Software Engineering

Assignment 2 - Report

Dương Đăng Hưng – BI10-073

University of Science and Technology of Hanoi

March 2021

Question 7.

7.1.

	Comparable	Document
Purpose	- The purpose of using interface Comparable is to apply method compareTo() to classes of the Student type hierarchy, makes them *comparable* and then we can sort them in orders. - Furthermore, in class Engine, we will declare objects as a TreeSet of Student, in order to do so, according to Java API, collection type TreeSet can only be applied for classes which already have implemented Comparable or Comparator.	- The purpose of using interface Document is to apply method toHtmlDoc() to classes of the Student type hierarchy, so that objects of this type hierarchy can generate a simple HTML document from the current state of itself.
Another way	Instead of implementing Comparable, we can manually write a method to compare 2 names by changing the first character of the name to integer type then compare them.	Instead of creating and implementing Document, we can manually create method toHtmlDoc() in class Student then override it in the classes of Student type hierarchy.
When to use the other way	- When abstraction, multiple inheritance and polymorphism are not required. - When we don't really need communications between objects.	
When not to use the other way	- When abstraction, multiple inheritance and polymorphism are required. - When we need to guarantee that classes contain concrete implementations of toHtmlDoc() and compareTo(), and we are able to invoke these methods safely. Therefore, two objects can communicate based on the "contract" defined in these interface.	

7.2.

- TreeSet is a member of the Java Collections Framework. It will perform all element (which are Student objects) comparisons, using its compareTo() method. And it has a few useful properties:

- It keeps sorted data, maintain objects in sorted order, and provides us methods such as search, insert and delete, which are really beneficial for managing Student objects.
- Different from most other types of Set or Collection, TreeSet does not allow null Object and throw NullPointerException, make sense since we do not allow null Student objects.
- TreeSet does not allow duplicated elements, which is the same as we do not allow duplicated Student objects.

- It is possible to develop the same software functionality without using TreeSet. Instead, we could use any Java Collection Classes to store Student objects.

When new elements are added into the collection:

- By making Student objects comparable (implement the interfaces or creating a method as proposed in 7.1), we can manually write a method that will sort the Student objects in the collection using sorting algorithms (merge sort, quick sort, heap sort, etc.).
- Adding the constraint “element cannot be null” when new elements are added into the collection.
- Adding a stage to verify if new elements added are already existed in the collection

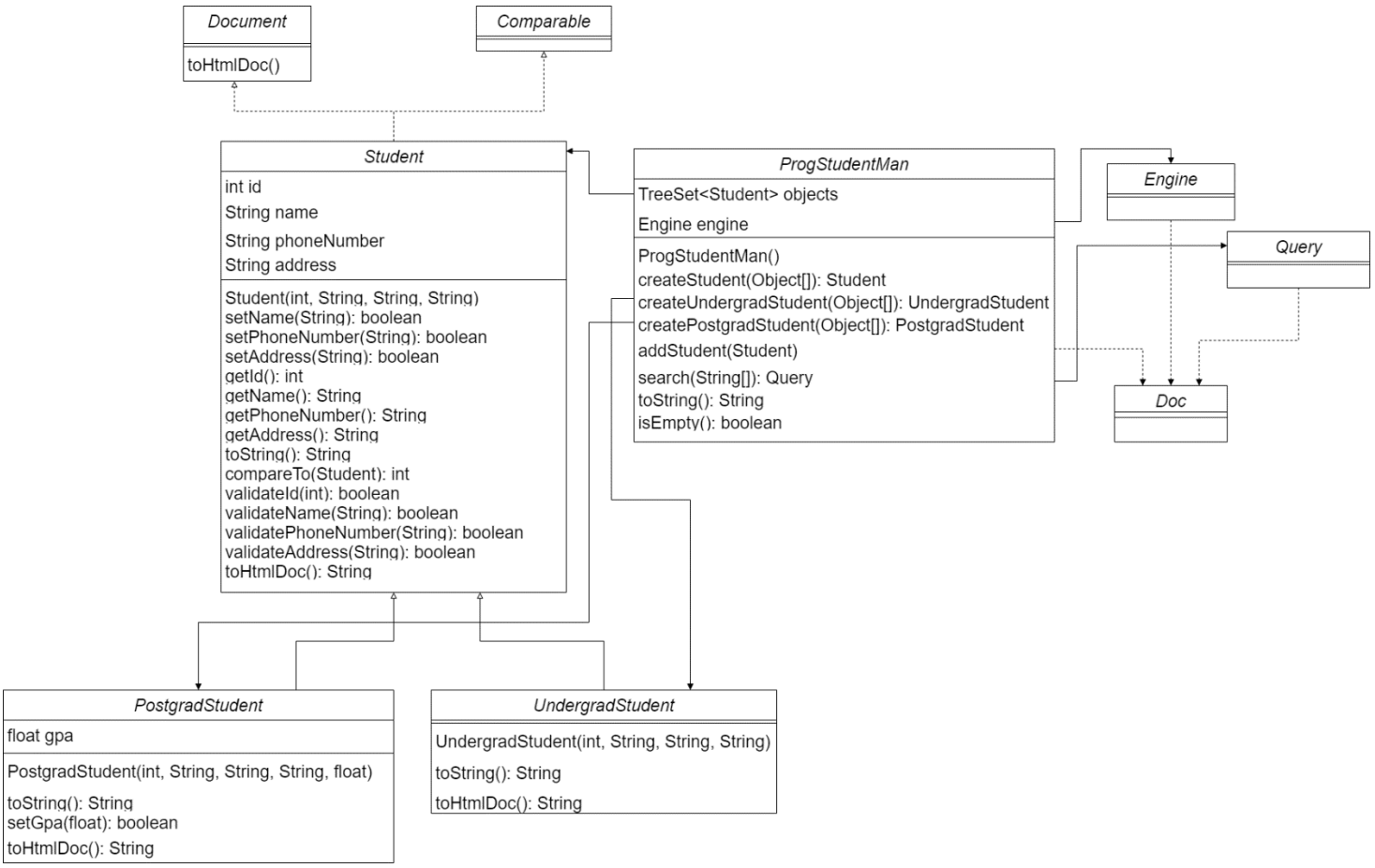
- We should not use this proposed way because it will cost a lot more time and may make the code look messy. We should maximize the convenience that Java already provided.

7.3.

- Interface Document with the method toHtmlDoc() is the key factor that makes it possible to search for Student objects using keywords.
- It generate somewhat of simple HTML Documents from the state of the Student objects, and use those document to search and return objects that matches.
- Also method search() created in ProgStudentMan directly participate in the process of searching for student objects.

7.4.

A complete UML design class diagram of the software:



7.5.

- Based on the design class diagram and the previous tasks:

- First, in assignment 1, we created Student type hierarchy with their attributes and essential methods
- Secondly, starting with assignment 2, we created interface Document and update the 3 Student classes
- Thirdly, we start to deal with class Query
- Then, we continue with a more general class - Engine
- Finally, we finish by completing our software - ProgStudentMan

→ In conclusion, the implementation strategy that was used to build the software is Bottom-Up.

- Using this strategy, we have some advantages and also some disadvantages:

Advantages	**Disadvantages**
- More general and more reusable. - Less resource up front - We have early prototypes of sub-systems - Therefore, testing is simplified because no stubs are needed.	- Late detection of design errors. For example, there may be some errors in Query that could only be detected when writing the code for ProgStudentMan. - More test driver coding: one driver per component