Homework 4 Trends

Nov 3, 2022

Q. 1:

What went well:

- The majority of the students have provided an appropriate design for SwVx class. The specifications provided were thorough and made the design more understandable.
- The majority of the students have used forward engineering to generate code from design. Almost everyone provided source code consistent with their UML class diagrams.
- A good number of students have established a correct generalization relationship between a class that needs to be a common class for the SwVx and SwEx classes.
- A good number of students have calculated the time on EarthX time differently than the time on VenusX as pointed out in the problem statement.
- The majority of the students have identified the correct relationships between different classes.
- The majority of the students have defined the correct set of arguments and return types for methods.
- Many students have specified the multiplicity and visibility of class attributes and methods which are essential for detailed design.
- A few students have also added notes in the design along to highlight certain aspects of their specifications.

What can be improved:

- Few students didn't create the expected class that serves as a common class for the SwEx and SwVx classes.
- Few students used the same logic to calculate the elapsed time for VenusX. The elapsed time calculation should be for VenusX.
- Detailed specifications of classes, methods, and attributes of classes defined in the UML class diagram are of utmost importance. It lets others know what specific design choices are made. Choices of visibility, scope, return type, arguments, and constraints for methods set the expectations for coders. Similarly, detailing such as visibility, scope, type, tags, and constraints for the attributes should be designed and then used by coders. A few students did not provide the specifications as asked for in the problem statement.
- A few students added comments in the UML class diagram about how tick would increment in both EarthX and VenusX, but that functionality is not incorporated in the UML class

diagram and is available in the generated code. For example, if the same Timer class is used and the incrementTime method accepts the tick, then your design should have a way to pass the different tick with respect to your stopwatch. There were a few inconsistencies in some answers.

- A few students have not included important classes, such as Timer, in their UML class diagrams.
- In a few cases, methods in parent and child classes have identical specifications. They were not different anyway (e.g., given a base and its subclass, one being abstract in the base class and another being concrete in the subclass). Thus, if you are overriding a concrete method from a base class, you need to provide a valid specification saying why it is getting overridden in a base class and how it is different from the base class.
- The logic to calculate the elapsed time differently from that of EarthX is missing. The attributes of time conversion ratio, tick or scaling factor, and their usage in some method to determine the difference between these two planets is essential. A few students defined two different timer classes that are used by SwEx and SwVx. But the timers have the same specification.
- Detailed, appropriate design choices of attributes and methods were important. The same applies to the design choices for relationships among classes. A few students did not pay much attention to these factors while developing their designs.

Q. 2:

What went well:

- The majority of students identified two benefits. They described the benefits in relation to the abstract and concrete classes.

What can be improved:

- Few students identified two benefits of an interface but did not compare it to abstract and concrete classes.
- Benefits provided by a few students were not quite meaningful to consider as benefits in comparison with abstract and concrete classes.
- Some students have provided only one benefit.

Q. 3:

What went well:

- The majority of the students extended the design properly to use the interface and provided the appropriate implementation class for it.
- It is important to show how the added interface is getting used in the design. Many were able to account for showing there is a class that uses the interface.
- The interface is added to the design for reusability and to "hide how it is working". Many students were able to represent this in the design by not providing the direct association of the class implementing an interface and the user class. (For example, if the class **Timer** is a realization class for the **ITimer** interface and **SwVx** class needs to use the Timer, then a good design includes a "usage" relationship between SwVX and the interface, instead of direct association with a class that has a realization relationship to the interface.
- A few students have also provided notes and proper comments to make the design more understandable. Also, a few students have provided the source code with detailed specifications for the additional changes as well.

What can be improved:

- An interface can not have attributes that can change during execution. An interface can have attributes that have constant values. A few students have provided attributes such as counter which is expected to change it value.
- Interface method should be accessible to its users. A method that is private is inaccessible to other classifiers. A few students have not taken the visibility into consideration.
- A few students have not shown the realization class (also referred to as implementation classes) for the interface.
- A realization relationship is required between the interface and its realization classes (i.e., one or more classes the implement the method signatures defined in the interface), and some students have shown incorrect relationships such as composition or unsuitable association relationship.
- Few students have not provided meaningful visibilities for attributes and methods in their UML classes.
- The interface is defined to hide "**how the methods are working**", and thus, some class should be provided that benefits from the interface. Such a class that is using the interface (not its implementation) was missing in a few answers. This is a "user" class.
- A class that has a realization relationship with an interface should not have a relationship to classes that are expected to use the interface.
- It was expected to modify the design provided in problem 1 as a part of question 3. A small percentage of students have provided the same answer as question 1 for this problem. Also, it was expected to use at least one interface in design, few students did not provide any interface for the solution of problem 3.

- Some students have given an idea about which class can be converted into an interface, or they have provided an interface class. A design was asked to be developed.

Q. Formatting:

What went well:

- Almost everyone adhered to the formatting specified. The class diagrams were readable and made examining them easier.

What can be improved:

- Many students have not followed the submission guidelines for Gradescope. It is important to align the pages with the appropriate question number. This is important for the questions that have answers spanning more multiple pages.