**Group 4**

**Objective #1:** Perform refactoring operations to remove the smell on the code component using the automated refactoring support in IntelliJ.

* Method/Class selected for refactoring: createMissionTripItem() of ItemModelMapper class
* Sign and symptom: message chains
* Treatment: (1) hidden delegation, or (2) extract method/variable
* Refactoring description:

1. Create a test case named ItemModelMapperTest to make sure the behavior of createMissionTripItem remains the same after refactoring.
2. Analyze the bad smell - missionTrip.getMissionCity().getMissionCountry().getName() - that exists in the createMissiontripItem() method of ItemModelMapper class.
3. There are three objects involved in this message chain, which are missionTrip, missionCity, and missionCountry. **NOTE: We found what appears to be a copy/paste bug.**

*// Before Bug Fix*

**if** (missionTrip.getMissionCity() != **null**)

missionTripItem.setTripCity(missionTrip.getMissionCity().getName());

**if** (missionTrip.getMissionCity() != **null**)

missionTripItem.setTripCountry(missionTrip.getMissionCity().getMissionCountry().getName());

*// After Bug Fix*

**if** (missionTrip.getMissionCity() != **null**) {

missionTripItem.setTripCity(missionTrip.getMissionCity().getName());

**if** (missionTrip.getMissionCity().getMissionCountry() != **null**)

missionTripItem.setTripCountry(missionTrip.getMissionCity().getMissionCountry().getName());

}

1. Highlight missionTrip.getMissionCity(), right-click, and select “Refactor → Extract → Variable” within IntelliJ to extract it into a variable shown as below:

**final** IMissionCity missionCity = missionTrip.getMissionCity();

**if** (missionCity != **null**) {

missionTripItem.setTripCity(missionCity.getName());

**if** (missionCity.getMissionCountry() != **null**)

missionTripItem.setTripCountry(missionCity.getMissionCountry().getName());

}

Rationale: the reason why we decided to extract missionTrip.getMissionCity() into a variable rather than a method is the former will reduce the calls to getMissionCity().

1. Replace all of the same fragments of missionTrip.getMissionCity() with missionCity

* Advantages and disadvantages using automated refactoring:
* Advantage: save time and effort once programmers trust it and know how to use it.
* Disadvantage:

1. Refactoring Tool cannot provide an explanation for programmers to see if the way you selected to refactor is appropriate. For example, Arie thought we could extract missionTrip.getMissionCity() into a method; however Daniel put forth a better solution to this bad smell, that is, to extract the function call into a variable. This way will not only improve the code readability but also reduce the the calls to getMissionCity().
2. If the codes were modified automatically across different classes, it would be hard to trace them and document which parts of code fragments were changed.
3. It takes some time to trial and error if a programmer is new to the automated refactoring tool supported by IntelliJ.

**Objective #2:** Perform refactoring operations to remove the smell on the code component without automation.

1. The createTabFieldItem() methods were refactored into the Builder pattern to remove the Feature Envy code smell. An inner class of ItemModelMapper is made responsible for creating a TabFieldItem according to the rules specified by the createTabFieldItem() header comments. In particular, the name and isCustom fields are not optional.
2. The TabFieldItem is merely a data class with no behavior. The logic and requirements for constructing the class can be wholly contained within the TabFieldItemBuilder class. In this case, an object cannot be constructed without specifying the two required fields. The original method would simply return null, which requires run-time verification vs. compile-time verification for the builder.

The manual refactoring was a bit tedious in this case, as it mostly mirrored the accessors within the data model. However, it allows for future logic on a per-variable basis which may apply to object creation that cannot be included in the data class.

Using an automated refactoring limits you to the level of ability that the tool provides. This is ideal for simple refactorings like rename or extraction. A manual refactoring, while more work, allows you the freedom to refactor beyond what tool support provides.