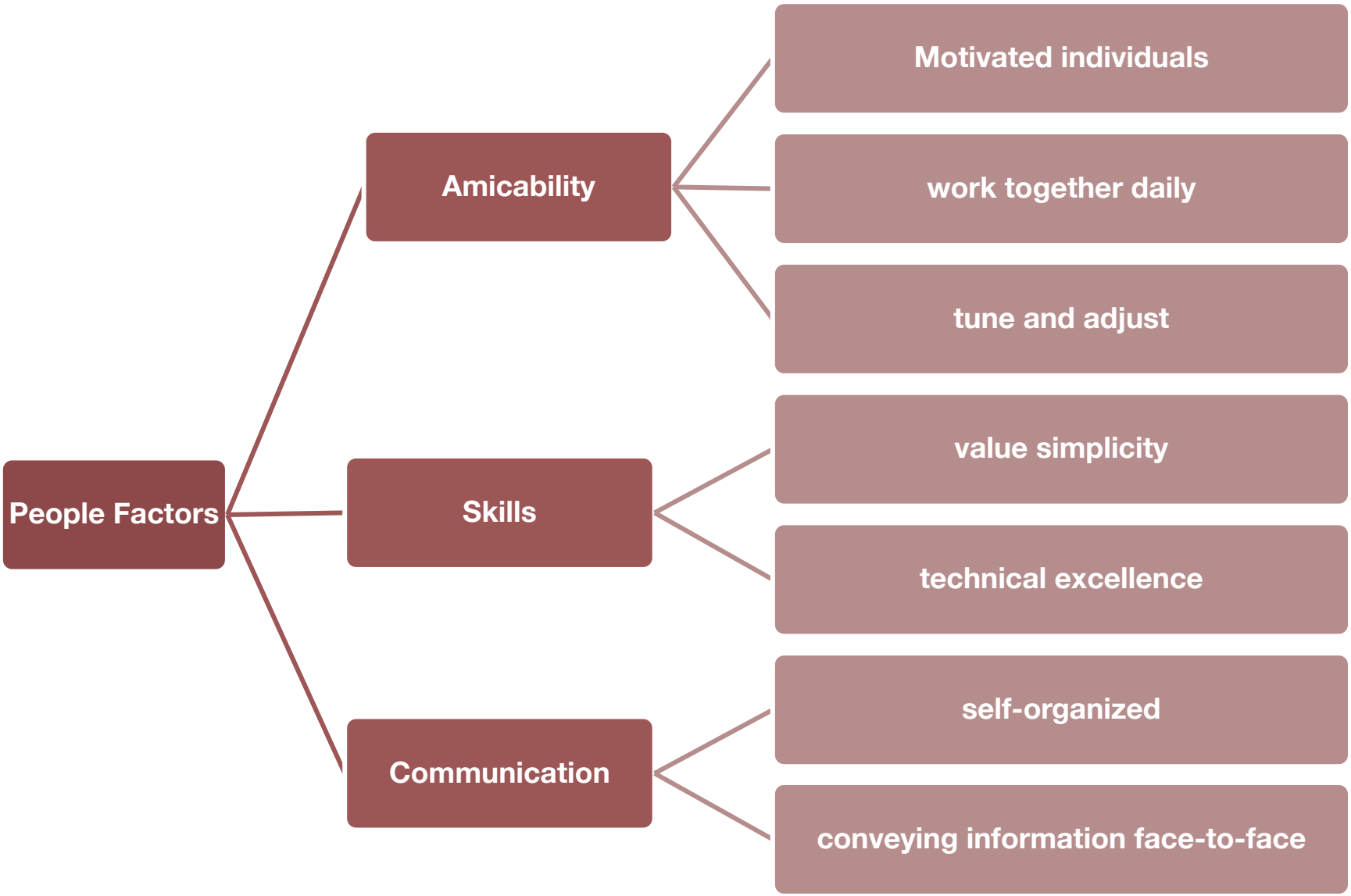
SER516
Software Agility:
Project and Process Management

