

CSC 508/9

Software Engineering (Design & Deployment)

Lecture 03. Blackboard

Dr. Javier Gonzalez-Sanchez

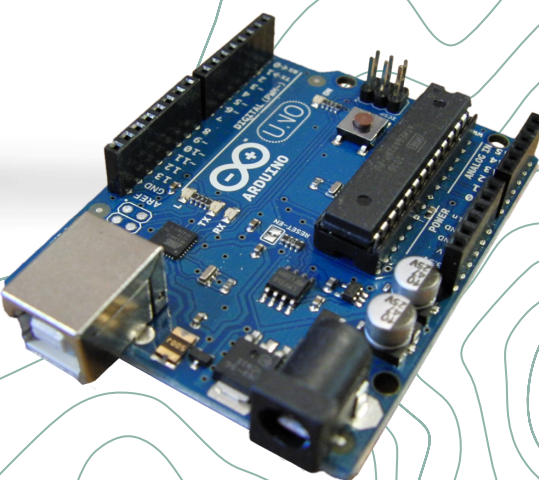
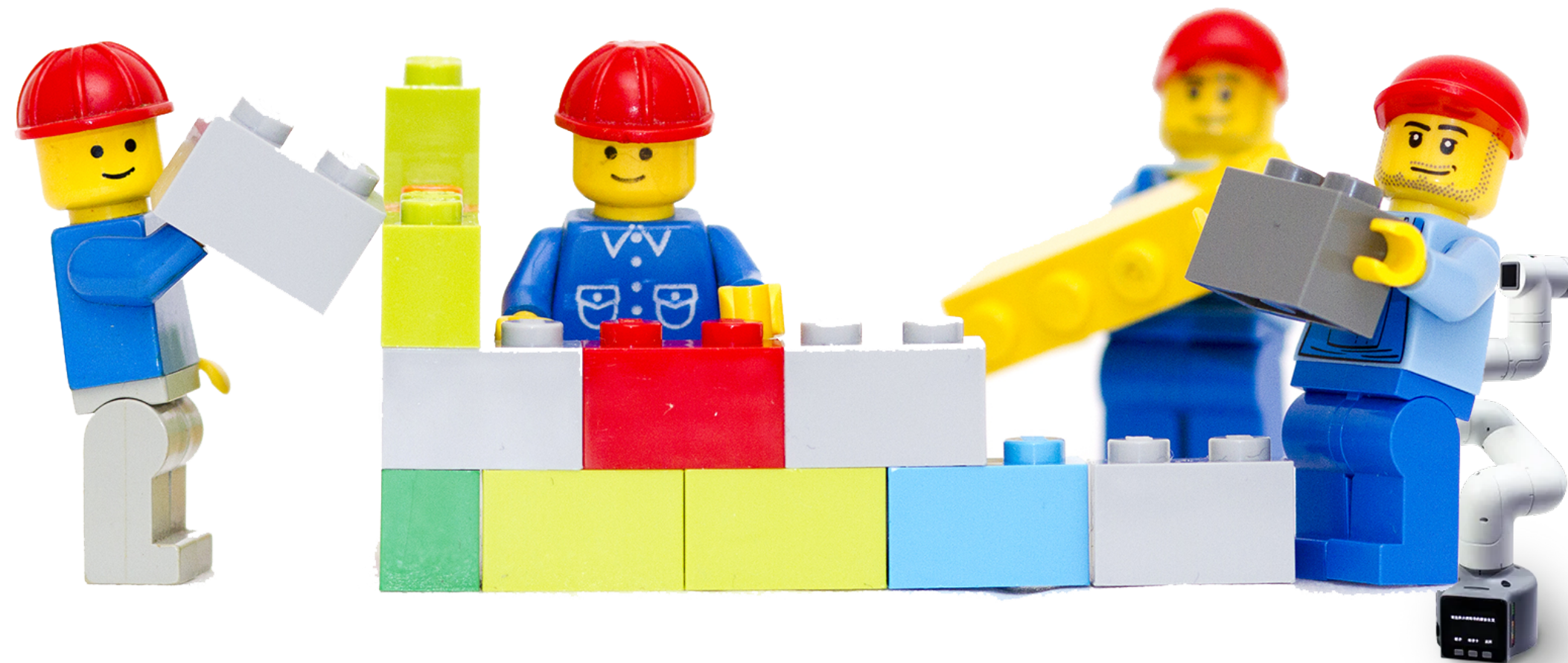
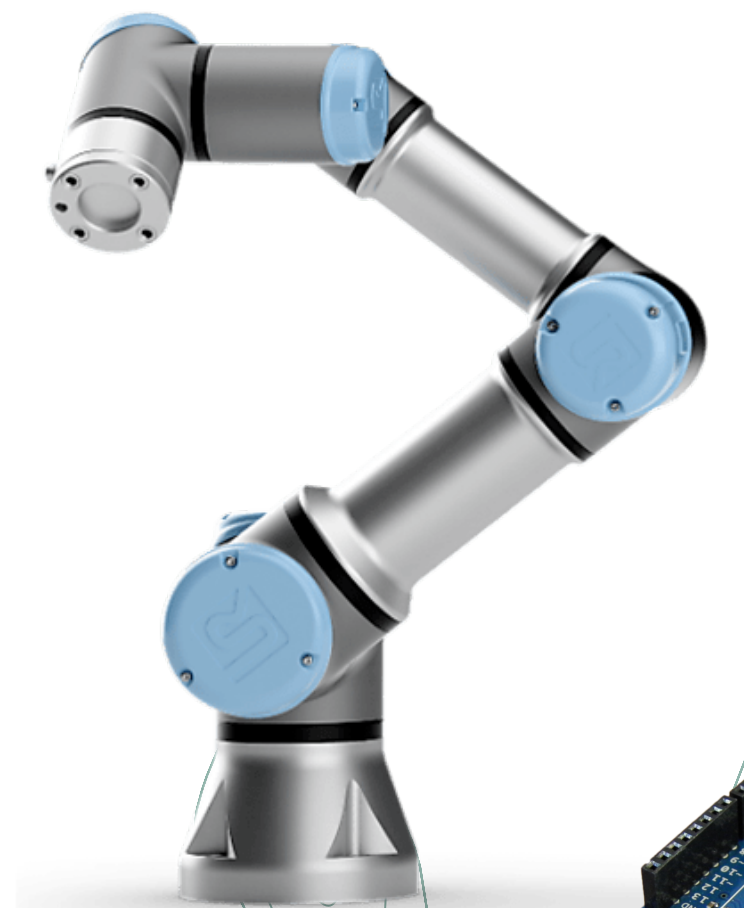
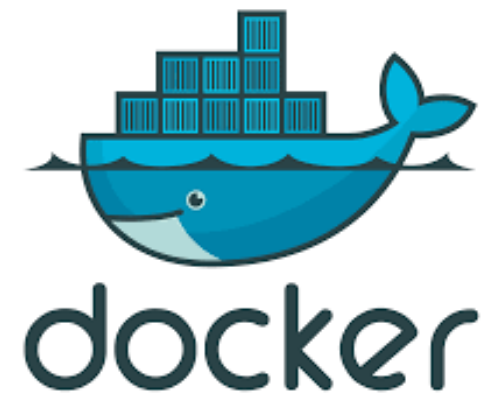
javiergs@calpoly.edu
www.javiergs.info

