Bonus Assignment | Robot Problem

Objectives

The goal of assignment is to practice 4 types of design patterns: **Factory**, **Builder**, **Bridge** and **Singleton**. You will be asked to complete missed functions to produce certain output. By filling these missed functions you can figure out the usage of these design patterns.

Submission

- Submit the **completed code**: it should run on the given test case.
- Document the code. Generate and deliver **Javadoc for all classes** and methods.
- **Write a report** on how your used the design patterns in the code and the **role** of each one in solving the corresponding problem.
- This is an **individual** assignment and should be solved individually. Any cheating will be punished.
- **Eng. Youmna** will be happy to help you with any difficulty in the assignment and she will be discussing it with you and grading it.
- Submit your code **on Acadox.** Name your zip file (code and report) CS352-A1-ID.zip

Problem Statement

Given the robot navigation problem that presented before. The company that asked for the robot controller asked for new requirements. As the robot tries to navigate through room with the possibility to get into collision with a human being, they then want the robot to interact with humans that prevent it from completing its navigation. The interaction should be via social cues, i.e. emotional expressions. Robot designer therefore decided to make two types of interactions: via sound and via facial expressions. There will be 3 types of sound and facial expressions:

> **Happy:** when human responses positively to robot request
> **Sad** or **Angry:** when human responses negatively to robot request
> **Wondering:** when robot asks human for a favor

For sound to be produced it takes the text which to be said by the robot and take the type of rhythm to specify the tone of the sound. For Facial Expression it should be expressed in a Robot Face with the following components: 2 Eyebrows, 1 lip, and 2 Eyes. This robot face could be implemented either by graphics "on screen" or in actual hardware components.

Bonus Assignment | Robot Problem

Problems & Suggested Solutions with DPs

Problem 1: Sound to be produced frequently and with different parameters.
Solution: Factory design problem.

Problem 2: Facial Expression to be built via complex integration of face components
Solution: Builder design problem.

Problem 3: different implementation of each facial components.
Solution: Separate abstraction definition of components from actual implementation (Bridge)

Problem 4: one instance of face should be active at the runtime
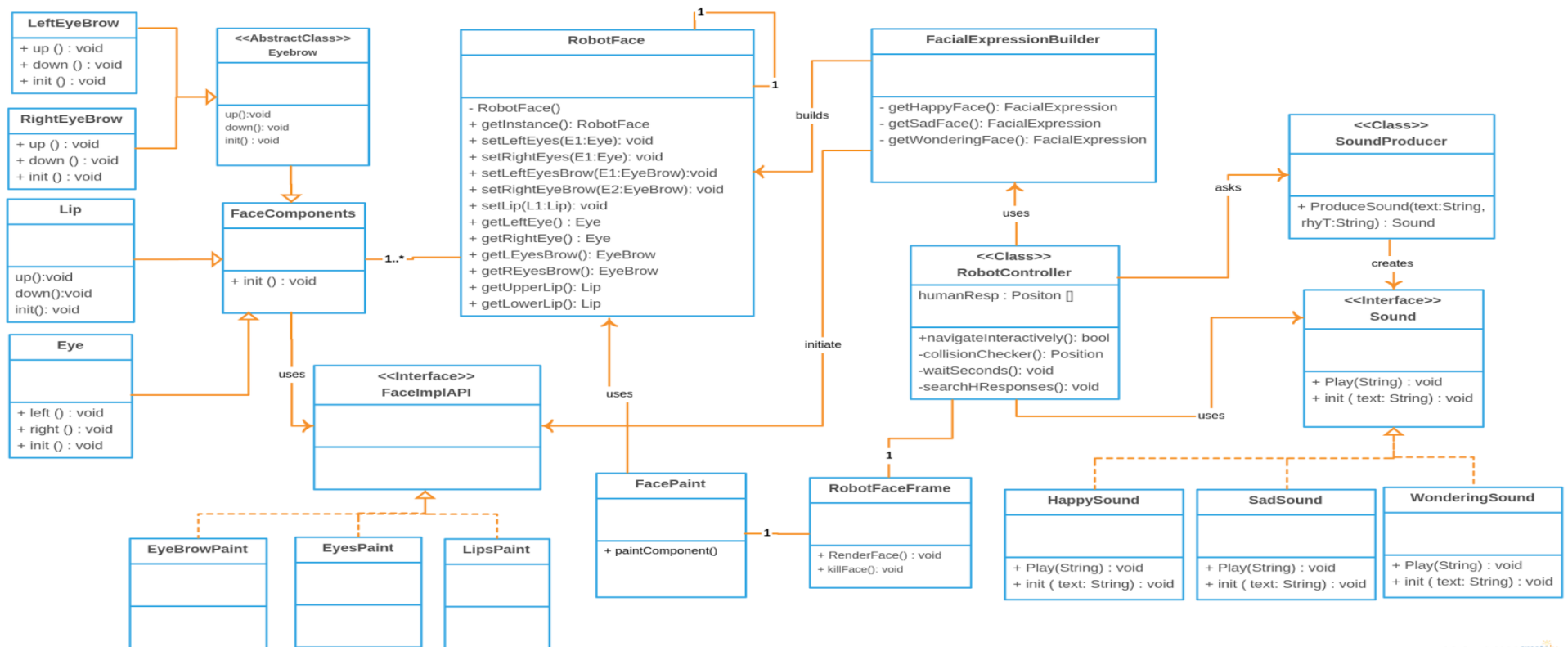Solution: Singleton design pattern.