Lecture 05. Agile is for People

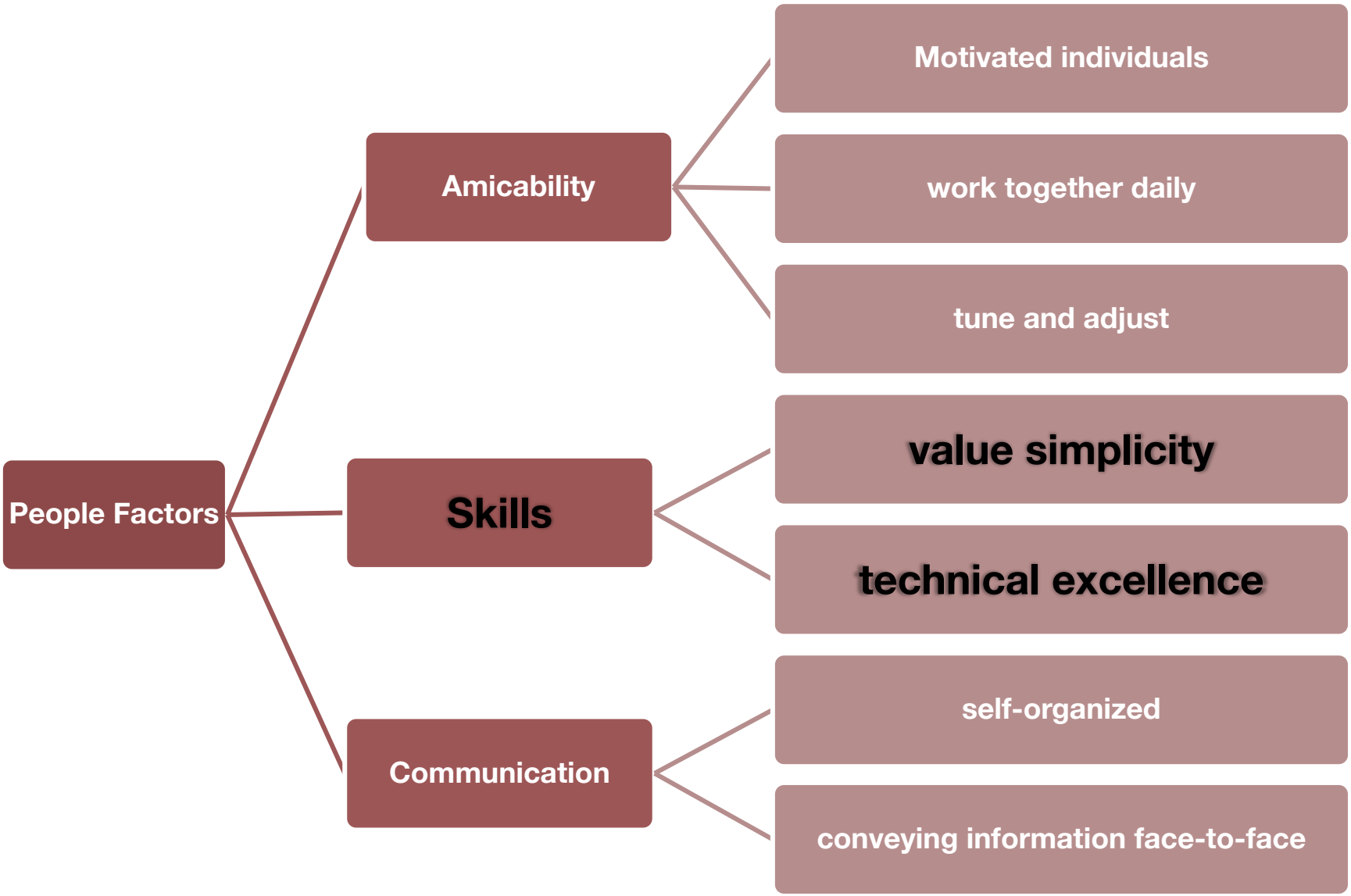
Javier Gonzalez-Sanchez
javiergs@asu.edu
javiergs.engineering.asu.edu

