# Assignment 2: Reaction Game

## Overview

Welcome to game programming!  We will start with a visual reaction/observation game, of course with the help of SimpleGUI. The game description:

Show two faces on the screen, each out of a collection of 6 possibilities:

- 3 ellipse faces:  smiling, serious, frowning
- 3 box faces:  smiling, serious, frowning.

The user has to decide as fast as possible, if the two faces on the screen have the same facial expression (smiling, serious, frowning), independent of face color and shape; he/she then types "e" or "u" on the keyboard - "e" for equal, "u" for unequal (these keys go easy with left and right hand). If the choice was correct, the time between face drawing and key-input is measured. We do this ten times, and average the time. A score will be given depending on average reaction time. If the user makes a wrong choice, the game is over immediately.

What will you learn in this assignment?

- inheritance
- writing a program with a slightly more complex program flow than your usual one method assignments
- Creating a program from a workflow description
- thinking about program flow. Is it the best way to write such a program in a sequential way? What would happen if we want to use 2 buttons instead?
- the experience that writing games is fun

## Your Task in Steps:

### 1. Preparation

Your starting point is the in-class project "Faces2" (see website, sources). You MUST use this project as your starting point --- it will ease your task immensely.

From EllipseFace, you inherit 3 classes:

- SmilingEllipseFace
- SeriousEllipseFace
- FrowningEllipseFace

From BoxFace, you inherit 3 classes:

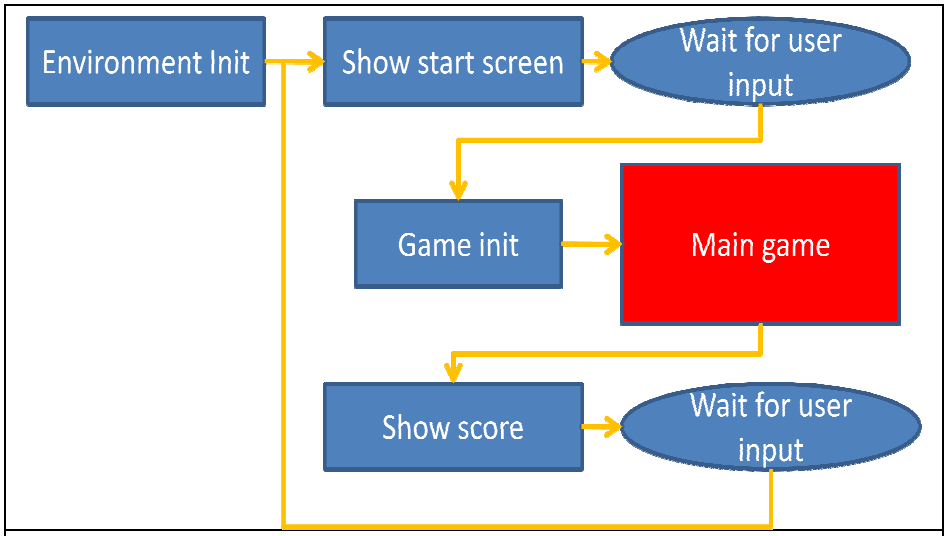
- SmilingBoxFace
- SeriousBoxFace
- FrowningBoxFace

The serious versions are already done, they are the default BoxFace or EllipseFace expression. However, just for the consistency of naming, create them (and don't implement any methods in there --- this is a usual approach to make a project more readable).

Think: which methods do you need to overwrite in your classes? (Hint: you want to change the facial expression!)

## 2. Workflow, the Framing the Program

Now create the game! Here is the workflow:



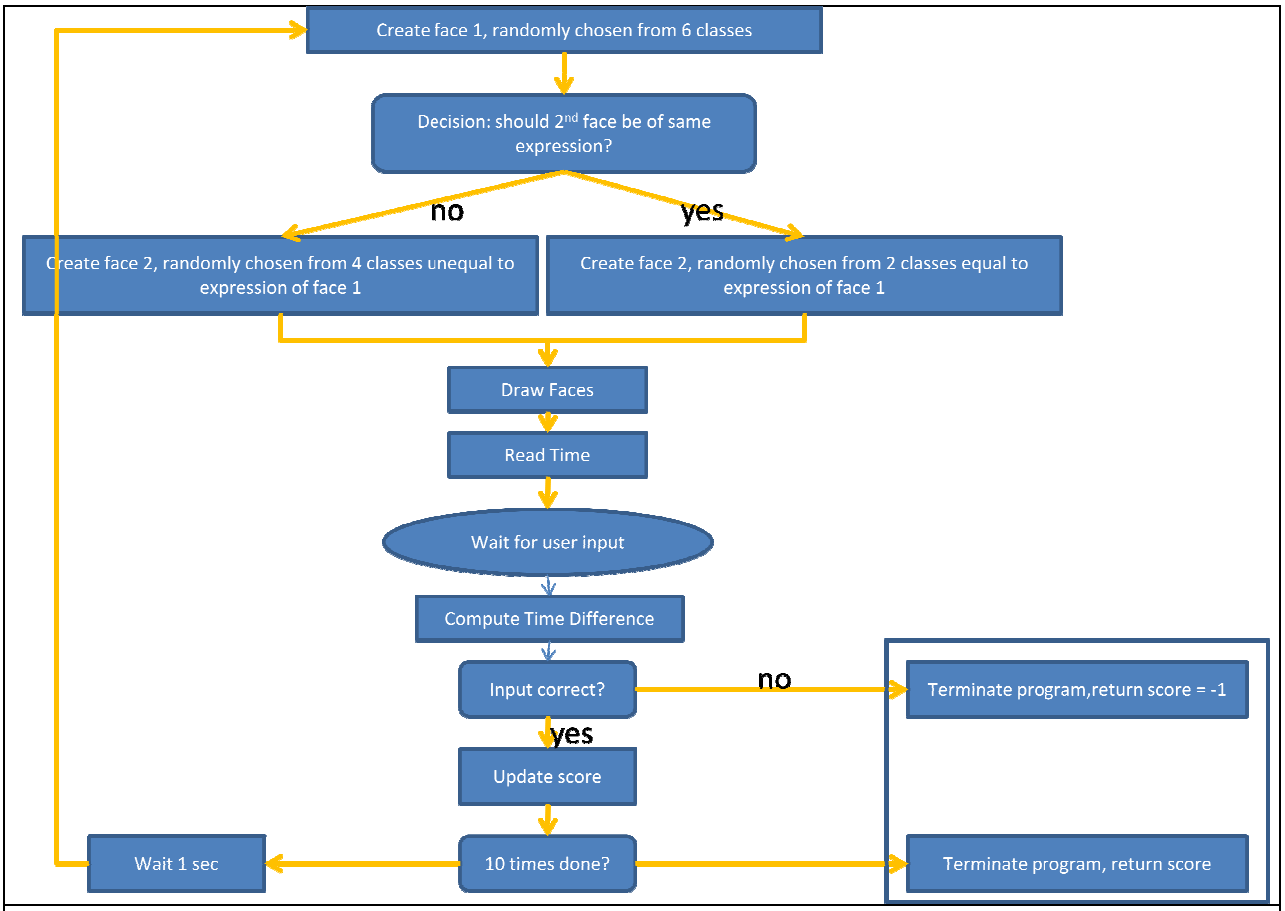

Game Workflow. Each of these elements should be either one line of code, or a single method call in your main routine!

Please don't panic. The image above translates into about 10 lines of code (e.g. "show start screen" => show an image; Wait for user input = waitForKey etc.).

The main game should be implemented in a method "mainGame(.)", so, for now that's also just a single line of code (plus a click in netbeans to create the method declaration for you).

## 3. Main Game Implementation

Now, after all the dust around the main game has settled, you can start writing the main game:




Detailed flow of the main game

## Hints:

- The design apart from the workflow is on you. Convince the TA and me with great start and end screens, Implement more beautiful faces if you want to. Bonus points for beauty!
- you are NOT allowed to change any of the given classes in Faces2! All your face-implementations MUST be done in your own, inherited classes! Just imagine i would have given you the Face2 project as a jar-file, not as source-code.
- Some details are left to you. For example, a program flow would typically not tell you to erase the previous faces before you draw new ones. Details like these are on you. Always.
- Time assessment in JAVA: the call System.currentTimeMillis() returns a "long" variable, with time passed since some start-event. Time difference computation:
    - long time1 = System.currentTimeMillis();
    - ....<program>...
    - long timeDifference = System.currentTimeMillis() - time1

- Random decisions: Math.random() creates a random double between 0 and 1 (1 excluded). Think how to use the random number generator in the facial expression decision. This influences the playability a lot!
- you MUST use the six inherited classes. no if-statements in the graphics part! Only in the non-graphics related game-code!
- ALWAYS separate between graphic and non-graphic related code. ALWAYS. Your output should just be modules, that can be exchanged. E.g., if someone wants to write a 3D version of this game, just the face-classes would have to be exchanged, nothing else.
- For the keyboard - input, use SimpleGUI's "keyReadChar()" method, which returns immediately after a single key was hit, no <enter> required.
- Pausing the program for a second: SimpleGUI, method pauseProgram(time in milliseconds);
- This is a one week assignment. Use the workflow and translate it to code!