office: 14 -227



Project Proposal

Environment



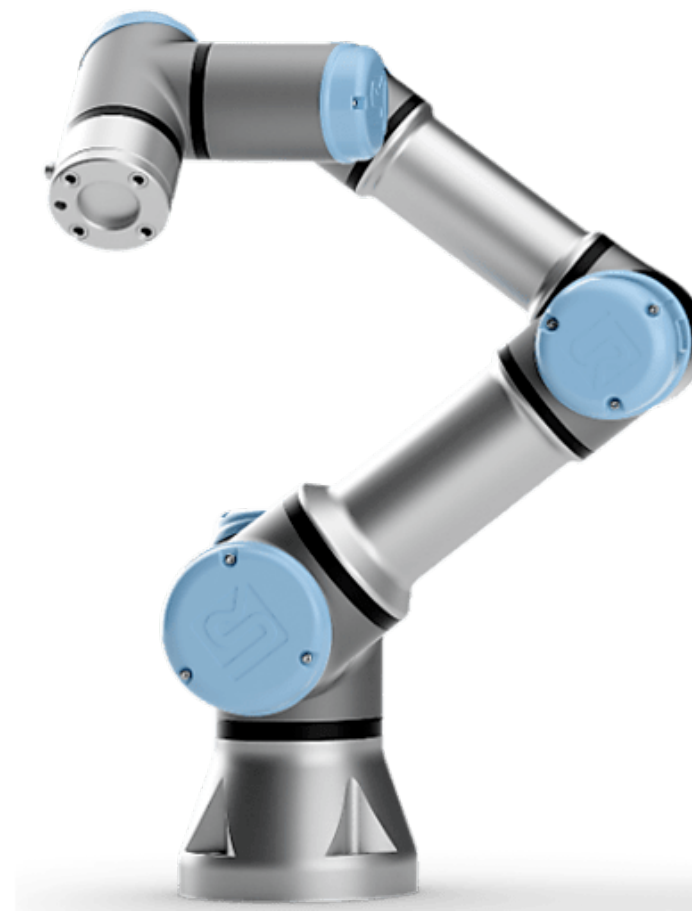
Cobot

- **Universal Robotics e-Series**

<https://www.universal-robots.com/media/1802432/e-series-brochure.pdf>

- **Commands that can be used**

[https://s3-eu-west-1.amazonaws.com/ur-support-site/29983/Script command Examples.pdf](https://s3-eu-west-1.amazonaws.com/ur-support-site/29983/Script_command_Examples.pdf)



Brain-Computer Interfaces

- **Emotiv Headsets**

<https://www.emotiv.com/>

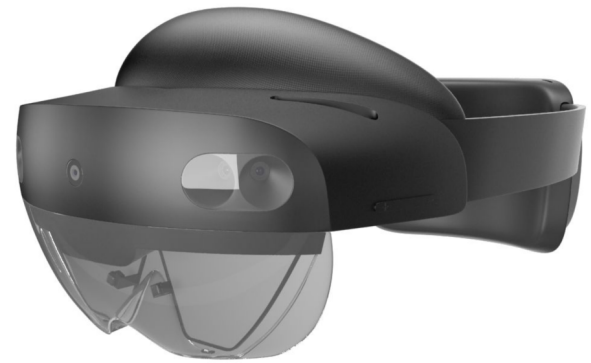
- **Data Samples**

<https://emotiv.gitbook.io/cortex-api/data-subscription/data-sample-object>



- **Unity Development for HoloLens**

<https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/unity-development-overview>



- **Meta Quest**

<https://developers.meta.com/horizon/resources/compare-devices/>



Eye Tracking

- **Tobii Spark**

<https://www.tobii.com/products/eye-trackers/screen-based/tobii-pro-spark>

- **Tobii SDK**

<https://www.tobii.com/products/software/applications-and-developer-kits/tobii-pro-sdk>