Office Hours: By appointment

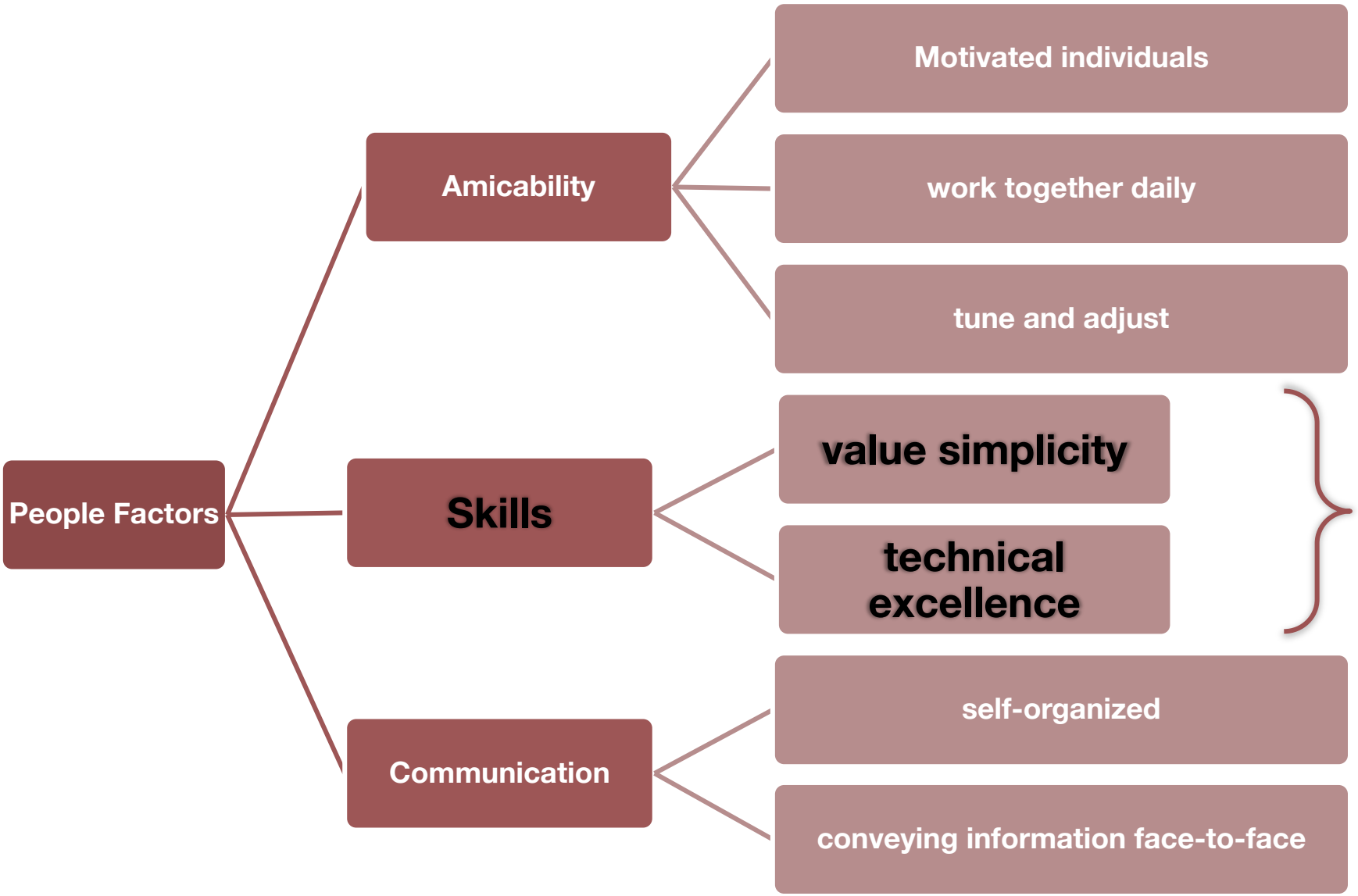
People Factors



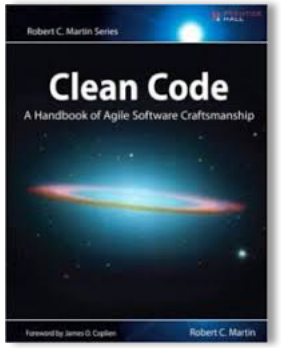
People Factors



People Factors



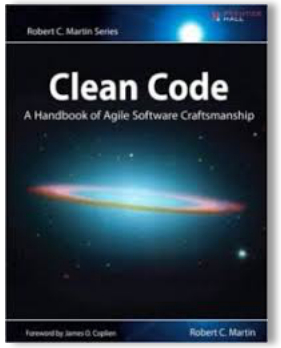
Coding



"Writing **clean code** is what you must do in order to call yourself a **professional**."

–Robert C. Martin

Coding



"Writing **clean code** is what you must do in order to call yourself a **professional**."

-Robert C. Martin

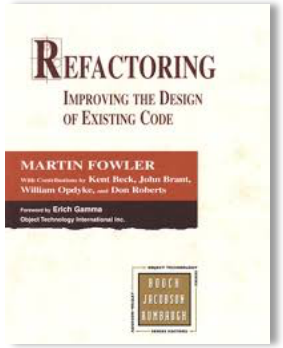
Clean Code:

a) Readable and Well-structured

b) Extensible

c) Testable

Coding



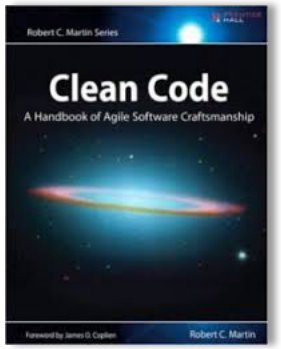
- *"Any fool can write code that a computer can understand. **Good programmers** write **code** that **humans** can **understand**".*

