CIS*3260 W21 – Assignment 1

Questions and Answers – Part 1

Is [randomize()] simply to "roll" the dice and "flip" the coin?

Yes.

If so, would it need to check the attributes of the instance to know which of the two methods to call

Think of it the other way around. The randomize() method is the general one. It can have a default method that can be overridden by randomize() in Coin and Die if necessary (the default method might work fine, depending on how you write it). Remember, since randomize() is a method in the abstract class, it must be present in the subclasses.

If you can't think of a general default way of writing randomize() in the abstract class, just have it raise an exception informing the user that randomize() must be overridden in an implementing subclass in order for the class to function properly.

Inside Coin, you can have randomize() call flip(), or flip() call randomize(), depending on how you want to write it.

In the players tally, sum, and results method, what are the descriptions for? Why would we set a description on a throw that has already occurred? Are these methods intended to further filter the results for throws? I am quite confused by why they are there.

Can you elaborate as to why you have them as parameters for those 3 methods?

Let's say you have a Throw object stored in throw1 that has a number of dice and coins results.
We can ask

```
throw1.tally
```

and find out that there are 36 randomizers in throw1.
We now set the description

```
throw1.description { item: :die }
```

When we now ask `throw1.tally` it comes back with 6.
Now let us run

```
throw1.description { item: :coin }
throw1.tally
```

This returns 30, i.e we had 6 dice and 30 coins in the cup when we performed the throw method.

---

1. **For cup.throw(), Can we just make a copy of the rc's and store it in the Throw object while also keeping it in Cup since we need to be able to access all the other attributes of the rc's in the Throw object as well?**

Yes. There are many ways to do it, but that is probably the easiest.

2. **But then, also how would we store items in a Throw object since it doens't have a store method? Can we create our own method for it?**

You store it using the constructor. The constructor is given an instance of the cup, and the constructor (the initialize method) can then query the cup and store the information in an instance variable inside the Throw object being constructed.

3. **Can we create our own methods as well? For instance, a method that returns the rc array so cup.getArray would return an array of all the randomizers of the cup object. Or as I mentioned, a store method for Throw object**

You may add any methods you like. However, coin.getArray is bad OO style since you should never yank the internals of an object. That violates the principle of encapsulation. Taking out an array is especially dangerous. It is much safer to request one randomizer at a time from the object, and leave the array inside.
One way to get the object out, is to empty the cup into a Hand and repeatedly ask "next" to look at all the randomizers and record their results. This would leave the cup empty. However you could take the items and place them into a new hand, then reload the cup. Alternatively, if you made a duplicate of the cup (by creating a method that ask the cup for a duplicate), as you suggest in 1, then the original cup is already still loaded and you don't have to reload.

4. **For the sum method of throw, if I had such results**
   (Dice1:4, Coin1: H, Dice2 :3, Coin2: T), what would the sum method return? Would it be 4132? if not, what would it be?

4 + 1 + 3 + 0 = 8

5. **Also, do we return a string or integer as the result?**

For sum and tally, you return an integer.
For results, you can decide. I would return an array.
In your example above, I would return [4, :H, 3, :T]

6. **Is there going to be unorthodox descriptions like { item: :coin, colour: red } or even put a different type of item like item: :box ?**

While I do error checking in my code, the specification is silent on the matter.

7. **Is it okay to use initialize instead of create_coin/create_die**

The assignment has already been changed, removing create_coin/create_die and it now uses initialize()

8. **Can we have keywords in our initialize/create function: for instance, def initalize(sides:, colour:) so that when you make a call to it, you would do it like so: coin = coin.New(sides:5, colour: :red)**

No, stick to the interface provided. Otherwise your code will not run on our tests.

For classes which don't have an explicit initialize() method signature stated in the assignment instructions, I understand that instances will be instantiated through ClassName.new(). But can I add a parameter in which there is a default value? For example, in Hand I have def initialize(randomizers = []) { … } in which I pass in an array of randomizers to store in the container.

That would be fine, as it would work with our test suite. Actually, it is a nice design extension.

Should we denote whether our tests have passed or failed by comparing expected output vs actual output?

Yes, but this is just something you have to report, not something that has to be automated.

Given the randomized nature of many of the use cases, is it sufficient to verify the return object is as expected, the result is not nil, etc since in some cases we can't guarantee the exact result?

I am looking for typical results, as opposed to provably correct results (which you would need statistics for).

Is there some kind of format you would like for testgames.rb?

Your choice.

How many use cases should we aim for?

Number of use cases: 3-5

I heard that we aren't asserting anything. So, are we to just print expected and actual results of use cases to the terminal?

You only need to print the actual results to the terminal. The expected results just goes in your report.

Should we have one test per use case, or several?

The use case is the "test". As I mentioned in class, these are not to be considered actual tests as prescribed in the Software Testing course (although they roughly match what is called system testing, as opposed to Unit testing and Integration testing). Think of it as exercising the system as a whole as opposed to each method individually, as you would do in a unit test.

Should they be unit tests?

No. I just want the use cases run and reported.

How can we report an expected result if we are working with random values.

Report typical results and see if you get typical results. They do not have to match exactly. For example, if I was to test sum, by having two players throw their cups and have the player with the greater sum win. I then print the results, check the sums and see if the correct player won the game. This exercises sum nicely (as well as throw).