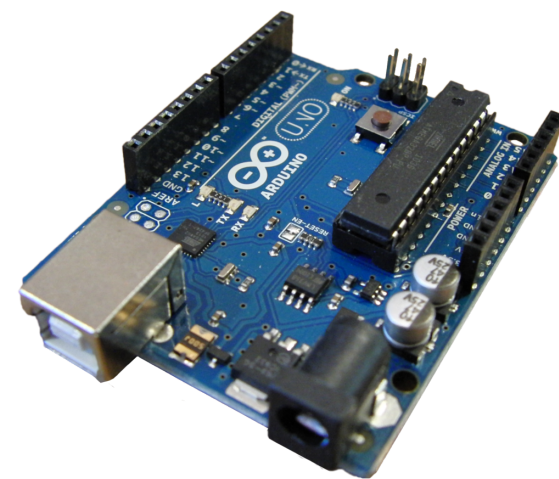
Eye Tracking

- **Petoi**

<https://www.petoi.com/pages/software-apps>

- **Arduino IoT (with MQTT)**

<https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-device/>



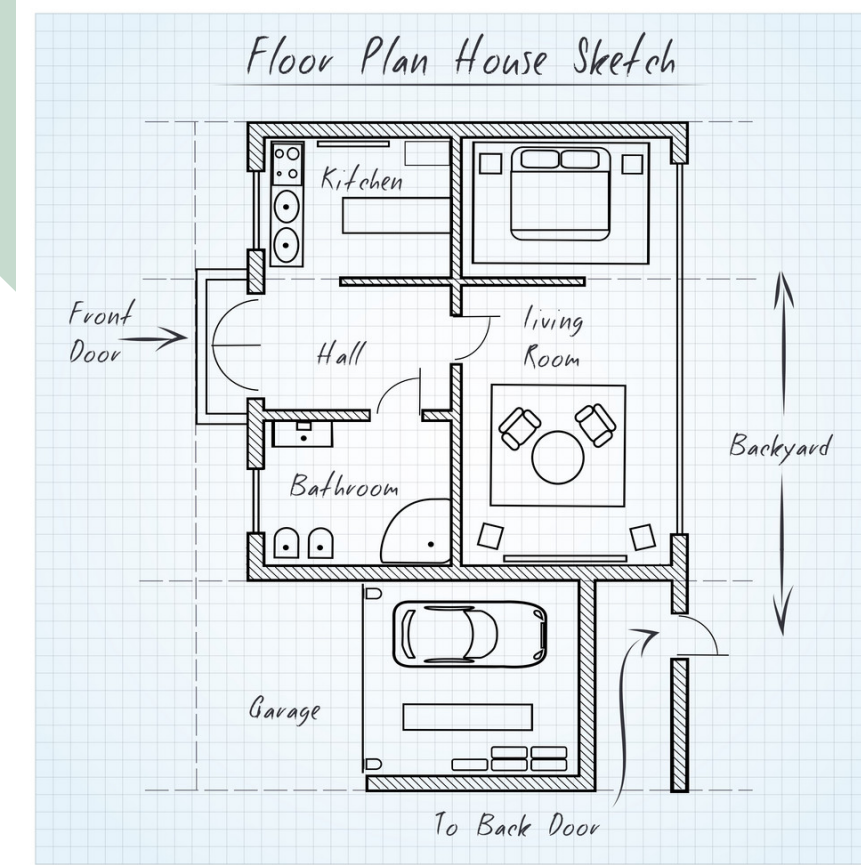
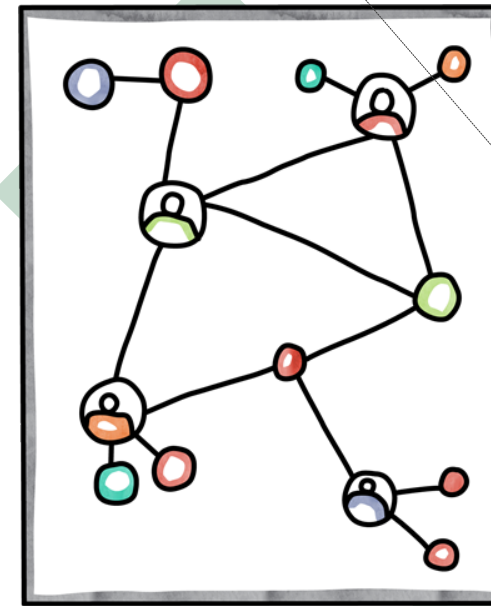
Previously

