CIS*3260 W21 – Assignment 1

Questions and Answers – Part 2

Also, for the store(randomizer) methods, can I create a duplicate of the randomizer and then store it in the randomizer Container's array? Because say if I create a Coin, randomize it, then store the original version of the Coin in a bag, its result would be nil since we reset it. Now, if I tried to store that same Coin in a cup right after storing it in the Bag, then since we stored that Coin in the bag and reset its results there, the same Coin in the cup would also be reset since we are referring to the same exact object but we wouldn't want that.

You have come up with an impossible use case. A single instance of a randomizer, like a coin, should not be in two places at once when using the system I specified. Either the coin is in the cup, or it is in a bag. The interface I created is designed to enforce that.

Notice that the enforcing is happening through the interface, i.e. the design. The language does allow you to store an instance in two places, with the possible negative interaction effect you have hypothesized.

Whereas, if we create a duplicate of the item right before we store it, it prevents this issue

However, then you lose the property of “identity” of objects in the model. When you create a coin, the way I have the design set up (based on the model of the system I am using) it should only be in one place at a time. If it is in a bag it cannot be in a cup or hand. If it is in one hand, it cannot be in another.

That is not to say the duplicates cannot be made. However, you should treat the duplicate as a new coin or die that happens to have the same settings as the coin or die you have based it on (duplicated it from). In my solution, I use this approach to store the information I need in Throw, allowing a “record” of a die or coin to exist.

Creating too many duplicates isn’t good from a separate standpoint: too many duplicates will ultimately require you to abandon many. This clogs up the memory and invokes the garbage collection more often than need be, slowing the system down.

So while your approach will work, I would not use it the way you have suggested … and the sytem should never be used in the manner you have suggested (placing a coin back in the bag while still having it in the cup: the coin should either be in the cup, in a hand for transportation, or in the bag)

When we use the method move_all does the original container get emptied as well?

Yes.

For the select method in Bag class, are you every going to ask for more items than there actually are? E.g. ask for 3 when there are only 2 items in the bag

I could. This is in the specs. If I ask for 3 and there are only 2, you store the 2 in the hand and return it. In other words, the number should be considered as a maximum rather than exact value requested.

If the description in the select method from the Bag class is impossible do we still return a hand holding nothing or just return nil?
I envisioned you returning an empty hand. However, as the spec is silent on this, you could return nil if you desired.

Is it okay that I wrote 3 use cases and then from those use cases I create 10 test cases that ensure that the API can accomplish the 3 use cases? That we create use cases and then from the use cases we can derive test cases that helps achieve tasks required by a use case.

You did it the way it is taught in Software Testing. So I will accept it as is.

However, I actually wanted something a bit more “simple”. Just run the use cases that you designed and report the results, making sure they match what you expected in the use case.

From a software testing point of view, this approach is approximately what is done for the “systems” testing phase … as opposed to “unit” tests or “intermediate” tests.

Just FYI.