# SER516
# Software Agility:
# Project and Process Management

## Lecture 15. Clean Design

**Javier Gonzalez-Sanchez**
**javiergs**@asu.edu
**javiergs**.engineering.asu.edu

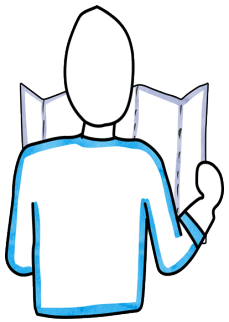
Office Hours: By appointment

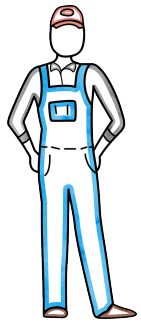
# Building a City

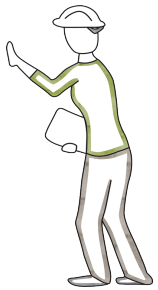
- How to handle all the details?
  Team working

- But some people is responsible for the **big picture,** while others focus on the **details.**

- Architecture





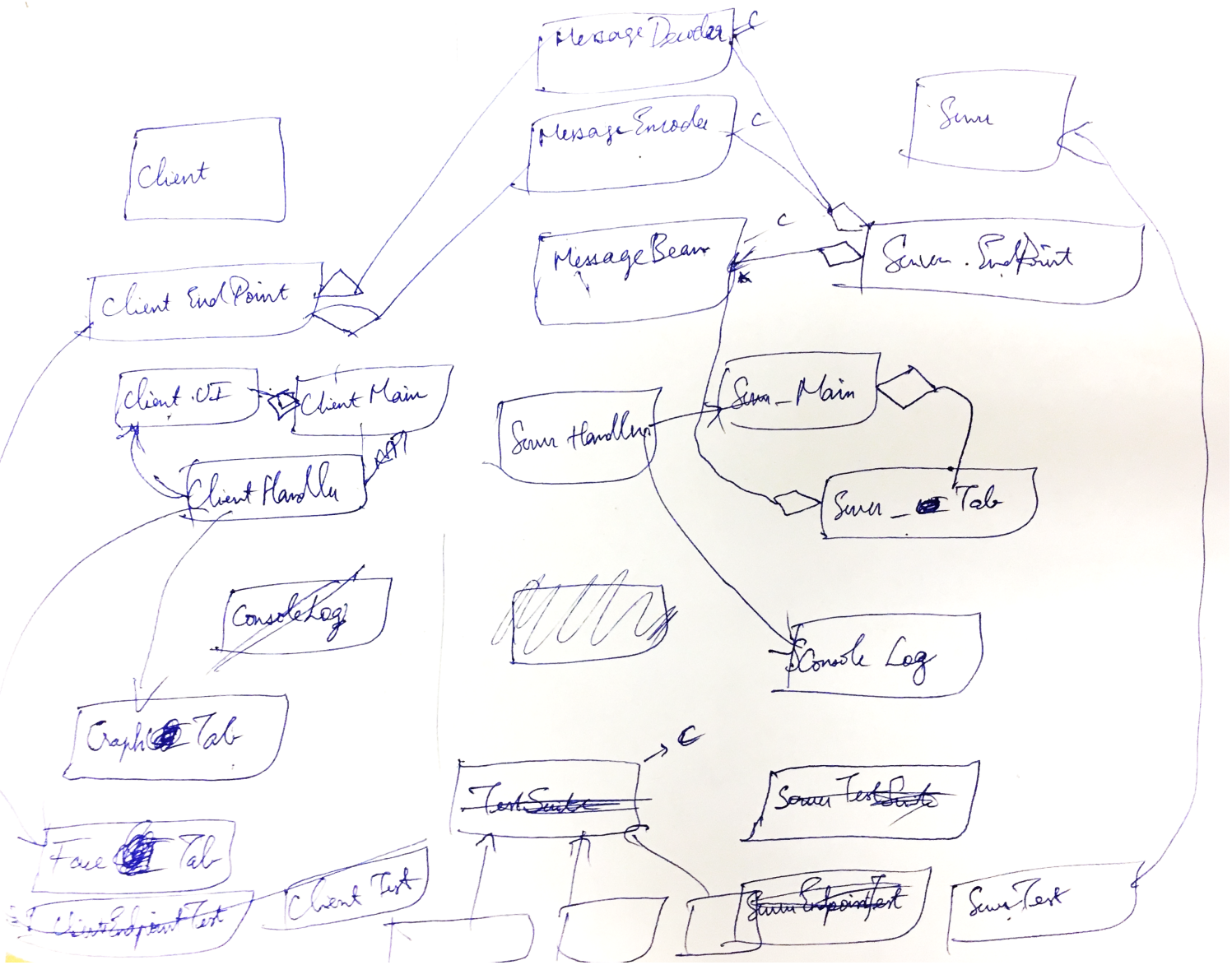


# Goal

- **Modularized** domains of concerns
- **Integrated** with minimal invasive aspects (dependencies)
- It should *feel* like **this is the simplest thing…**

- Be Careful:
- Big Design Up Front (**BDUF**)
- It could Inhibit adapting to change

# Case A


Message Decoder
Message Encoder
Client
Server
MessageBean
Client EndPoint
Server . EndPoint
Client UI
