Q1. Identify and explain problems prevalent in the students’ design

This fails to separate what changes. The AmazonDataStandardizer is directly reliant on DataStandardizer to print out its data. What should happen is that there should be some Parseable interface such that we can add support for all of the companies we currently have in addition to any further companies. This interface will allow us to simply have several Parseable behaviors “Amazon, Google, Standard, anything else” which will minimize code duplication and simply allow the DataStandardizer to do the same thing everytime which is called Parseable object.parse();

Q2. Explain why sub-classing DataStandardizer and overriding the parse() method is not the best idea (think about: what if someone else needs to add support for yet another format? Which class does that person subclass? What if those people don’t talk to each other?)

If the user needs to add support for another data format they need to extend the class that is lowest in the hierarchy to add handling for all data formats forming a super calling tree. But they have no real way of knowing which subclass of datastandardizer is lower in the tree without asking so they are unable to find out which class to subclass.

Q3. Create a UML Class Diagram to present your new design idea and explain it in a few lines.

Basically we break the behaviors up into the little extensions off of the interface and we separate the classes so that all of them are cohesive. In that they only do the things related to their functionality, the UI is generated entirely by itself and all of the parsing is handled by things that only parse.