  -Martin Fowler

- *"Even bad code can function".*

  -Robert C. Martin

Coding

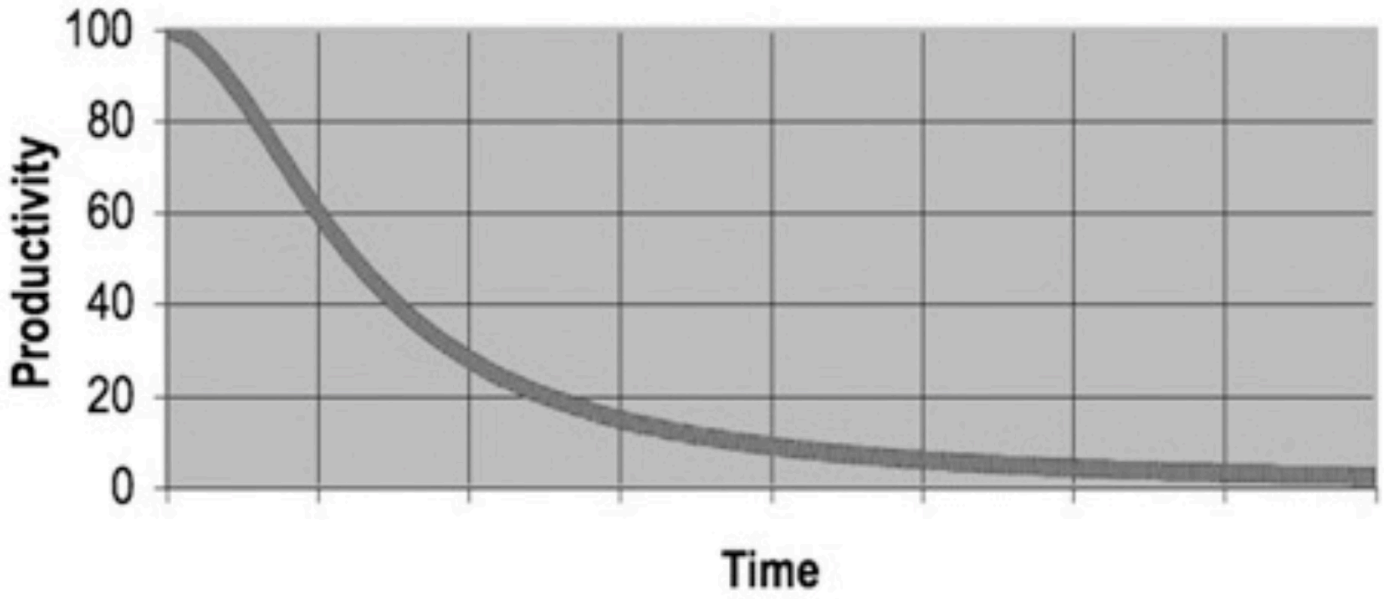


- "The ratio of time spent **reading vs writing** is well over **10:1**"

  –Robert C. Martin

Cost

As a mess build, the productivity of the team decrease.



Eventually the team ask for a **redesign** (i.e., more time)

Clean Code Principles

1. Readability
   e.g. Coding guidelines.

2. Simplification and Specification
   e.g. KISS (Keep It Simple by K. Johnson).

3. Decoupling
   e.g. LOD (Law of Demeter).

4. Avoid Code Bloat
   e.g. DRY (Do not Repeat Yourself).

Do your projects have clean code?

Readability

Readability

```
1  /**
2   *  For SER 516 only.
3   * @author  XXXXXXXXXX
4   * @version 1.0
5   * @since   01-21-2018
6   */
7  import javax.swing.*;
8  import javax.swing.Timer;
9  import java.awt.*;
10 import java.awt.event.ActionListener;
11 import java.awt.event.ActionEvent;
12
13
14 /**
15  *          It Adds two labels in the
16  *          JPanel with Background Color - White if the flag received through
17  *          constructor as a parameter is even and light blue if its odd.
18  *          The timer Increments or decrements every second.
19  *          It is incremented when the flag received from the constructor is even, goes from 0 to 9
20  *          and repeats,
21  *          It is decremented when the flag received from the constructor is odd, goes fro 9 to 0
22  *          and repeats.
23  */
24 public class P001_XXXXXXXXXX_Panel extends JPanel
25 {
26     private static int increment= -1;
27     private static int decrement=10;
28     private JLabel label;
29     private JLabel label1;
30
31     public P001_XXXXXXXXXX_Panel(int flag)//Constructor where flag decides the timer execution
32     {
33         /**
34          * Panel and label properties: Font, Color, Text, Layout,
35          * size and calls function made for the timer execution
36          */
37         setSize(100,100);
38         setLayout(new GridLayout(2, 1));
39         label = new JLabel("<html>XXXXXXXXXX<br>XXXXXXXXXX</html>", JLabel.CENTER);
40         label1 = new JLabel("", JLabel.CENTER);
41         label.setFont(new Font("Papyrus", Font.PLAIN, 15));
42         label1.setFont(new Font("Papyrus", Font.PLAIN, 15));
43         add(label);
44         add(label1);
45         start_timer(flag);
46     }
```

