackjack, and Roulette) and package them into a **single** casino program instead of multiple programs that must be run independently? It could list each option as a command line menu as follows:

```
Welcome to the Omojokun Casino!
(Beware, the house always wins)

The available games are:
1 - Craps
2 - Poker
3 - Blackjack
4 - Roulette
```

```
Enter the number for your desired game:
>
```

To process a player's choice, a switch statement could be written as follows:

```
switch (choice) {
case 1:
    // Create an instance of the game of craps
    Craps game = new Craps();
    // Start the game!
    game.go();
    break;
case 2:
    // Create an instance of the game of poker
    Poker game = new Poker();
    // Start the game!
    game.go();
    break;
case 3:
    // Create an instance of the game of blackjack
    Blackjack game = new Blackjack();
    // Start the game!
    game.go();
    break;
case 4:
    // Create an instance of the game of roulette
    Roulette game = new Roulette();
    // Start the game!
    game.go();
    break;
//FYI: You might have noticed some repetition in the above. Later, you'll learn ways to consolidate some of that.
}
```

Based on what you've learned about identifying classes, the code needed for the above menu and switch statement should be contained outside of Craps.class, Poker.class, Blackjack.class, and Roulette.class. The reason is that the menu display and input processing are not singularly tied to one of the four games. In fact, the code is capable of launching **any** of the games. And, just like CrapsLauncher, this new menu class would need access to go() but not have it for Craps or any other game class with same error of privatizing go(). Our goal of a user-friendly way of starting games within a single program would therefore not be possible.

Making go() public is all that would be needed to make it work however.

The error in stage2()

Unlike go(), stage2() should not be a public facing behavior of Craps as it is currently represented. It is intended to be a helper method that contains the logic for the second stage in a game. We certainly do not want a client class of Craps to be able to create an instance and jump directly to stage two, as the code below attempts to do:

```
public class CrapsCheater {
    public static void main (String[] args) {
        // Create an instance of the game of craps
        Craps game = new Craps();
        // Start the game!
        game.stage2();
    }
}
```

It turns out that the above class (CrapsCheater) compiles and runs, even though it violates the game rules presented in the earlier skit by not establishing an initial point. The fact that the code can actually run with that omission is quite problematic and shines further light on the importance of really thinking about visibility **not rushing to make methods public because it might make things easier for you to code**.

While this game is only a toy example, code that allows for the skipping of important steps could create huge security gaps in critical software -- hence the choice of "egregious" in the previous page's title. Imagine if we were writing another kind program that also happened to have two stages in its execution. Let's say that stage one serves as a password entry step and stage two presents the user with sensitive information -- with the assumption that a valid password was provided in stage one. A malicious actor could jump directly to that sensitive information by calling stage2() directly, if the method was haphazardly set to public.

Here's a version of Craps with the two visibility errors addressed:

```
public class Craps {
    private Die die1, die2;
    private int point;
    public Craps() {
        die1 = new Die();
        die2 = new Die();
    }

    private int toss() {
        int total = die1.roll() + die2.roll();
        System.out.println ("Die One: " + die1.getFaceValue()
                            + ", Die Two: " + die2.getFaceValue());
        return total;
    }
    public void go() {
        point = toss();
        System.out.println ("Point: " + point);
        if ((point == 7) || (point == 11)) {
            System.out.println("Winner!");
        }
        else if ((point == 2) || (point == 3) || (point == 12)) {
            System.out.println("You lost!");
        }
        else {
            System.out.println("Entering Stage 2");
            stage2();
        }
    }
    private void stage2() {
        boolean endGame = false;
        while (!endGame) {
            int total = toss();
            System.out.println("Total: " + total);
            if (total == point) {
                System.out.println("Winner!");
                endGame = true;
            }
            else if (total == 7) {
                System.out.println("You Lost!");
                endGame = true;
            }
        }

    }
    public static void main (String[] args) {
        // Create an instance of the game of craps
        Craps game = new Craps();
        // Start the game!
        game.go();
    }
}
```

< Anterior   Siguiente >