Homework 3: Stingy Slot Machine

COSC150

Description:

This assignment is a lot like the Slippery Slot Machine from last semester, except that the Slippery Slot Machine Company is now in chapter 11 (bankruptcy) because they gave away too much money. The *Stingy* Slot Machine has formed with new rules about payoffs. The new company has a program "StingySlot" that models the new payoffs. Your job is to test the new program using JUnit tests.

What does Stingy Slot do?

The slot machine is supposed to work like this: for a single $2 play, five numbers are selected independently and uniformly at random, each between 1 and 60 inclusive (this 5-number sequence we will call a "spin"). Payoff is according to the following rules, and if more than one rule applies, add the payment values.

1. If all 5 numbers are the same, pay $1,000,000.
2. If 4 are the same and 1 is different, pay $10,000.
3. If 3 numbers are the same as each other and the other 2 are also the same as each other (but not the same as the 3, i.e., a "full house"), then pay $500.
4. If 3 numbers are the same and the other 2 are not the same as the 3 or each other, pay $10.
5. If 2 numbers are the same (but not 3 of a kind) the payoff is $2. (If you get 2 pairs, that's nice, just use this rule twice, $4.)
6. Add $0.10 for each perfect square (multiple payoffs possible, for same or different perfect squares).
7. Add $0.35 for each occurrence or 42.
8. Add $0.17 for each number divisible by 17.

Rules 1-5 are mutually exclusive, but rules 6-8 can be used independently of the first 5 and potentially multiple times each.

The program StingySlot class has four methods:

```
int[] doSpin()
```

```
   generates a new random code and returns it.  The class also retains
this spin to use the compute the payoff.

void setSpin( int[] m )
      Set the spin numbers to a specific value

double payoff()
```
computes the payment (in dollars) due the user, If none of the rules (1 through 8) above apply, the function should return 0.

You will write a JUnit fixture named TestStingySlot.java that contains JUnit tests for the `payoff` method of StingySlot.class (which you will be given). Your program should not include a main(). You should also run a test to verify that the company is not guaranteed to lose money this time (that the average payoff is less than $2).

Your unit tests should be sufficiently comprehensive to catch major errors in the payoff logic described above. For your sanity's sake, you can assume that `payoff` does not have any malicious code and does not exhibit misbehavior if it observes a specific random number. (For example, it is not the case that `payoff` works correctly unless the number is 23-19-19-10-31.) In other words, your goal is to write JUnit tests that catch "unintentional" programming errors[1].

Important: I reserve the right to use several different broken StingySlot implementations for evaluating your unit tests. That is, while I'll provide one StingySlot for development of your unit tests, I may use different ones to see how your tester works. Your JUnit tests thus need to be fairly comprehensive.

Note that I will be giving your StingySlot.class in a jar fileThe jar file contains several compiled Java classes. The only one you need to know about is the StingySlot class.

You'll want to add the StingySlot.jar file to your Eclipse project. To do this, once you've created a new Java Project in Eclipse:

1. Go to Project → Properties
2. Select "Java build path"
3. Click on Libraries tab
4. Click on "Add JARS…"
5. Select StingySlot.jar

The plan on this end is to give your code several different implementations of StingySlot, and see if you program correctly decides which of them is buggy.

1

There are two ways to test (that I can think of). One is, call the payoff(m) function with an array m of values you know should have a certain payoff (like 7,7,7,7,5 should pay off $10K I think.) You can handpick a bunch of spins and put your hand-calculated payoffs for each into the Test and then see if it catches any errors.

Method two is to just call payoff() randomly and see if it gives the right amount. I.e., let the system pick random numbers, like a person playing, get m using the getSpin method, then run your own copy of MyStingySlot.payoff(m) and see if your payoff is the same as the class under test. So, you have to write your own copy of StingySlot, and implement the payoff rules. The good news is, I'm allowing you to share your individual MyStingySlot.jar files. Testing someone else's code is really the best way, so you are encouraged to test other people's StingySlot code to see if their code is right and if your tester is right. You can even challenge each other ... put in hard to find bugs, even the malicious kind mentioned above, to see how good your testers are. You are NOT to share test code. Got it?

What to turn in:

You will turn in a single zip file that contains both **TestStingySlot.java** and any other files you created to make this work, including some version of **MyStingySlot.java**. Your files should be compiled in a package that is named your netID. When I unzip the file, I should get that package/folder. You do not need to include *my* StingySlot.jar or anything else.

If you have trouble creating .zip files, please post a note on Piazza.

Where to go for help:

For questions about this assignment, please post to Piazza at https://piazza.com/georgetown/fall2016/cosc150/home.