By Friday, November 21st:

Implement feature 1. Will read all students from a text file and put them into a user specified group size. The classes of GroupManager and Group will be implemented, and the students will be placed into a Group object, in which GroupManager will have the relevant methods to change the group according to the user's specifications.

-Implement a basic version of group generating system with limited functionality. Focused on essential features only. These features include the following:
-Implement a GUI to let the "instructor" add project-specific information such as course name, group size, project deadline, and project description. Instructor in this case will just be the user running the GUI. (Features 2,3,4,5)
-Implement group system where added info is set to respective class objects. Implement the functionality to get this info as well. (Features 2,3,4,5)
-Implement an algorithm to generate students into groups of given size. This will be a basic strategy which iterates through all students assigning them to groups of given size. Any remaining students end up in a smaller group. (Feature 1)
-In this iteration, for simplicity, there is no GUI for output, standard output is used instead. Similarly, GUI testing is kept separate from group generator testing for simplification.

- - - - - - - - - - -
Generating groups (show ability for different algorithms)
    -look up & implement different search methods
    -ParameterSpec should have a "matches" method to compare Spec's

Add group parameters (like description, deadline, size, courseName)

Add student queries (like schedules, preferences, experience)
Respond to queries
  -Implement singleton class ParameterCollection to hold single instance of GroupParameter[*]
  -New class super class User with subclasses student and instructor (name, number)
  -List<Map<String,List<String>>> maps = new ArrayList<Map<String,List<String>>>();//This is the final list you need
    Map<String, List<String>> map1 = new HashMap<String, List<String>>();//This is one instance of the map you want to store in the above map
  -Change map to EnumMap, with Enum class having "Name,Query,Response"

```
        public enum Param{ NAME, QUERY, RESPONSE; }
        EnumMap<Param, String> map = new EnumMap<Param, String>(Param.class);
```

```
//Singleton collection of instructor-added parameters (questions)
public class ParameterCollection {
   private static ParameterCollection singleton = new ParameterCollection( );
   // A private Constructor prevents any other class from instantiating.
   private Singleton(){ }
   // Static 'instance' method
   public static ParameterCollection getInstance( ) { return singleton; }

  /* Other methods protected by singleton-ness */
   protected static void demoMethod( ) {
      System.out.println("demoMethod for singleton");
   }
}
```

-Implement a class GroupParameter that handles instructor-added queries for student info via a class ParameterSpec. This class will have a map with String keys like Name, Query, Response, and values most probably being strings.

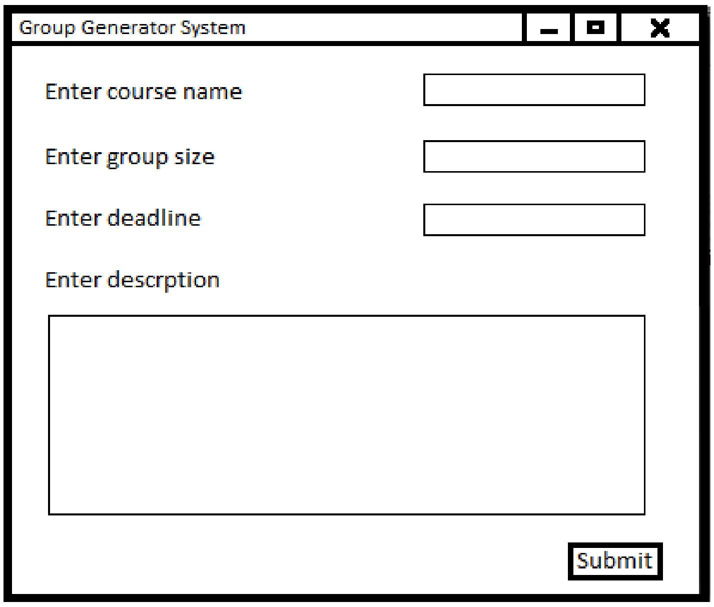
Ex. Name : Group Preferences

Query : Please indicate which 3 students you would like to have in your group.