Blueprint

Architecture

Design

Code

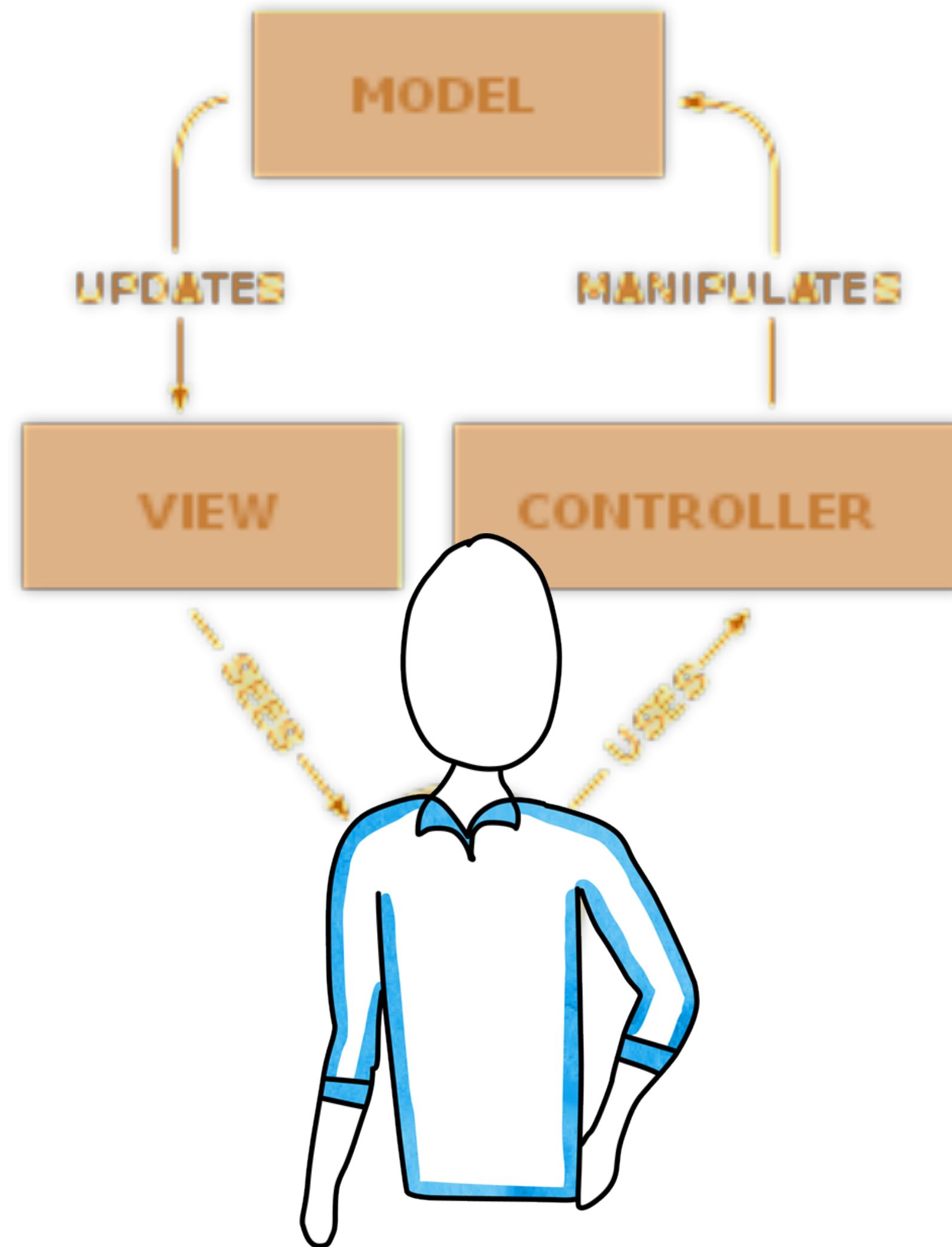


Architecture Patterns

