Amanda Trinh

1) Requirements:

1. At least one relational (==, !=, >, >=, <, <=) and one logical operators ( &&, ||, !)

**>= @ AP, line 64**

*|| @ Driver, line 61*

2. At least one “if-then-else” statements.

**@ Regular, line 88**

3. At least one of EACH of the following: “for-each”, “while” and “for” loops.

*For-each @ Driver, line 51*

**While @ Honors, line 52**

*For @ Driver, line 58*

4. At least ONE student-designed interface and at least THREE student-designed classes (one of which MUST be abstract), not including your driver class – which brings the minimum total to FIVE.

Abstract – Transcript

Interface – CalculateGPA

Subclasses – Regular, Honors, AP

5. An Inheritance hierarchy must be implemented with the student-designed classes.

6. Polymorphism must be implemented with the student-designed classes.

8. An ArrayList must be used in at least ONE student-designed class and it MUST be involved in some non-trivial

processing task.

**ArrayList @ Driver, line 31**

9. Comments explaining logic and operation of program at “key points” .

10. Meaningful variable/class names throughout (class, methods, variable name, instance variables, etc.)

11. Read from and/or write to a “text” file (or some equivalent interaction with external media).

*Read from file @ AP, line 56*

2) A description of specifically what my program does and how it does it.

My program GPA, first asks the user whether or not they would like to see their full school Transcript with or without AP courses they may go to for their current courses. Then accordingly to their answer if no (N), the driver will print out the subclasses’ toString for each individual courses (already created) that includes Type of Course and its’ name, grade for the class, if they earned credit, and if the course is weighted. If they answer yes (Y), will print out everything mentioned before and additionally whether the course has an available AP exam/course. Lastly, the driver will calculate and output the student’s GPA from calling the method, getGradePoint().

The abstract class holds all of the method and variables that correspond with the output mentioned above that’s included in the Transcript and the interface gives the method for getting the amount of points earned towards the GPA. There are 3 student-designed subclasses, Regular, Honors, and AP and each of them overrides the toString() method from the abstract class and have their own way of calculating grade-point as their main subclass functions. Also, the Regular and Honors subclass both have a method called iseligibleAP() that reads from a text file that contains a list of the names of all available AP courses and checks if these object courses have an AP equal or not.

In the Driver class, this is where the JVM receives its message print out everything. This is also where I have already instantiated all the objects that correspond to their subclasses for type of course and asks the user for their input on whether or not they would like to know available AP courses for the current courses that they are taking. This class also wraps all the course objects’ method, getGradePoint(), and then instantiated into multiple variables that will then be used to calculate the GPA. Last thing the Driver then does is outputs this GPA.

3) What I learned!

My problem with my Driver class being unable to call some of my object methods from my subclasses, eventually giving me an error saying it cannot find the method, led me to use wrapper classes in order to wrap these methods up. This project made me search online and in the textbook multiple times to further my understanding of how an ArrayList works and also taught me multiple times by my complaints earlier haha about my problem with not being able to grasp the whole scope of variables in my subclasses. How I furthered my understanding of the scopes of variables? Well, first I had a Boolean variable named eligibleAP in my Regular and Honors subclasses and a method called iseligibleAP() that was supposed to determine whether or not these types of object courses had an AP equivalent or not. At the time, I failed to realize that I made the mistake of instantiating eligibleAP to equal false on the “outside” of all my methods in the subclass, instead of in the iseligibleAP() method. This led to numerous problems including why the toString() actually had eligibleAP always instantiated as false and never true. I actually ended up moving the whole purpose of this method into the driver class. I almost failed to realize until the day after the rough draft due date that this removal of the method made my Honors subclass have none of its own method/variable. I ended up creating the method again in both subclasses and this time I actually saw my error and was able to fix my mistake. Additionally, this method included a read from text file inside and I there was a huge problem with my method not being able to throw out exceptions so I ended up doing a lot of research over winter break and even asked people on stackoverflow.com for help with my coding, so I ended up learning a whole lot about how the throws IOException keywords worked and a little about try and catch. Lastly, off topic, but I also learned how you can’t bold, italizes, color, etc. printed text straight from my Driver class like in HTML for example.

4) Extras: I wrapped doubles…...

I used instanceof and I technically used a static method ( public static void main )