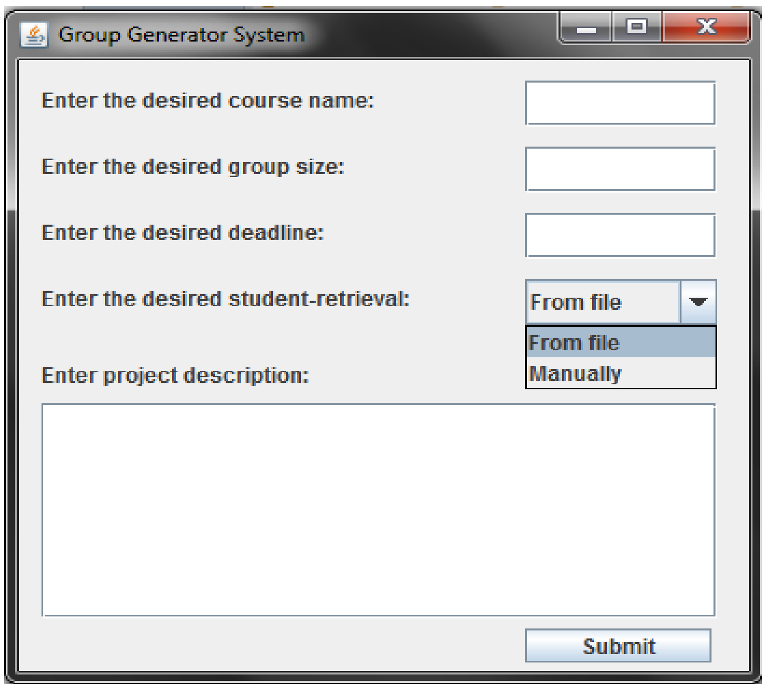
Response : Brett McLaughlin, Gary Pollice, David West

User class. Only the instructor will be allowed to add or remove group parameters, whereas students will only be able to modify the response in the map.

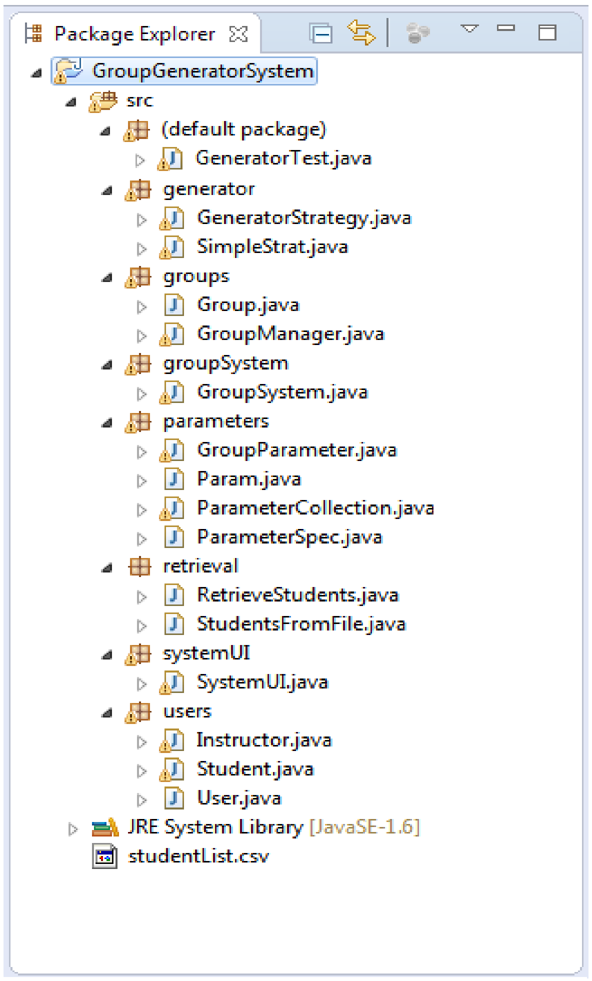
We also now have a collection of group parameters in the system class which is delegated through both the Student and Instructor classes from the GroupParameter class. This collection will be used to manage the modifications by both students and the instructor. potential for different group-generating algorithms.



SystemGUI - Concept

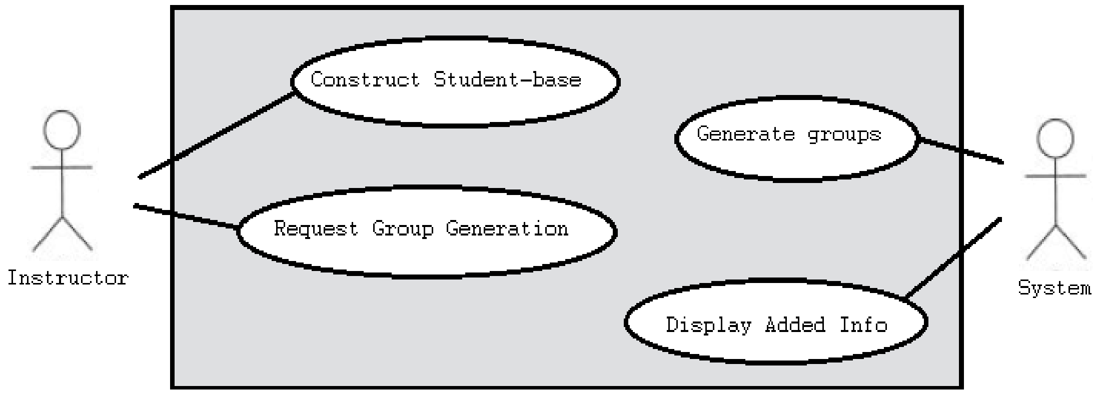

Group Generator System
Enter the desired course name:
Enter the desired group size:
Enter the desired deadline:
Enter the desired student-retrieval:
From file
From file
Manually
Enter project description:
Submit

