Labsheet1: The World-of-Zuul

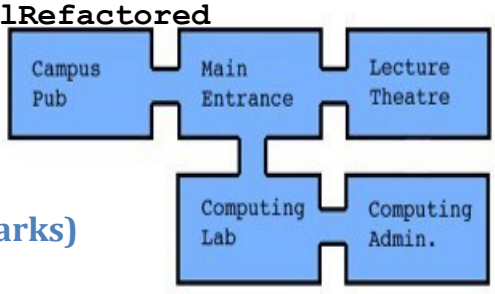
Setter: Idris Skloul Ibrahim

Attempt Task 0A, and 1A in the tutorial and finish Task 2B in the lab. Don't forget to commit your code before you leave the lab! (5 marks)

Task 0A: JUnit test (0 Mark)

Download and import attached code (ExamplesMinimumNumber.java) into your java workspace then,
- Run the code
- Have a look and try to understand it (what it does)?
- Use the Junit attached (MaxNumberTest.java) to test the following two methods in ExamplesMinimumNumber.java:
- *public static int* ***MaxNumber*** *(int num1, int num2)*
- *public static double* ***MaxNumber*** *(double num1, double num2)*

For the rest Tasks please use the code in **`lab1/zuulRefactored`** for the following exercises.



Task 1A: Back Command and JUnit test (2 Marks)

1. Add a *back* Command

- This command does not have a second word. Entering the back command takes the player into the previous room he/she was in.

- You should account for the cases where the user accidentally entered a second word and also if there is no previous room.

2. Test your new command using a pre-defined JUnit test.

- Uncomment the last two lines in **`TestBack.java`** and try to run it.

- For now you have change the visibility of the **`goBack()`** method to **`protected`**. (NB: Later on you'll learn a way to work around this.)

- Describe how the JUnit code works by adding comments above each line of code.

Task 1B: Refactoring Code (3 Marks)

1. Add a Player Class

- Refactor your project to introduce a **`Player`** class. A Player object should store (at least) the current room of the player and the player's name.

- Add the Player to the Game:
  - You need to make sure that the Player is initialized correctly and set into the first Room.
  - Modify the **`printWelcome()`** method that to inform the Player about his current room.
  - Modify the **`goRoom()`** method. Make sure you'll notify the player about his moves.

2. Collecting Items

- Implement an extension that allows a player to pick up a single item. This includes implementing two new commands: **`pickUp`** and **`drop`**.

- You might also want to add more items to different rooms.

- Extends your implementation to allow your player to carry any number of items. Think ahead: how can you make this extension modular? (low coupling).

- Add a restriction that allows the player to carry items only up to a specified maximum weight.

- Implement an *items* command that prints out all items currently carried and their total weight.

3. Write your own JUnit test

- Create a new JUnit test "TestPerson.java" in Eclipse (File -> New -> Junit -> Test Case), or (File -> New -> other -> Java ->Junit ->TestCase).

- Create tests for your **`pickUp()`** and **`drop()`** methods.

References:

- Lars Vogel. JUnit - Tutorial. Version 2.3, 2012. http://www.vogella.com/articles/JUnit/article.html
- https://www.youtube.com/watch?v=v2F49zLLj-8
- https://github.com/kentbeck/junit/wiki