Readability

```
1  /**
2   *  For SER 516 only.
3   * @author  XXXXXXXXXX
4   * @version 1.0
5   * @since   01-21-2018
6   */
7  import javax.swing.*;
8  import javax.swing.Timer;
9  import java.awt.*;
10 import java.awt.event.ActionListener;
11 import java.awt.event.ActionEvent;
12
13
14 /**
15  *          It Adds two labels in the
16  *          JPanel with Background Color - White if the flag received through
17  *          constructor as a parameter is even and light blue if its odd.
18  *          The timer Increments or decrements every second.
19  *          It is incremented when the flag received from the constructor is even, goes from 0 to 9
20  *          and repeats,
21  *          It is decremented when the flag received from the constructor is odd, goes fro 9 to 0
22  *          and repeats.
23  */
24 public class P001_XXXXXXXXXX_Panel extends JPanel
25 {
26     private static int increment= -1;
27     private static int decrement=10;
28     private JLabel label;
29     private JLabel label1;
30
31     public P001_XXXXXXXXXX_Panel(int flag)//Constructor where flag decides the timer execution
32     {
33         /**
34          * Panel and label properties: Font, Color, Text, Layout,
35          * size and calls function made for the timer execution
36          */
37         setSize(100,100);
38         setLayout(new GridLayout(2, 1));
39         label = new JLabel("<html>XXXXXXXXXX<br>XXXXXXXXXX</html>", JLabel.CENTER);
40         label1 = new JLabel("", JLabel.CENTER);
41         label.setFont(new Font("Papyrus", Font.PLAIN, 15));
42         label1.setFont(new Font("Papyrus", Font.PLAIN, 15));
43         add(label);
44         add(label1);
45         start_timer(flag);
46     }
```

Readability

```
1  /**
2   *  For SER 516 only.
3   * @author  XXXXXXXXXX
4   * @version 1.0
5   * @since   01-21-2018
6   */
7  import javax.swing.*;
8  import javax.swing.Timer;
9  import java.awt.*;
10 import java.awt.event.ActionListener;
11 import java.awt.event.ActionEvent;
12
13
14 /**
15  *         It Adds two labels in the
16  *         JPanel with Background Color - White if the flag received through
17  *         constructor as a parameter is even and light blue if its odd.
18  *         The timer Increments or decrements every second.
19  *         It is incremented when the flag received from the constructor is even, goes from 0 to 9
20  *         and repeats,
21  *         It is decremented when the flag received from the constructor is odd, goes fro 9 to 0
22  *         and repeats.
23  */
24 public class P001_XXXXXXXXXX_Panel extends JPanel
25 {
26     private static int increment= -1;
27     private static int decrement=10;
28     private JLabel label;
29     private JLabel label1;
30
31     public P001_XXXXXXXXXX_Panel(int flag)//Constructor where flag decides the timer execution
32     {
33         /**
34          * Panel and label properties: Font, Color, Text, Layout,
35          * size and calls function made for the timer execution
36          */
39         setSize(100,100);
40         setLayout(new GridLayout(2, 1));
41         label = new JLabel("<html>XXXXXXXXXX<br>XXXXXXXXXX</html>", JLabel.CENTER);
42         label1 = new JLabel("", JLabel.CENTER);
43         label.setFont(new Font("Papyrus", Font.PLAIN, 15));
44         label1.setFont(new Font("Papyrus", Font.PLAIN, 15));
45         add(label);
46         add(label1);
47         start_timer(flag);
48     }
```