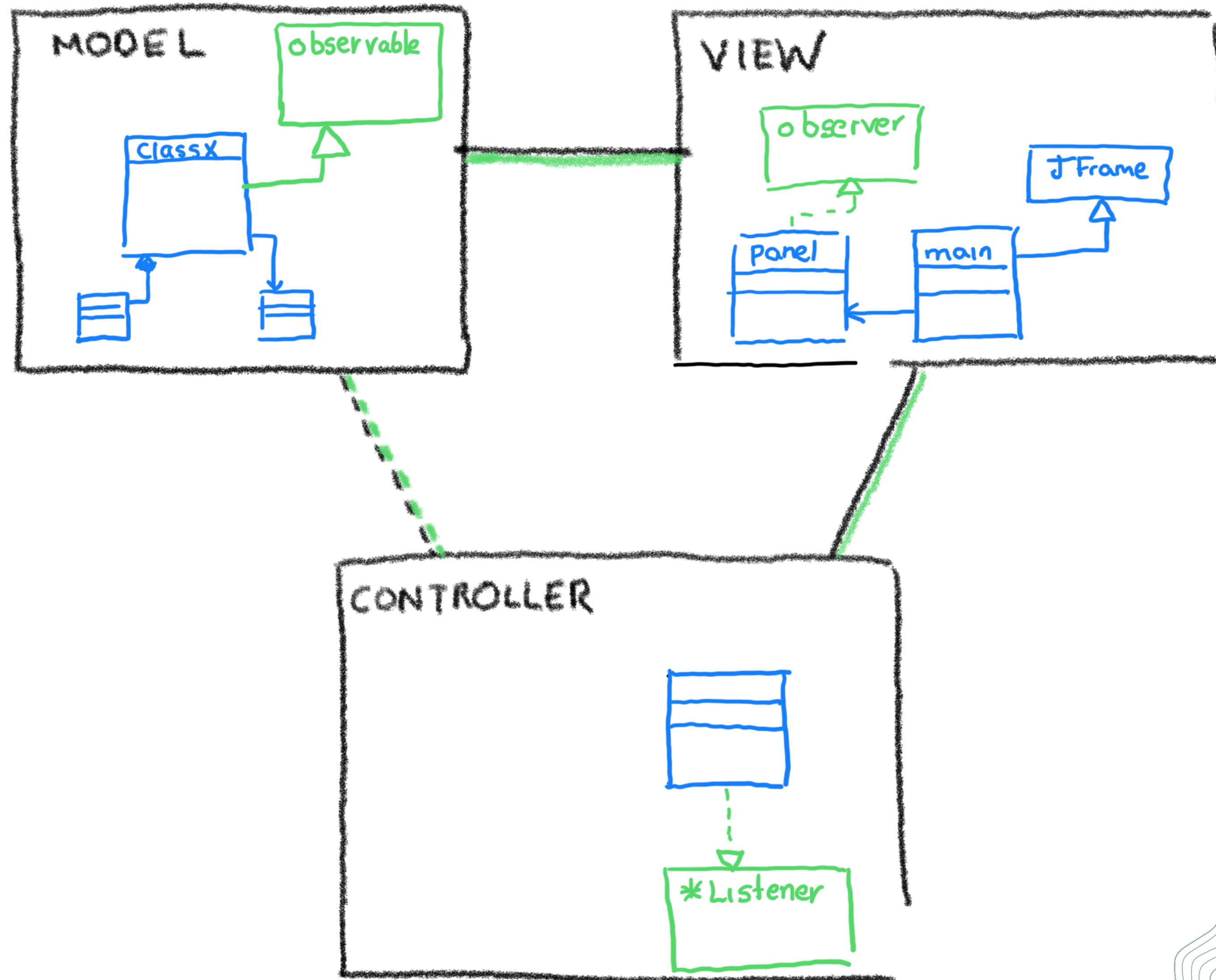
- Model - View - Controller
- Layered
- Blackboard
- **Client-Server**
- **Publisher-Subscriber**

