**Jakob Roberts - V00484900 - SENG 371 - Lab 8**

**Question 1: This is a refactoring lab. Why was it important that we run through the steps above? What did we learn about the project? What did we learn about the unit tests? What can we infer about changes we make moving forward in the context of this knowledge? Put succinctly: why are unit tests important in the context of legacy software?**

It was important to run through the steps above to show how an error can occur during a test. This taught us how the project was built and how each class has a tester class associated with it. We can infer that any changes we make need to be confirmed with a/the tester class. The unit tests are important because they ensure that everything is working the way they should prior to compilation/deployment.

**Question 2: Why was this refactor a bad idea?**

Because the “available” attribute is being used by the Cell class and not just its children!

**Did Eclipse break anything --- or are the refactoring tools pretty solid?**

They are pretty solid, nothing broke after the pull down and pull up!

**Question 3: Discuss the pros and cons of the different method signature options here.**

The first way, not including rent and monopolies, could mean that the method generated could be used more broadly by multiple functions as it takes in the parameters it needs to do its work. By including the rent and monopolies, it simplifies the getRent() method so it doesn’t to more than its name describes.

**Question 4: Why is this type of refactor difficult?**

Because on an abstract method, often times a lot of other classes extend the abstract method’s class. When changing the information on the abstract method, it requires all other methods to be changed in the same fashion. Eclipse makes it easy, but doing it by hand would be a difficult challenge.