Readability

```
1  /**
2   *  For SER 516 only.
3   * @author  XXXXXXXXXX
4   * @version 1.0
5   * @since   01-21-2018
6   */
7  import javax.swing.*;
8  import javax.swing.Timer;
9  import java.awt.*;
10 import java.awt.event.ActionListener;
11 import java.awt.event.ActionEvent;
12
13
14 /**
15  *         It Adds two labels in the
16  *         JPanel with Background Color - White if the flag received through
17  *         constructor as a parameter is even and light blue if its odd.
18  *         The timer Increments or decrements every second.
19  *         It is incremented when the flag received from the constructor is even, goes from 0 to 9
20  *         and repeats,
21  *         It is decremented when the flag received from the constructor is odd, goes fro 9 to 0
22  *         and repeats.
23  */
24 public class P001_XXXXXXXXXX_Panel extends JPanel
25 {
26     private static int increment= -1;
27     private static int decrement=10;
        private JLabel label;
        private JLabel label1;
30
31      public P001_XXXXXXXXXX_Panel(int flag)//Constructor where flag decides the timer execution
32      {
33          /**
34           * Panel and label properties: Font, Color, Text, Layout,
35           * size and calls function made for the timer execution
36           */
            setSize(100,100);
            setLayout(new GridLayout(2, 1));
39          label = new JLabel("<html>XXXXXXXXXX<br>XXXXXXXXXX</html>", JLabel.CENTER);
40          label1 = new JLabel("", JLabel.CENTER);
41          label.setFont(new Font("Papyrus", Font.PLAIN, 15));
42          label1.setFont(new Font("Papyrus", Font.PLAIN, 15));
            add(label);
            add(label1);
            start_timer(flag);
46      }
```

Readability

```
1  /**
2   *  For SER 516 only.
3   * @author  XXXXXXXXXX
4   * @version 1.0
5   * @since   01-21-2018
6   */
7  import javax.swing.*;
8  import javax.swing.Timer;
9  import java.awt.*;
10 import java.awt.event.ActionListener;
11 import java.awt.event.ActionEvent;
12
13
14 /**
15  *         It Adds two labels in the
16  *         JPanel with Background Color - White if the flag received through
17  *         constructor as a parameter is even and light blue if its odd.
18  *         The timer Increments or decrements every second.
19  *         It is incremented when the flag received from the constructor is even, goes from 0 to 9
20  *         and repeats,
21  *         It is decremented when the flag received from the constructor is odd, goes fro 9 to 0
22  *         and repeats.
23  */
24 public class P001_XXXXXXXXXX_Panel extends JPanel
25 {
26     private static int increment= -1;
27     private static int decrement=10;
28     private JLabel label;
29     private JLabel label1;
30
31     public P001_XXXXXXXXXX_Panel(int flag)//Constructor where flag decides the timer execution
32     {
33         /**
34          * Panel and label properties: Font, Color, Text, Layout,
35          * size and calls function made for the timer execution
36          */
37         setSize(100,100);
38         setLayout(new GridLayout(2, 1));
39         label = new JLabel("<html>XXXXXXXXXX<br>XXXXXXXXXX</html>", JLabel.CENTER);
40         label1 = new JLabel("", JLabel.CENTER);
41         label.setFont(new Font("Papyrus", Font.PLAIN, 15));
42         label1.setFont(new Font("Papyrus", Font.PLAIN, 15));
43         add(label);
44         add(label1);
45         start_timer(flag);
46     }
```

Readability

```
47
48      /**
49       * Timer execution : From 0 to 9 when flag is even
50       * And from 9 to 0 when flag is odd
51       * @param flag
52       */
53      public void start_timer(int flag)
54      {
55          try {
56              Timer timer1 = new Timer(1000, new ActionListener() {
57                  public void actionPerformed(ActionEvent e) {
58                      if (flag % 2 == 0) {
59
60                          if (increment == 9) {
61                              increment = -1;
62                          }
63                          increment = increment + 1;
64                          label1.setText("" + increment);
65
66                      }
67
68                      else {
69
70                          setBackground(new Color(171, 216, 230));
71
72
73                          if (decrement == 0)
74                          {
75                              decrement = 10;
76                          }
77                          decrement = decrement - 1;
78                          label1.setText("" + decrement);
79                      }
80
81                  }
82              });
83
84              timer1.start();
85          }
86          catch (Exception e)//Exception thrown
87          {
88              System.out.print("Exception Thrown in Timer");
89          }
90      }
```

Flag?

