COSC 111 Project Overview

In addition to lab assignments, you will also work on a project of building a simple computer game. Guidance will be provided by both the instructor and TAs. You may work on the project during the lab time or on your own time. You will be required to write small pieces of code that completes the game. The deadline for these pieces, named P1, P2, etc., are shown in the schedule below.

The aim of the project is to see how small pieces of code similar to the ones you write in the lab assignments may fit into a larger program.

The game you are going to work on features a simple quiz competition in which up to three contestants are presented with questions from different categories of knowledge, and they must answer them to gain score ($). You may think of it as a modified version of Jeopardy, the famous TV game show. Watch this video clip 111_FullGameSampleRun.mp4 to get an idea of how the complete game will eventually look like.

Starter Java Code

You will be provided with starter code that you should download from Connect and import into your Eclipse project. The starter code includes five classes: BDialog.java, Game.java, GameFrame.java, Player.java, and Main.java. **You should only work on (i.e., modify) the code in Main.java**. You will also be provided with two images: background.jpg and MisterX.jpg that must be copied to your project. This video clip 111_HowToImportStarterCode.mp4 shows how to include these files into Eclipse.

Custom Defined Methods

Most of the time you will use standard Java keywords and statements that you learned in class (e.g. variable declaration, while loops, if statements, etc.). However, the starter code involves some custom 'methods' that help you achieve the desired objectives.

In general, a method is a collection of statements that are grouped together to perform a specific function. For example, `System.out.println()` is a standard Java method that when called will cause the system to execute several statements in order to print a message on the console (screen).

You will learn more about how to create your own methods later in this course, but for now you only need to use the provided methods as is. A list of the methods that are defined in the starter code is given in the table below. You must read this table carefully to understand how they methods could be used. You should seek help from the instructor or the TA in case you have any questions about them (or of course about the project in general).

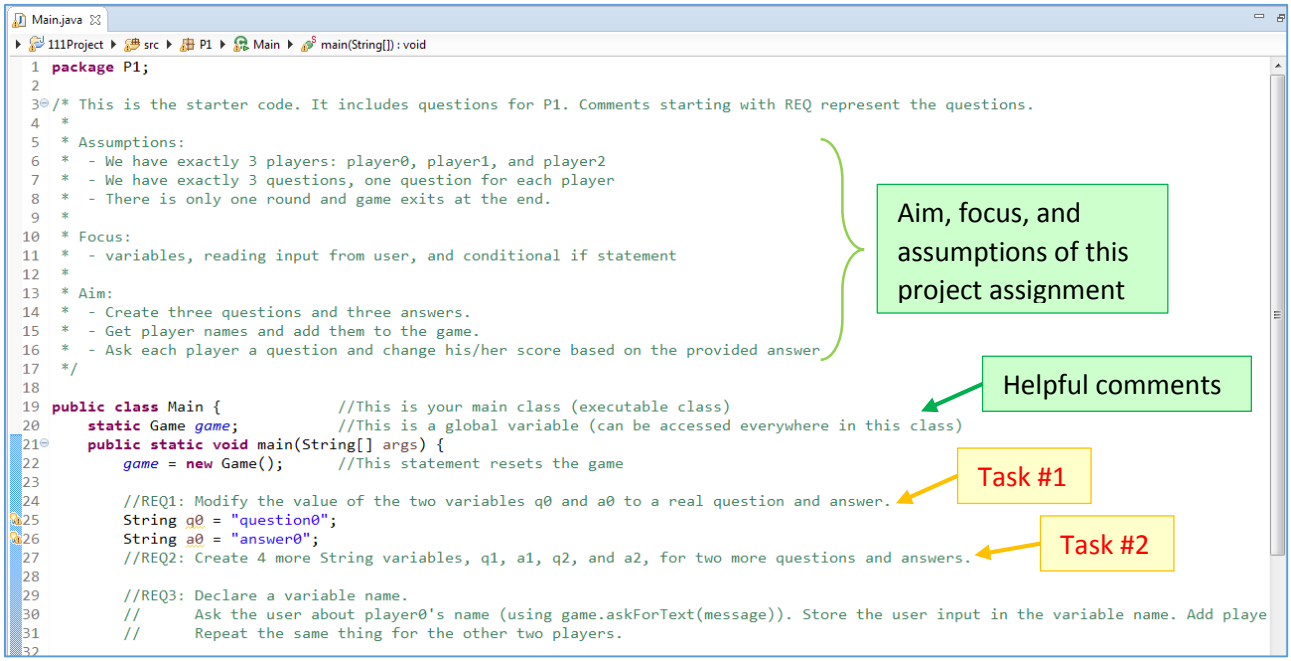
`game.askForText(String msg)`	**Aim**
	To read input text from user.
	Description
	This method displays the given message ***`msg`*** to the user, waits for the input, and then returns the text the user enters in the text box. This method is similar to the `Scanner`'s `nextLine` method.
	Example
	`String x = game.askForText("what's your name? ");`
	`System.out.println("Hello " + x);`