Client Main
Client Handler
Server Handler
Server_Main
Server_Tab
ConsoleLog
Console Log
Graph Tab
Face Tab
Client Test
Server Test

# Design Principles

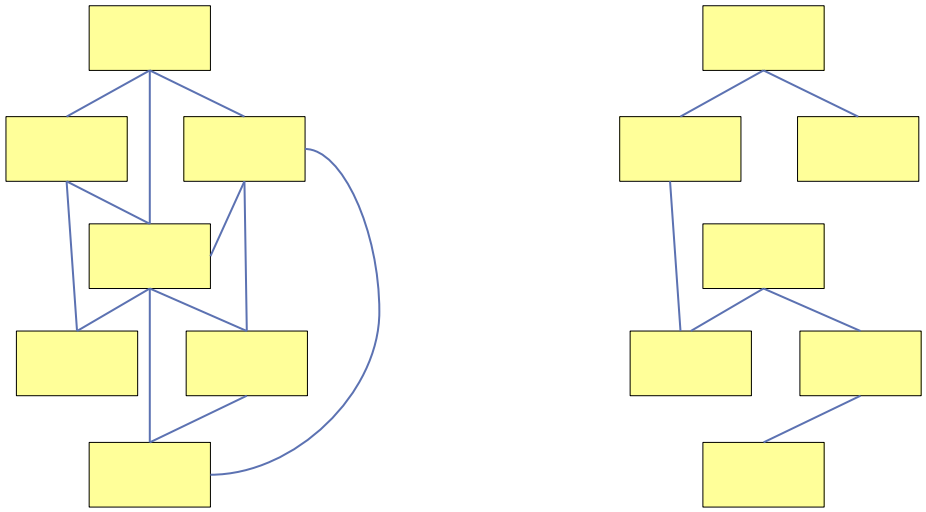
- Concerns and **Separation of concerns**
- Dependencies and **Dependency Injection**
- **Low Coupling**

# Separation of Concerns (SoC)

# Separate concerns

- **Concern**: a matter of interest or importance to someone.

- **Divide** and Conquer: each unit should only talk to its friends; don't talk to strangers



# Patterns

- Observer (behavior)
- Delegate (behavior)
- Factory (creation)

# Observer

```
import java.util.Observable;

public class ObservableValue
    extends Observable {

  private int n = 0;

  public ObservableValue(int n) {
    this.n = n;
  }
  public void setValue(int n) {
    this.n = n;
    setChanged();
    notifyObservers();
  }

  public int getValue() {
    return n;
  }
}
```