SystemGUI – Actual

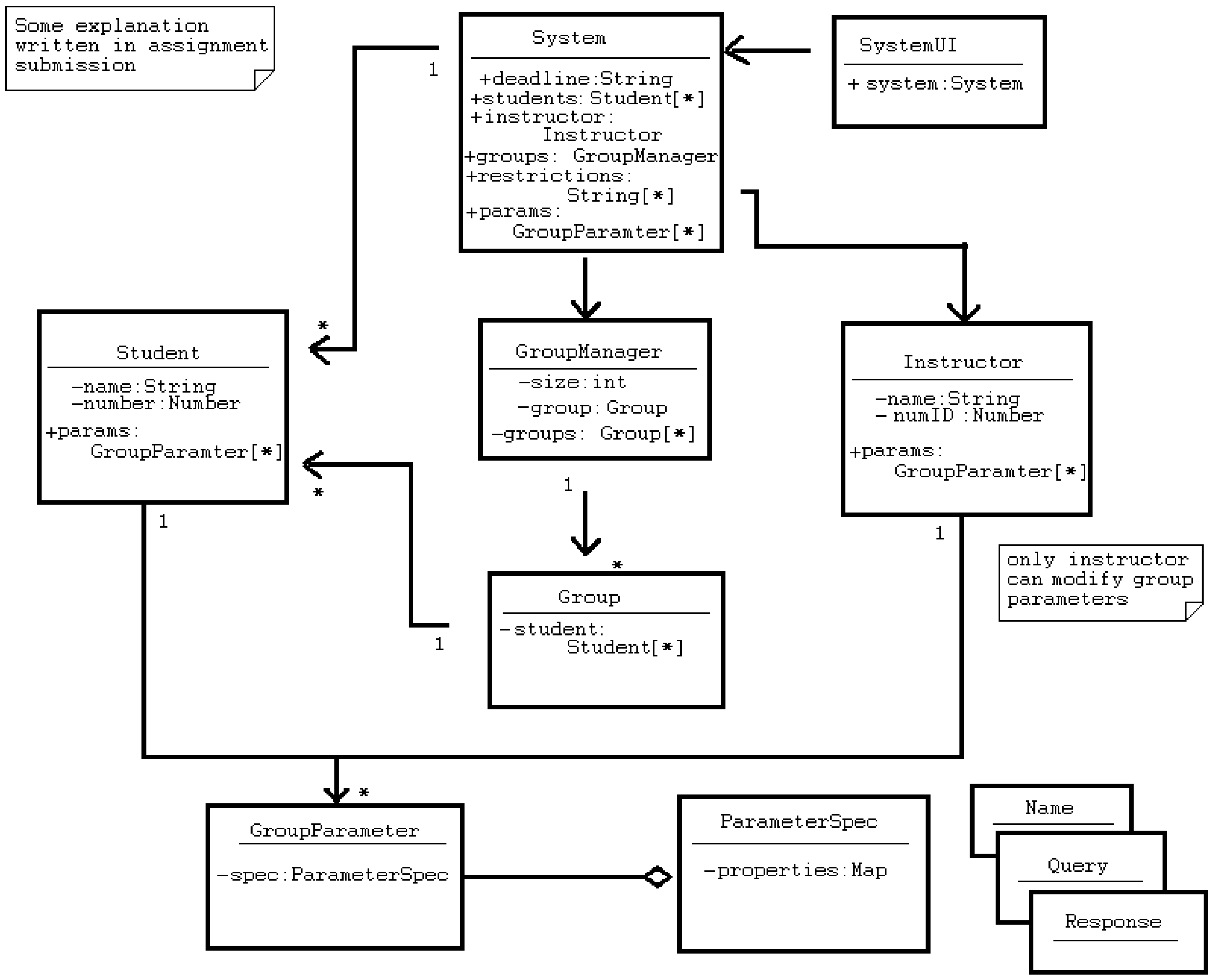


```
Package Explorer
◢ GroupGeneratorSystem
  ◢ src
    ◢ (default package)
      ▷ GeneratorTest.java
    ◢ generator
      ▷ GeneratorStrategy.java
      ▷ SimpleStrat.java
    ◢ groups
      ▷ Group.java
      ▷ GroupManager.java
    ◢ groupSystem
      ▷ GroupSystem.java
    ◢ parameters
      ▷ GroupParameter.java
      ▷ Param.java
      ▷ ParameterCollection.java
      ▷ ParameterSpec.java
    ◢ retrieval
      ▷ RetrieveStudents.java
      ▷ StudentsFromFile.java
    ◢ systemUI
      ▷ SystemUI.java
    ◢ users
      ▷ Instructor.java
      ▷ Student.java
      ▷ User.java
  ▷ JRE System Library [JavaSE-1.6]
    studentList.csv
```