Readability

```
47
48      /**
49       * Timer execution : From 0 to 9 when flag is even
50       * And from 9 to 0 when flag is odd
51       * @param flag
52       */
53      public void start_timer(int flag)
54      {
55          try {
56              Timer timer1 = new Timer(1000, new ActionListener() {
57                  public void actionPerformed(ActionEvent e) {
58                      if (flag % 2 == 0) {
59
60                          if (increment == 9) {
61                              increment = -1;
62                          }
63                          increment = increment + 1;
64                          label1.setText("" + increment);
65
66                      }
67
68                      else {
69
70                          setBackground(new Color(171, 216, 230));
71
72
73                          if (decrement == 0)
74                          {
75                              decrement = 10;
76                          }
77                          decrement = decrement - 1;
78                          label1.setText("" + decrement);
79                      }
80
81                  }
82              });
83
84              timer1.start();
85          }
86          catch (Exception e)//Exception thrown
87          {
88              System.out.print("Exception Thrown in Timer");
89          }
90      }
```

Readability

```
47
48      /**
49       * Timer execution : From 0 to 9 when flag is even
50       * And from 9 to 0 when flag is odd
51       * @param flag
52       */
53      public void start_timer(int flag)
54      {
55          try {
              Timer timer1 = new Timer(1000, new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
58                    if (flag % 2 == 0) {
59
60                        if (increment == 9) {
61                            increment = -1;
62                        }
                          increment = increment + 1;
64                        label1.setText("" + increment);
65
66                    }
67
68                    else {
69
70                        setBackground(new Color(171, 216, 230));
71
72
73                        if (decrement == 0)
74                        {
75                            decrement = 10;
76                        }
77                        decrement = decrement - 1;
78                        label1.setText("" + decrement);
79                    }
80
81                }
82            });
83
84            timer1.start();
85        }
86        catch (Exception e)//Exception thrown
87        {
88            System.out.print("Exception Thrown in Timer");
89        }
90    }
```

Readability

```
47
48      /**
49       * Timer execution : From 0 to 9 when flag is even
50       * And from 9 to 0 when flag is odd
51       * @param flag
52       */
53      public void start_timer(int flag)
54      {
55          try {
              Timer timer1 = new Timer(1000, new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
58                    if (flag % 2 == 0) {
59
60                        if (increment == 9) {
61                            increment = -1;
62                        }
                          increment = increment + 1;
64                        label1.setText("" + increment);
65
66                    }
67
68                    else {
69
70                        setBackground(new Color(171, 216, 230));
71
72
73                        if (decrement == 0)
74                        {
75                            decrement = 10;
76                        }
77                        decrement = decrement - 1;
78                        label1.setText("" + decrement);
79                    }
80
81                }
82            });
83
84            timer1.start();
85        }
86        catch (Exception e)//Exception thrown
87        {
88            System.out.print("Exception Thrown in Timer");
89        }
90    }
```

Readability

```
91
92      /*public static void main(String args[])
93      {
94          P001_XXXXXXXXXX_Panel p= new P001_XXXXXXXXXX_Panel(1);
95          JFrame frame = new JFrame();
96          frame.setSize(500, 500);
97          frame.add(p);
98          frame.setVisible(true);
99
100     }*/
101 }
102
```

Readability

```
74                  public Counter(JLabel lbl, int c) {
75                      this.lblCounter = lbl;
76                      this.counter = c;
77                      this.flag = c;
78                  }
79
80                  @Override
                    public void run() {
82                      try {
83                          while (true) {
84                              try {
85                                  lblCounter.setText(this.counter + "");
86                                  if (flag == 0) {
87                                      if (++this.counter > 9)
88                                          this.counter = 0;
89                                  } else {
90                                      if (--this.counter < 0)
91                                          this.counter = 9;
92                                  }
                                    Thread.sleep(1000);
94                              } catch (InterruptedException e) {
                                    e.printStackTrace();
96                              }
97                          }
98
99                      } catch (Exception e) {
                            e.printStackTrace();
101                     }
102                 }
103             }
104     }
```

Do your projects have clean code?

Simplification

Simplification

```
public class P_XXXXX_Tab extends JPanel {

    public P_XXXXX_Tab() {
        this.setLayout(new GridLayout(5, 5));
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel().preparePanelUI(2));
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel().getJPanelInstance());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel().preparePanelUI(new Color(173, 216, 230)));
        this.add(new P_XXXXX_Panel());
        this.add(P_XXXXX_Panel.getInstance());
        this.add(new P_XXXXX_Panel().P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
        this.add(new P_XXXXX_Panel());
    }
}
```