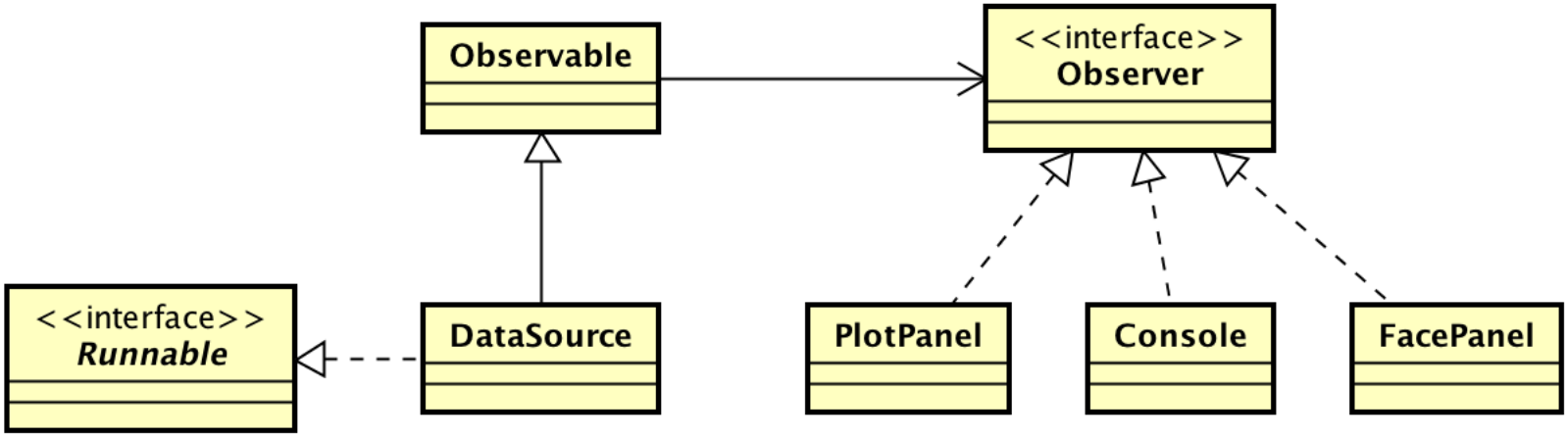
```
import java.util.Observer;
import java.util.Observable;

public class TextObserver
    implements Observer {

  private ObservableValue ov = null;

  public TextObserver(ObservableValue ov) {
    this.ov = ov;
  }

  public void update(Observable obs, Object obj) {
    if (obs == ov) {
      // do something here...
    }
  }
}
```

# Observer

- Maybe, you have a **Console**, a **PlotPanel**, a **FacePanel**, and all of them need **access** to the **data** collected from the server.

- **Queries** vs **Notifications**

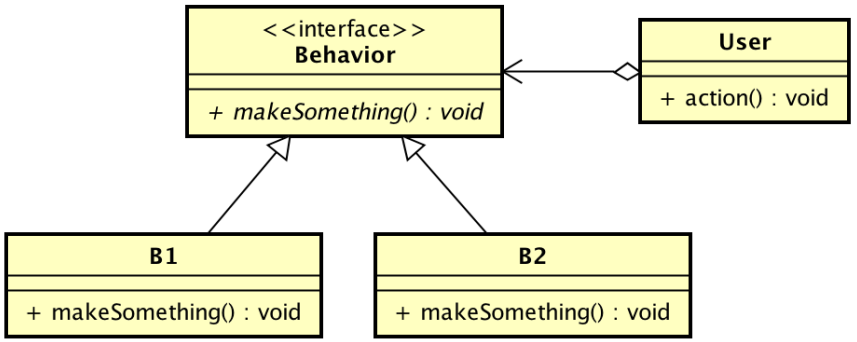
- What about this?



# Delegation

```
public interface Behavior {
  public void makeSomething();
  }
}

public class B1 implements Behavior {
  public void makeSomething() {
    // ...
  }
}

public class B2 implements Behavior {
  public void makeSomething() {
    // ...
  }
}
```

```
public class User {
  private Behavior b;

  public void action() {
    b.makeSomething();
  }

  public User(Behavior b) {
    this.b = b;
  }

}
```