Model-View-Controller

Model-View-Controller



Model-View-Controller



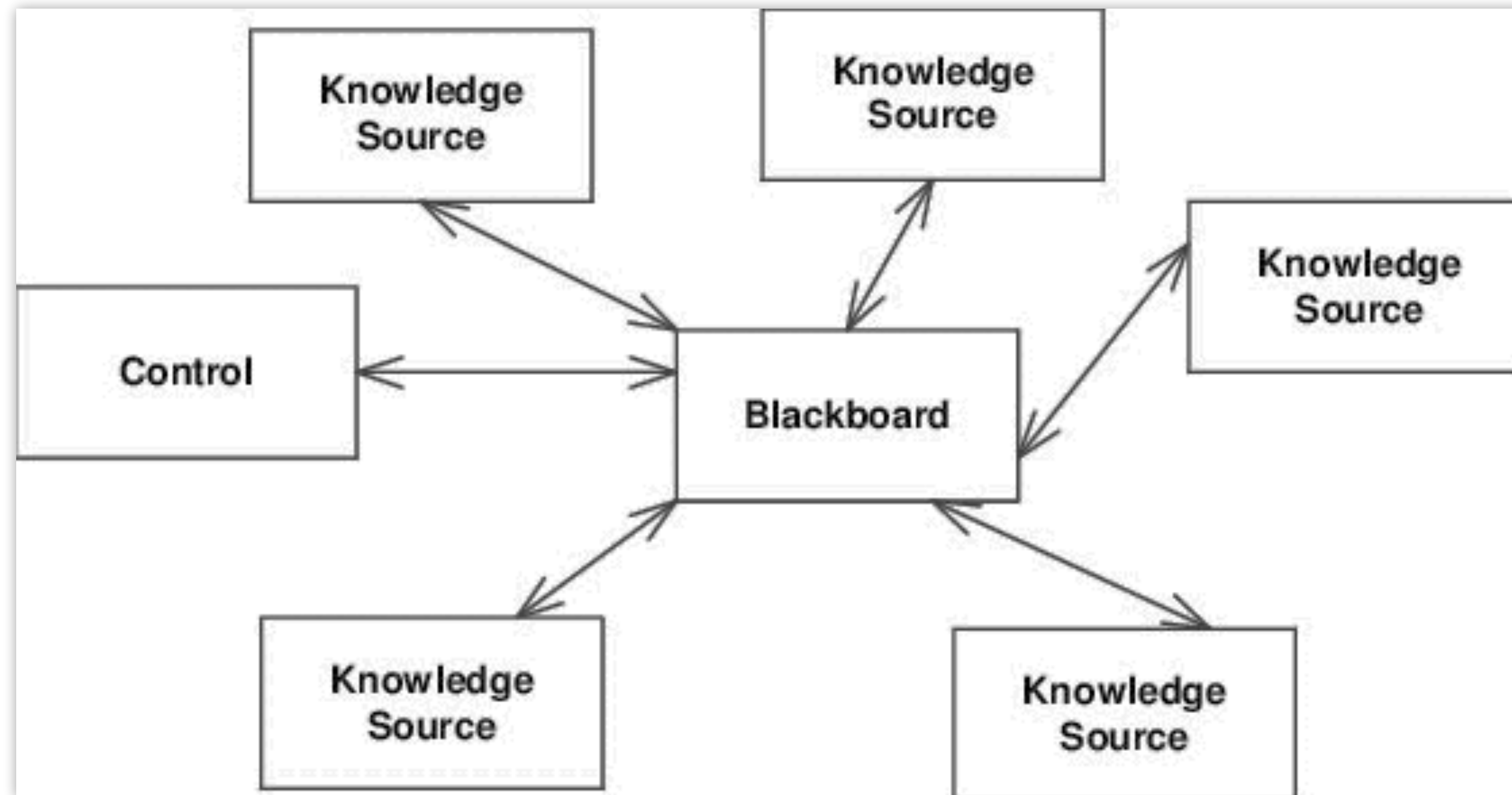

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  ▶ public class Main extends JFrame {
5
6      Main() {
7          Controller controller = new Controller();
8          JButton button1 = new JButton("One");
9          JButton button2 = new JButton("Two");
10         JButton button3 = new JButton("Three");
11         setLayout(new GridLayout(3, 1));
12         add(button1);
13         add(button2);
14         add(button3);
15         button1.addActionListener(controller);
16         button2.addActionListener(controller);
17         button3.addActionListener(controller);
18     }
19
20  ▶ public static void main(String[] args) {
21       Main main = new Main();
22       main.setSize(500, 500);
23       main.setVisible(true);
24   }
25 }
```

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  public class Controller implements ActionListener {
5      @Override
6  @ public void actionPerformed(ActionEvent e) {
7      String action = e.getActionCommand();
8
9      if (action.equals("One")) {
10         System.out.println("Action 1");
11     } else if (action.equals("Two")) {
12         System.out.println("Action 2");
13     } else if (action.equals("Three")) {
14         System.out.println("Action 3");
15     }
16 }
17 }
```