Bonus Assignment | Robot Problem

Class Diagram

LeftEyeBrow
+ up () : void
+ down () : void
+ init () : void

RightEyeBrow
+ up () : void
+ down () : void
+ init () : void

Lip
up():void
down():void
init(): void

Eye
+ left () : void
+ right () : void
+ init () : void

<<AbstractClass>>
Eyebrow
up():void
down(): void
init() : void

FaceComponents
+ init () : void

RobotFace
- RobotFace()
+ getInstance(): RobotFace
+ setLeftEyes(E1:Eye): void
+ setRightEyes(E1:Eye): void
+ setLeftEyesBrow(E1:EyeBrow):void
+ setRightEyeBrow(E2:EyeBrow): void
+ setLip(L1:Lip): void
+ getLeftEye() : Eye
+ getRightEye () : Eye
+ getLEyesBrow(): EyeBrow
+ getREyesBrow(): EyeBrow
+ getUpperLip(): Lip
+ getLowerLip(): Lip

FacialExpressionBuilder
- getHappyFace(): FacialExpression
- getSadFace(): FacialExpression
- getWonderingFace(): FacialExpression

<<Class>>
SoundProducer
+ ProduceSound(text:String, rhyT:String) : Sound

<<Class>>
RobotController
humanResp : Positon []

+navigateInteractively(): bool
-collisionChecker(): Position
-waitSeconds(): void
-searchHResponses(): void

<<Interface>>
Sound
+ Play(String) : void
+ init (text: String) : void

<<Interface>>
FaceImplAPI

EyeBrowPaint

EyesPaint

LipsPaint

FacePaint
+ paintComponent()

RobotFaceFrame
+ RenderFace() : void
+ killFace(): void

HappySound
+ Play(String) : void
+ init (text: String) : void

SadSound
+ Play(String) : void
+ init (text: String) : void

WonderingSound
+ Play(String) : void
+ init (text: String) : void

builds
uses
asks
creates
uses
uses
initiate
uses
1
1
1..*
1
1

Bonus Assignment | Robot Problem

Implementation Notes

Some simplification: the map only contains human and robot only (no objects)

1. To understand the code start by the **RobotNaviagatePanel**: there is only one object of type robot controller that call a **navigateInteractively()** function.

2. At **RobotController** the function **navigateInteractively():**
    [1] Necessary definition for filling the map and human responses by opening two files:
    → **imap**: map file
    → **imapr**: human responses for **imap** file defined as pair of (position, response)
    → response = 1: human response positively
    → response = 0: human response negatively

    [2] Then there are definitions for **FacialExpressionBuilder** and **SoundProducer**: Try to understand what these initializations do.

    [3] Then the navigation steps, trying to find nextPosition by calling function called **getNextPositionInter()**

3. At **RobotController** class **getNextPositionInter()** function: It does the same of the normal **getNextPosition()** but it should handle the interactive response. This interactive response happens in **collisionChecker()** function *it's not complete*

4. **FacePaint** is responsible of drawing all the facial components by calling the current active face and call **activateComponent()** function on it. "it should attaches a graphics object to them to be able to draw"

5. **RobotFaceFrame** is the frame that contains the drawing area, it initializes FacePaint to holds the graphics on a frame and set it visible to you via activating a frame.