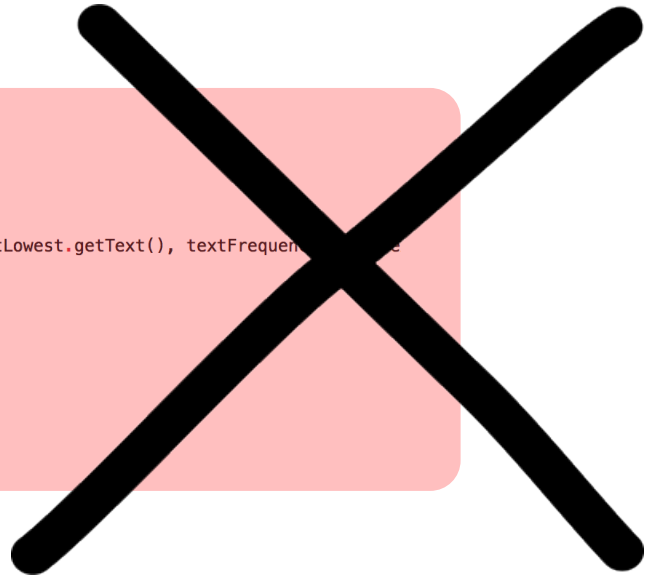


# Do you remember this?

```
106
107         JTextPane textFrequencyHZ = new JTextPane();
108         textFrequencyHZ.setBounds(490, 161, 126, 61);
109         mainPanel.add(textFrequencyHZ);
110         textFrequencyHZ.setBackground(ServerConstants.LIGHTRED);
111         textFrequencyHZ.setFont(ServerConstants.COURIERFONT);
112         textFrequencyHZ.setBorder(BorderFactory.createLineBorder(Color.black));
113
114         JButton btnNewButton = new JButton("Start / Stop");
115         btnNewButton.setBackground(ServerConstants.LIGHTRED);
116         btnNewButton.addActionListener(new ActionListener() {
117                 public void actionPerformed(ActionEvent arg0) {
118                         serverRunning = !serverRunning;
119                         if (serverRunning == true) {
120                                 T.start();
121                                 TextInputs.SetInputs(textHighest.getText(), textLowest.getText(), textFrequencyHZ.getTe
122                                 setServerThread(Server.createServerThread());
123                         } else {
124                                 T.stop();
125                                 getServerThread().stop();
126
127                         }
128                 }
129         });
130         btnNewButton.setFont(ServerConstants.COURIERFONT);
131         btnNewButton.setBounds(448, 13, 192, 34);
132         btnNewButton.setBorder(BorderFactory.createLineBorder(Color.black));
133         serverFrame.getContentPane().add(btnNewButton);
134
135         JLabel lblhighestvalue = new JLabel("<html>Highest<br>value:</html>");
136         lblhighestvalue.setBounds(345, 13, 133, 61);
137         mainPanel.add(lblhighestvalue);
138         lblhighestvalue.setBackground(ServerConstants.LIGHTBLUE);
139         lblhighestvalue.setOpaque(true);
140         lblhighestvalue.setHorizontalAlignment(SwingConstants.CENTER);
141         lblhighestvalue.setVerticalAlignment(SwingConstants.CENTER);
142         lblhighestvalue.setFont(ServerConstants.COURIERFONT);
143         lblhighestvalue.setBorder(BorderFactory.createLineBorder(Color.black));
144
145         JLabel lbllowestvalue = new JLabel("<html>Lowest<br>value:</html>");
146         lbllowestvalue.setBounds(345, 87, 133, 61);
147         mainPanel.add(lbllowestvalue);
```