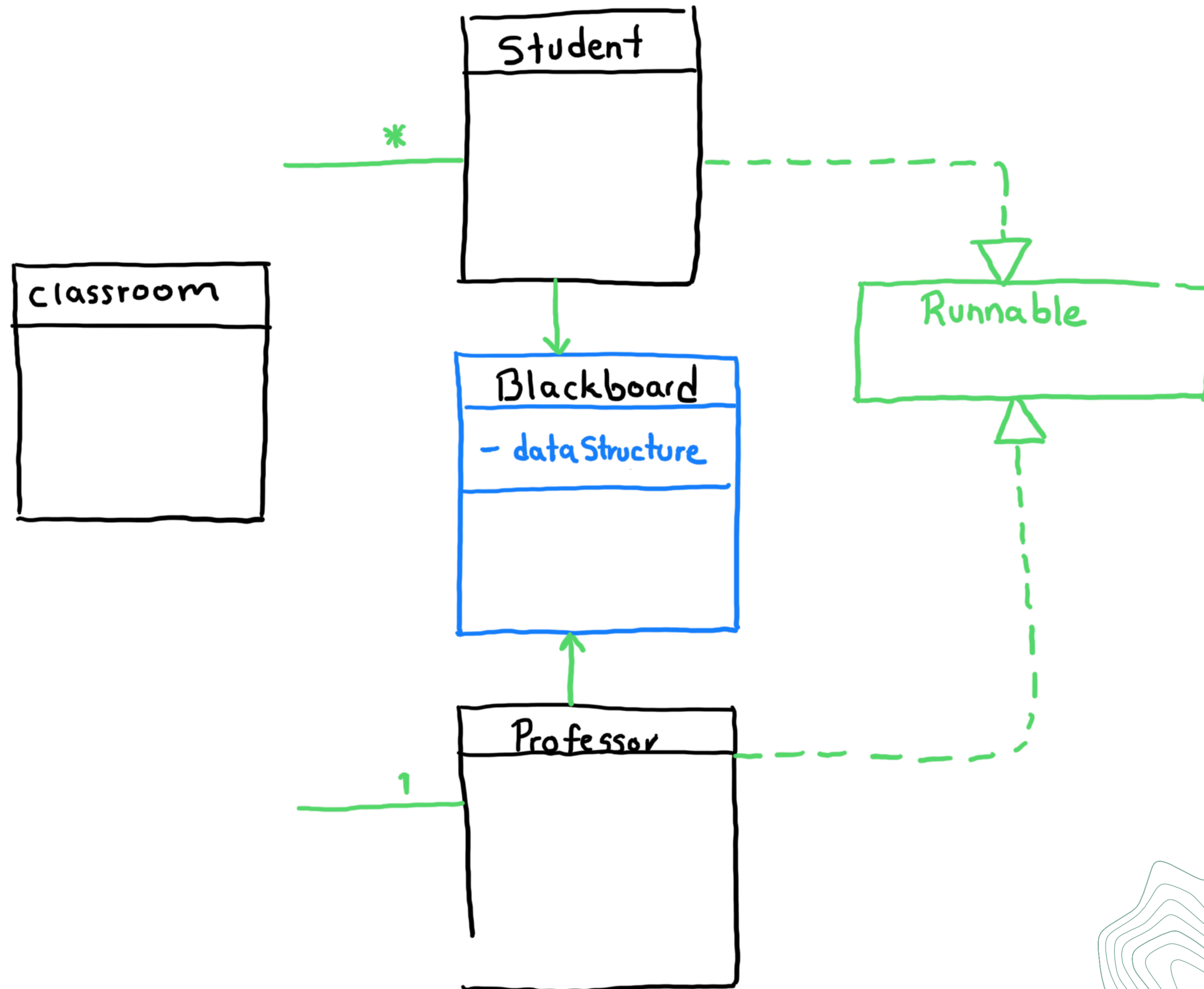

Architecture Patterns

Blackboard

Blackboard



Drafting an Implementation



What about Blackboard

- Singleton (yes or not)?
- Create Blackboard in main method (yes or not)?
- Connection between Blackboard and Professor (association, aggregation, composition, realization, generalization) ?
- Connection between Blackboard and Student (association, aggregation, composition, realization, generalization) ?
- Connection between Blackboard and Classroom (association, aggregation, composition, realization, generalization) ?


```
1 ▶ public class Classroom {  
2  
3 ▶   public static void main(String[] args) {  
4     Thread thread1 = new Thread(new Student(0));  
5  
6     Thread thread2 = new Thread(new Student(1));  
7     Thread thread3 = new Thread(new Professor());  
8     thread1.start();  
9     thread2.start();  
10    thread3.start();  
11  }  
12 }
```

```
1  public class Professor implements Runnable {
2
3      @Override
4  0↑ public void run() {
5      while(true) {
6          System.out.println("The professor Here! " );
7          System.out.println( "-- reading: " + Blackboard.getInstance().read(0) );
8          System.out.println( "-- reading: " + Blackboard.getInstance().read(1) );
9          try {
10             Thread.sleep(1000);
11         } catch (Exception e) {
12
13         }
14     }
15 }
16 }
```



```
1  public class Student implements Runnable {
2      private int id;
3
4      public Student(int id) {
5          this.id = id;
6      }
7
8      @Override
9      public void run() {
10         while(true) {
11             System.out.println("Student Working Here! " + id);
12             Blackboard.getInstance().write(id, ((int)(Math.random()*10)));
13             try {
14                 Thread.sleep(1000);
15             } catch (Exception e) {
16
17             }
18         }
19     }
20 }
```

```
1  import java.util.Vector;
2
3  public class Blackboard {
4
5      private static Blackboard instance;
6      private Vector<Integer> data;
7
8      private Blackboard() {
9          data = new Vector<Integer>(10);
10         data.add(0); data.add(0);
11     }
12
13     public static Blackboard getInstance() {
14         if (instance == null)
15             instance = new Blackboard();
16         return instance;
17     }
18
19     public void write(int index, int item) {
20         data.add(index, item);
21     }
22
23     public int read(int index) {
24         return data.get(index);
25     }
26 }
```


Questions



CSC 509 Software Engineering
Javier Gonzalez-Sanchez, Ph.D.
javiergs@calpoly.edu
Fall 2024

Copyright. These slides can only be used as study material for the class CSC509 at Cal Poly.
They cannot be distributed or used for another purpose.