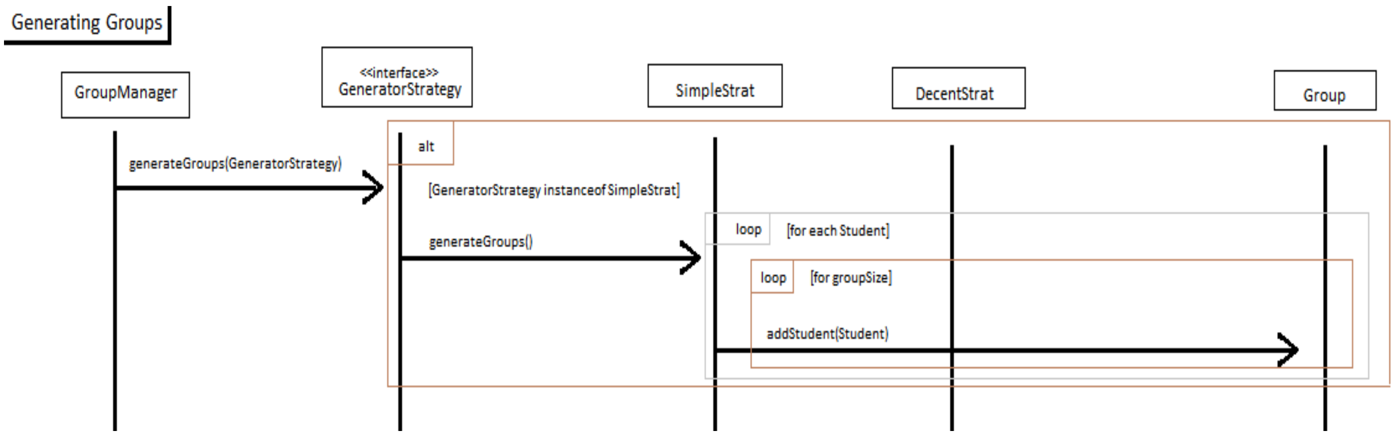
Modules and Classes



Rough Use Case Diagram

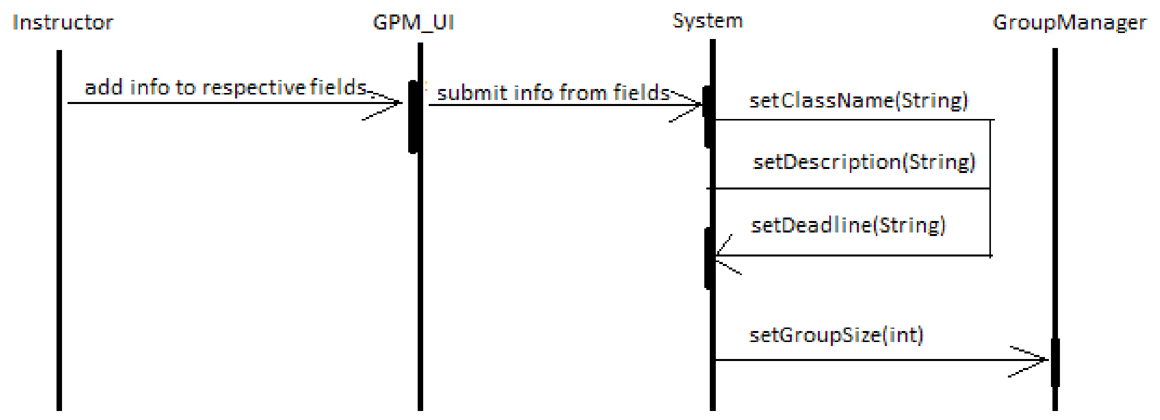


Domain Model



Sequence Diagram – Generating Groups

Add Group Parameters



Sequence Diagram – Add Group Parameters