# Do you remember this?

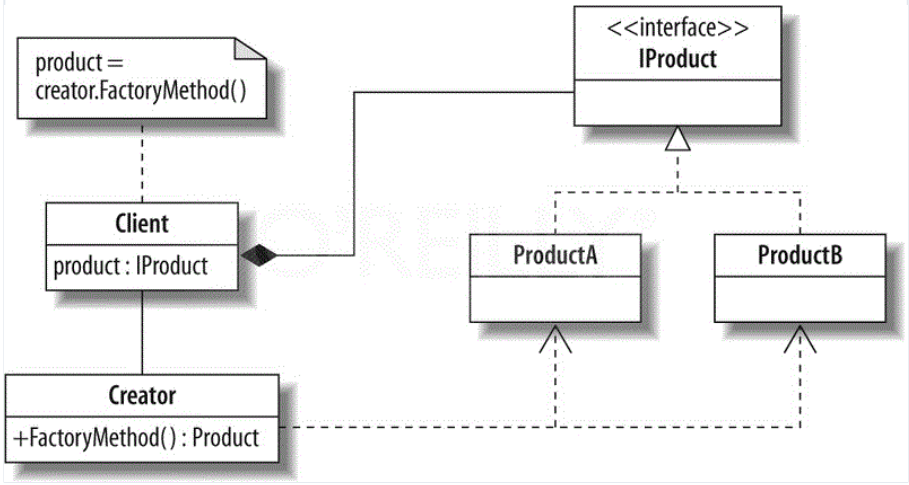
```
106
107         JTextPane textFrequencyHZ = new JTextPane();
108         textFrequencyHZ.setBounds(490, 161, 126, 61);
109         mainPanel.add(textFrequencyHZ);
110         textFrequencyHZ.setBackground(ServerConstants.LIGHTRED);
111         textFrequencyHZ.setFont(ServerConstants.COURIERFONT);
112         textFrequencyHZ.setBorder(BorderFactory.createLineBorder(Color.black));
113
114         JButton btnNewButton = new JButton("Start / Stop");
115         btnNewButton.setBackground(ServerConstants.LIGHTRED);
116         btnNewButton.addActionListener(new ActionListener() {
117                 public void actionPerformed(ActionEvent arg0) {
118                         serverRunning = !serverRunning;
119                         if (serverRunning == true) {
120                                 T.start();
121                                 TextInputs.SetInputs(textHighest.getText(), textLowest.getText(), textFrequen
122                                 setServerThread(Server.createServerThread());
123                         } else {
124                                 T.stop();
125                                 getServerThread().stop();
126
127                         }
128                 }
129         });
130         btnNewButton.setFont(ServerConstants.COURIERFONT);
131         btnNewButton.setBounds(448, 13, 192, 34);
132         btnNewButton.setBorder(BorderFactory.createLineBorder(Color.black));
133         serverFrame.getContentPane().add(btnNewButton);
134
135         JLabel lblhighestvalue = new JLabel("<html>Highest<br>value:</html>");
136         lblhighestvalue.setBounds(345, 13, 133, 61);
137         mainPanel.add(lblhighestvalue);
138         lblhighestvalue.setBackground(ServerConstants.LIGHTBLUE);
139         lblhighestvalue.setOpaque(true);
140         lblhighestvalue.setHorizontalAlignment(SwingConstants.CENTER);
141         lblhighestvalue.setVerticalAlignment(SwingConstants.CENTER);
142         lblhighestvalue.setFont(ServerConstants.COURIERFONT);
143         lblhighestvalue.setBorder(BorderFactory.createLineBorder(Color.black));
144
145         JLabel lbllowestvalue = new JLabel("<html>Lowest<br>value:</html>");
146         lbllowestvalue.setBounds(345, 87, 133, 61);
147         mainPanel.add(lbllowestvalue);
```



# Factory

```
class Creator {
    public Product create() {
      return new Product ();
    }
}

class Client {
  private Iproduct product;

  public Client(Creator factory){
      product= factory.create ();
  }

  public void run() {
     // ...
  }

}
```

```
public static void Main() {

      Creator creator = new Creator();
      Client client = new Client(creator);
      client.run();
     }
```



# Factory

- JSON string to JSON object

- Gson and/or Guava library

```
Gson g = new Gson();
MyClass example = g.fromJson(stringToParse, MyClass.class)
```

- JSON-Simple library

```
JSONParser parser = new JSONParser();
JSONObject json = (JSONObject) parser.parse(stringToParse);
```

- Jackson library

```
MyClass object =
new ObjectMapper().readValue(jsonString, MyClass.class);
```

# Reference

- Clean Code, Chapter 11.

# SER516 – Software Agility

Javier Gonzalez-Sanchez
javiergs@asu.edu

Spring 2018