Simplification

```
11  public P_XXXXX_Main() {
12
13      try {
14
15          /* Tabbed Pane allows the creation of tabs in a window and each
16          *  tab can be linked with a JPanel.
17          * */
18          JTabbedPane myTabPane = new JTabbedPane();
19
20          /* Creating JPanel Objects to Link to the tabs.
21          Variables Named after the last names of the Tab Managers.
22          * */
            JPanel tab1 = new P1_XXXXX_Tab();
            JPanel tab2 = new P2_XXXXX_Tab();
            JPanel tab3 = new P3_XXXXX_Tab();
            P4_XXXXX_Tab pretab4 = new P4_XXXXX_Tab();
27          JPanel tab4 = pretab4.createTab();
            JPanel tab5 = new P5_Bahl_Tab();
29
30          /* Creating tabs from Panels and assigning other attributes
31          * like border and Titles to each tab.
32          * */
33          myTabPane.addTab("XXXXXnameXXXXX", tab1);
34          myTabPane.addTab("XXXXXnameXXXXX", tab2);
35          myTabPane.addTab("XXXXXnameXXXXX", tab3);
36          myTabPane.addTab("XXXXXnameXXXXX", tab4);
37          myTabPane.addTab("XXXXXnameXXXXX", tab5);
38
39          myTabPane.setBorder(new EmptyBorder(10, 10, 10, 10));
40
            setLayout(new BorderLayout());
            add(myTabPane, BorderLayout.CENTER);
```

Simplification

```
11  public P_XXXXX_Main() {
12
13      try {
14
15          /* Tabbed Pane allows the creation of tabs in a window and each
16          *  tab can be linked with a JPanel.
17          * */
18          JTabbedPane myTabPane = new JTabbedPane();
19
20          /* Creating JPanel Objects to Link to the tabs.
21          Variables Named after the last names of the Tab Managers.
22          * */
            JPanel tab1 = new P1_XXXXX_Tab();
            JPanel tab2 = new P2_XXXXX_Tab();
            JPanel tab3 = new P3_XXXXX_Tab();
            P4_XXXXX_Tab pretab4 = new P4_XXXXX_Tab();
27          JPanel tab4 = pretab4.createTab();
            JPanel tab5 = new P5_Bahl_Tab();
29
30          /* Creating tabs from Panels and assigning other attributes
31          * like border and Titles to each tab.
32          * */
33          myTabPane.addTab("XXXXXnameXXXXX", tab1);
34          myTabPane.addTab("XXXXXnameXXXXX", tab2);
35          myTabPane.addTab("XXXXXnameXXXXX", tab3);
36          myTabPane.addTab("XXXXXnameXXXXX", tab4);
37          myTabPane.addTab("XXXXXnameXXXXX", tab5);
38
39          myTabPane.setBorder(new EmptyBorder(10, 10, 10, 10));
40
            setLayout(new BorderLayout());
            add(myTabPane, BorderLayout.CENTER);
```

Where should the name be?

Do your projects have clean code?

Avoid Code Bloat

Bloat

```
74          public Counter(JLabel lbl, int c) {
75                  this.lblCounter = lbl;
76                  this.counter = c;
77                  this.flag = c;
78          }
79
80          @Override
            public void run() {
82              try {
83                  while (true) {
84                      try {
85                          lblCounter.setText(this.counter + "");
86                          if (flag == 0) {
87                              if (++this.counter > 9)
88                                  this.counter = 0;
89                          } else {
90                              if (--this.counter < 0)
91                                  this.counter = 9;
92                          }
                            Thread.sleep(1000);
94                      } catch (InterruptedException e) {
                            e.printStackTrace();
96                      }
97                  }
98
99              } catch (Exception e) {
                        e.printStackTrace();
101             }
102         }
103     }
104 }
```

Bloat

```
49      public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
51          @Override
            public void run() {
53              /* Main Method that creates a JFrame object and adds the JTabbedPane Object. */
54              JFrame frame = new JFrame();
55              frame.add(new P_XXXXX_Main());
56              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
57              frame.setLocationByPlatform(true);
58              frame.setSize(800, 800);
59              frame.setTitle("XXXXXnameXXXXX");
60              frame.setVisible(true);
61          }
62      });
63  }
64  }
```

Conclusion

Minimum Guidelines

- Follow standard conventions
  https://google.github.io/styleguide/javaguide.html

- Be **consistent**. If you do something a certain way, do all similar things in the same way.

- Use explanatory variables.

- Don't write methods which works correctly depending on something else in the same class.

- Keep lines short.

Minimum Guidelines

- Always try to explain yourself **in code, i.e.,** avoid comments if possible.
- Don't be redundant.
- Don't add obvious noise.
- Don't use closing brace comments.
- **Don't comment out code. Just remove.**
- Use comments as (1)explanation of intent, (2) clarification of code, (3) warning of consequences.

Test Yourselves

a) *"The ratio of time spent reading vs writing is well over 10:1"*

b) Will you be able to read your classmate code?

c) Will your classmate be able to read your code?

SER516 – Software Agility

Javier Gonzalez-Sanchez

javiergs@asu.edu

Spring 2020