Bonus Assignment | Robot Problem

Task

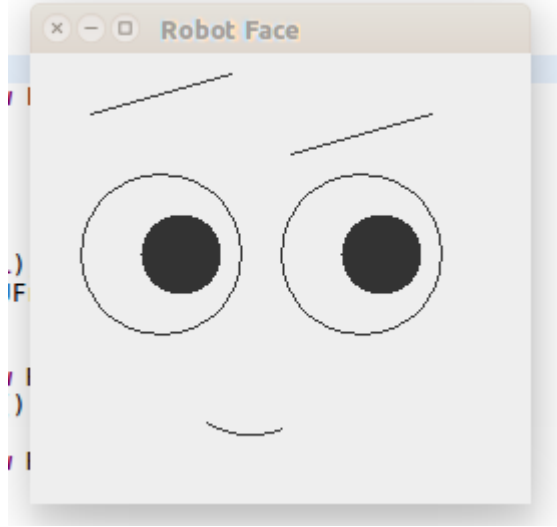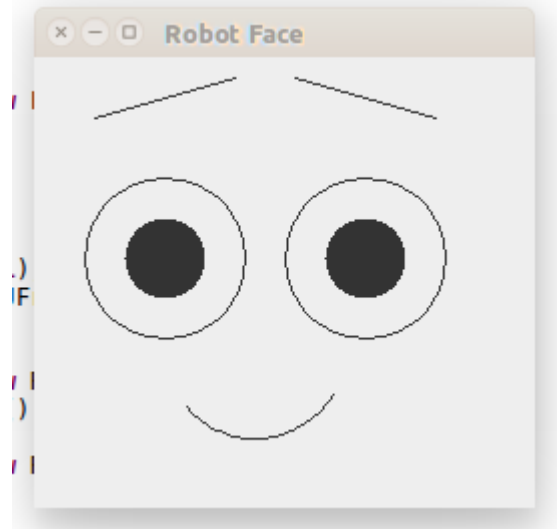Given the code that implemented the previous class diagram:

1. Understand the flow of code. Try to connect it with the class diagram.

2. (**2.5 Points**) In **FaceComponents** you can find an object of type **FaceImpAPI**. This API is the bridge implementer. We want to use it in the **FaceComponents** subclasses

   → check the classes that implements the **FaceImpAPI** for **changeComponentParameters()** function which is responsible for adjust the parameters for the implementer. When calling **changeComponentParameters ("lu")** on **LeftEyebrowPaint** class, it adjusts itself to draw the **LeftEyeBrow** after applying up movement on it.

3. (**1.5 Point**) In **robotController** there is function called **collisionChecker(Position).** This function should return a next position after doing collision check. The missed part is what we want to add the interactive part as when we find collision with human. To achieve that:
   a. Produce sound <use **SoundProducer**>
   b. Build a facial expression <use the **FacialExpressionBuilder** object>
   c. Render the result of facial expression <use **RobotFaceFrame**>

   → you may need to use the **waitSeconds()** function that is defined in **RobotController** to check the result of rendering.
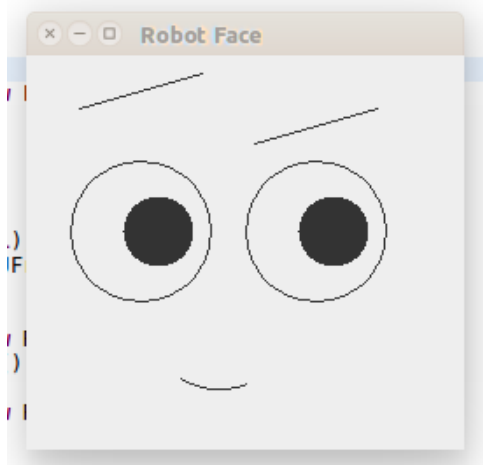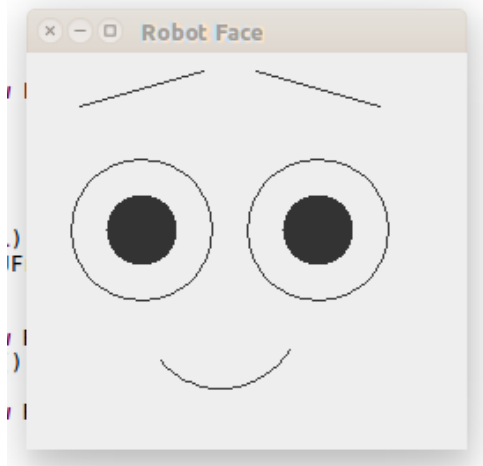
Sample Output

R0	H	H	0	initial map
H	0	0	0	
0	0	0	H	
0	0	0	0	

Bonus Assignment | Robot Problem

Wondering:Pos (0,1) could you please allow me to pass!!!	
Human allows robot to pass	
HAPPY :Pos (0,1)THANK YOU	
R0 R1 H 0 H 0 0 0 0 0 0 H 0 0 0 0	

Bonus Assignment | Robot Problem

Wondering :Pos (0,2) could you please allow me to pass!!!	
Human allows robot to pass	
HAPPY :Pos (0,2)THANK YOU	

R0	R1	R2	0	
H	0	0	0	
0	0	0	H	
0	0	0	0	

R0	R1	R2	R3	
H	0	0	0	
0	0	0	J	
0	0	0	0	

Bonus Assignment | Robot Problem

R0	R1	R2	R3	
H	0	0	R4	
0	0	0	J	
0	0	0	0	

Wondering :Pos (2,3) could you please allow me to pass!!!	
Human refused to move	
SAD :Pos (2,3)ohh I won't able to reach my goal	