Ryan Elias

Lab 7 Documentation

In an attempt to remain organized, I am first going to create all subclasses of class Person first and implement all of their inherited methods according to the UML.

The first thing I noticed about the two classes that extend from Person is that they are both to be sorted differently. I am trying to figure out if I need to create separate comparable classes like in lab 6 but I do not think that is necessary as the class Person has an abstract method of compareToImpl which is called by the compareTo method leading me to believe that if the compareTo method is called for that subclass then it will use the corresponding compareToImpl method. So, I will write those methods first.

SickPerson(s) are ordered by their severity in decreasing order, so largest to smallest. I also just realized that SickPerson requires an additional severity value that is not inherited by the superclass. One way to circumvent this I think is to create an int value severity in the superclass to be inherited by both subclasses and then initialize all HealthyPerson(s) severity to -1 since, after all, they are healthy.

That was my first thought, but after reading the message board on the canvas page and looking at the hint about the instanceOf keyword, I can just set an if statement. However, even if Person p was an instaneOf SickPerson, I would still need to call a getSeverity method on that type Person, which would probably result in an error.

After looking over the UML a bit longer I have a slightly better understand of how these classes are going to interact. It seems that each Hospital will sort their list (I will most likely be using ArrayLists as that is simplest in my opinion) differently. With QueueHospital being the first in first out, meaning regardless of patient type or severity the first Object added to that list will be the next Person or Animal to be treated. StackHospital being last in first out regardless of Type and the PriorityQueueHospital sorting by the natural ordering of each type. Although the first question that comes to mind is what happens when there is an instance of SickPerson being compared to Animal in that Hospital? I think it is best to revisit this later and return to SickPerson and HealthyPerson.

After looking at the driver it seems the constructor for SickPerson implemented by the TA that wrote that code calls for a second int value to be passed, which means that my suspicion of adding int severity to the constructor was correct. HealthyPerson constructor called a second String value which can only be the Reason.

After considering the write up for the lab and the class Person being green in the UML, creating a method and variable in that class seems like it is not the right move. I’m confident that it would solve the problem just not in the way intended I’m sure.

Well my earlier assumption was incorrect and java is apparently smarter than I gave it credit for, after writing if p instanceof SickPerson, it understood that in the following statement p was SickPerson and not Person and I was able to use the getSeverity method I created in that class provided I casted p to SickPerson. **However, I am assuming that when SickPerson and HealthyPerson are being compared that the SickPerson will automatically take precedence.**

I can’t find anything that specifies what the toString should output for SickPerson or HealthyPerson however the abstract method allPatientInfo in the Hospital class is supposed to print out all information truncated. So this leads me to believe that I am to print out, in the case of SickPerson, ”this.name,this.age,this.severity” separated by commas. If that is the case should I not also add another comma at the end of severity? The Animal toString method prints the age and breed of that animal so I think the toString for SickPerson should be “%s, a %d year old with severity %d.” with the values being name, age, severity. And as long as nothing else comes up I should be done with SickPerson for the time being.

Moving along to HealthyPerson, I expanded the constructor to take a second String that will be the reason for visit. And also create a getter for that String. I am also going to write the toString first as I already know what it will look like. Essentially SickPerson’s toString but replacing the severity int with a reason String.

Moving on to the compareToImpl method, HealthyPerson(s) are being sorted by their first name alphabetically. So in order to do that it seems I must create an array of chars for each item being compared using the toCharArray method used in project 2. I created one char array called self and another called pLetters. Aftewords, I will create the outer if-else statement with the instanceof keyword. Like I did with SickPerson, if the instanceof p is NOT HealthyPerson, I defaulted the precedence to go to the SickPerson. Inside of the if statement I simply ran the String class compareTo method to determine which String was had alphabetical precedence.

Following my two Person subclasses I will go ahead and create all 3 hospitals and attempt get their “easier” methods done first. I had the IDE create placeholder methods for all abstract methods in class Hospital because it was easier than typing them all out. Best way I can figure to go about this is to just pick one and start from there.

So the first Hospital child class I’ll work on is the StackHospital class. First of all I need a way to store patients in a list. ArrayLists are usually my go to as they allow me to easily extend the size of the ArrayList. I made the ArrayList of type Object so that I could encompass all patientTypes. I’m sure there is another way to do this but this is what came to mind first.

This doesn’t have anything to do with the Hospital classes but I just realized the compareTo method for Animal is way simpler than the ones I made for SickPerson and HealthyPerson. Making a note of that and I will go back in and change it later.

Back to StackHospital, the methods were relatively simple to write. I used an ArrayList and used the add and remove methods for that ArrayList. Add will add the new patient to the end of the list automatically. Remove(int index) will remove the object at that index. On the nextPatient and treatNextPatient methods I returned the index of length – 1 because all arrays begin at index 0.

For the allPatientInfo method I ran a for each loop and added the result of that objects’ toString method to the running String answer and then returned answer.

I just realized that my return types for most of my methods in StackHospital were of return type Object and not the generic PatientType. I changed these accordingly.

Moving on to QueueHospital, which treats the first patient first and then moves down the list regardless of precedence, FIFO. The first thing was to create the list that this class will store patients in. Then I copied my code from StackHospital’s allPatientInfo method to QueueHospitals allPatientInfo method because it does the same exact thing.

These methods should be just as simple as StackHospital. ArrayList.add simply adds the object to the end of the list and the next patient should always be index at 0. However, that last part is only true if I can move all objects after index 0 to the left in that list. After testing the remove method with index 0 of some ArrayList I have found that all the items shift automatically so that’s pretty neat.

After writing the code to reflect my thought process in the last paragraph I can’t help but get the “that was too easy” feeling. But the last Class that needs to be written is the PriorityQueueHosptial which is probably going to be the most complex.

Like the other Hospital subclasses the first thing I did was make an ArrayList that will contain all of the patients. Then I wrote all of the methods that did not have anything to do with actually sorting the list because those were going to be the difficult ones.

It also just occurred to me as I referenced the UML again that I did not create default constructors for my hospitals. Will fix later.

After reviewing lab 6 and the shape comparators it occurred to me that I am sorting by the default sort method for each type. So theoretically, Collections.sort(patients) should take care of that for me.

After messing with the driver a bit everything seems to be in order. Now I am going to fix the two things that caught my attention earlier and then begin to write tests. I mean creating default constructors isn’t necessary because java will do it for me but it is in the UML so I will put them in.