`game.askForInt(String msg,int a,int b)`	**Aim**
	To read input number (integer) from user.
	Description
	This method displays the given message ***`msg`*** to the user in addition to valid range for the required number, waits for the input, and then returns the integer the user enters in the text box. If the user enters invalid input (e.g., a number outside the ***`a`*** to ***`b`*** range, a non-numeric value, or `null`), the method will display an error message and ask the user to enter a valid input, and then repeats the same process abve. This method is similar to the `Scanner`'s `nextInt` method.
	Example
	`String n=game.askForInt("How old are you?",0,99);`
	`System.out.println("You are " + n + " years old");`

`game.addPlayer(String name)`	**Aim** To add a player to the contest. **Description** This method takes a player's name as add him/her to the game, displaying an image that represent the player on the game interface. **Example** `game.addPlayer("Lili");  //adds Lili to the game`
`game.setCurrentPlayer(int id)`	**Aim** To set the current player who is to be asked next. **Description** This method causes the game to draw a rectangle around the current player, writes "*playerName*'s turn" on the top of the game window, and then increments the player's score if s/he answered correctly. You will not need to use this method as it is already written for you in the correct location in the `Main.java` file.
`game.correct()`	**Aim** To do subsequent actions if the user correctly answers a question **Description** This method displays "Correct" on the game window, increments the *current player*'s score, and change his/her frame color to green. You should call this method only if the player answers a question correctly. **Example** `game.correct();`
`game.incorrect()`	**Aim** To do subsequent actions if the user answers a question incorrectly **Description** This method displays "Sorry, incorrect answer" on the game window and change frame color of the *current player* to red. You should call this method only if the player answers a question incorrectly. **Example** `game.incorrect();`

Required Tasks

With every new part of the project, you will be required to do few tasks that involve adding extra code or changing existing one. The required tasks are written as comments within the `Main.java` class (see figure below). These comments are placed at the specific locations that need to be modified. Note that the task comments always start with the word **REQ** followed by the task number (e.g., REQ1, REQ2, etc.). Other comments are placed to help you read and better understand the code. A **video clip** will be provided with the expected outcome after performing the required tasks.




```
1  package P1;
2
3  /* This is the starter code. It includes questions for P1. Comments starting with REQ represent the questions.
4   *
5   * Assumptions:
6   *  - We have exactly 3 players: player0, player1, and player2
7   *  - We have exactly 3 questions, one question for each player
8   *  - There is only one round and game exits at the end.
9   *
10  * Focus:
11  *  - variables, reading input from user, and conditional if statement
12  *
13  * Aim:
14  *  - Create three questions and three answers.
15  *  - Get player names and add them to the game.
16  *  - Ask each player a question and change his/her score based on the provided answer
17  */
18
19 public class Main {             //This is your main class (executable class)
20     static Game game;            //This is a global variable (can be accessed everywhere in this class)
21     public static void main(String[] args) {
22         game = new Game();       //This statement resets the game
23
24         //REQ1: Modify the value of the two variables q0 and a0 to a real question and answer.
25         String q0 = "question0";
26         String a0 = "answer0";
27         //REQ2: Create 4 more String variables, q1, a1, q2, and a2, for two more questions and answers.
28
29         //REQ3: Declare a variable name.
30         //      Ask the user about player0's name (using game.askForText(message)). Store the user input in the variable name. Add playe
31         //      Repeat the same thing for the other two players.
32
```

There will be **no other documentation** provided for each project part. At the time you should start working on a new part (**according to the course syllabus**), you should download the **updated starter code** and perform the required tasks as explained by the comments. The updated code includes the solution to the previous project part (i.e., P2 includes the solution of P1 + required tasks for P2). While it is recommended to use the provided solution, you can also use your own solution from the previous part.

Here is a video clip to show you how to import the updated starter code into your project (111_HowToImportUpdatedStarterCode.mp4).

Submission Instruction

For submitting your solutions for the Project, upload to Connect the zipped "src" folder for the project part you are working on, e.g., P1, or at least upload the zipped Main.java file for that part. Name your zipped file as studentID_PartNumber.

You should follow the same process you used for submitting the assignments: click on the *PartNumber* link on Connect (e.g. P1 link), click "Browse My Computer", select your zipped file, click "Open", and then click "Submit".