Applet Review

What do these lines of code do?

1. showStatus(“Hi”);
2. resize(230, 340);
3. setBackground(Color.pink);

Why isn’t there anything in front of setBackground?

Lines of code related to a button.

```
1.  JButton enter = new JButton ("Enter");

2.  enter.setBackground (Color.black);
3.  enter.setForeground (Color.white);
4.  enter.setPreferredSize (new Dimension (128, 128));
5.  enter.setFont (new Font ("Arial", Font.BOLD, 40));

6.  enter.addActionListener (this);
7.  enter.setActionCommand ("enter");
8.  add (enter);

9.  enter.setIcon(createImageIcon("name.jpg"));
10. enter.setText(“hi”);
```

Declaration

- Making a new variable. It sets aside RAM to hold your information.
- Done by writing: int a;
- Or with widgets as: JLabel b;

Construction

- Prepares RAM for a complex variable like widgets
- A widget must be constructed before it can be used.
- It’s keyword is new

Constructors

1. new JLabel
2. new JButton
3. new JTextField

Mutation

- Changes the widget's RAM
- Makes the widget look different
- It's keyword is set
- Some examples: setBackground, setForeground, setIcon, setFont, setPreferredSize, setActionCommand

Mutators

1. setIcon
2. setText
3. setActionCommand
4. setEnabled

5. setForeground
6. setBackground
7. setFont
8. setPreferredSize

Accessor

- A method that allows the programmer to access the widget and find out what is in it.
- It is only used in ActionPerformed. In init we know the values because we just set them. In AP, they may have changed.
- It’s keyword is “get”
- Examples: getText() from a JTextField, e.getActionCommand()

Accessors

1. getActionCommand
2. getText

RGB Colours – Red, Green, Blue

Red – 255, 0, 0
Green – 0, 255, 0
Blue – 0, 0, 255

Black – 0, 0, 0
White – 255, 255, 255

Red + Green = Yellow

Red + Blue = Magenta

Blue + Green = Cyan

Places in the Code

Libraries

- Start with import

Global

- Declarations
- int total = 0;
- JLabel result;

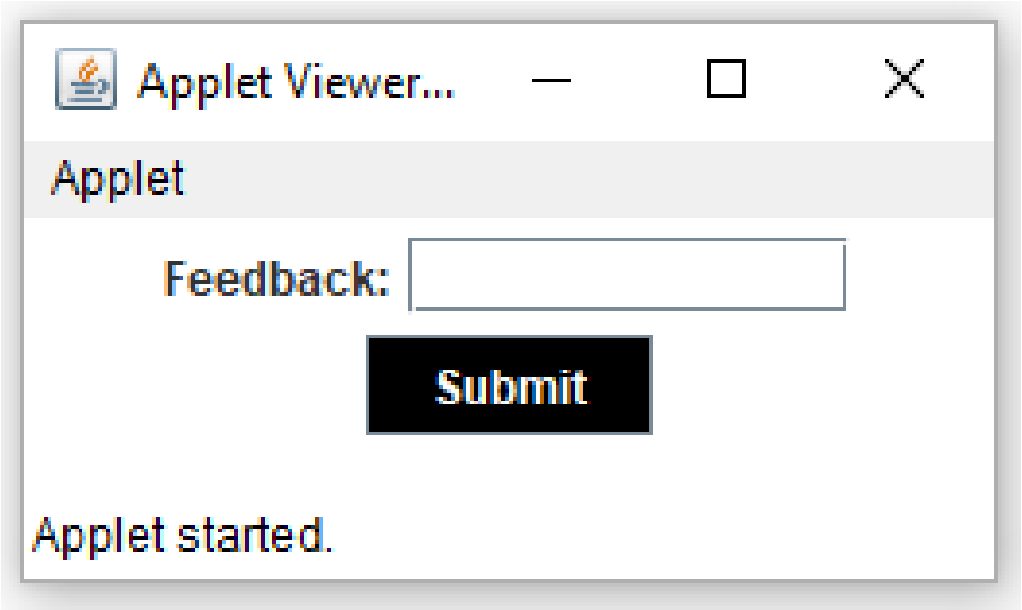
Init

- Construction (new)
- Add
- setActionListener
- addActionCommand

AP

- Gets
- setText
- setIcon

How many widgets on the screen?



How many widgets on the screen?

Fill in the blanks to make this button.


```
JButton end = new JButton("_____________");
_____.setBackground(Color.red);
_____.setForeground(Color._____);
add(_____);
```

The solution.


```
JButton end = new JButton("Game Over!");
end.setBackground(Color.red);
end.setForeground(Color.blue);
add(end);
```

Fill in the blanks to make this textfield.

```
JTextField name = new _____(5);
_____(_____);
```

The solution.

```
JTextField name = new JTextField(5);
add(name);
```

Fill in the blanks to make this label.

Welcome!

```
_____ _____ = _____ JLabel("Welcome!");
title.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));
add(_____);
```

The solution.


```
JLabel title = new JLabel("Welcome!");
title.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));
add(title);
```

What stage of the widget lifecycle is highlighted?


```
JLabel title = new JLabel("Welcome!");
title.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));
add(title);
```

What stage of the widget lifecycle is highlighted?


```
JLabel title = new JLabel("Welcome!");
title.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));
add(title);
```

What stage of the widget lifecycle is highlighted?


```
JLabel title = new JLabel("Welcome!");
title.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));
add(title);
```

What stage of the widget lifecycle is highlighted?


```
JLabel title = new JLabel("Welcome!");
title.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));
add(title);
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class Example1 extends Applet implements ActionListener
{
    public void init ()
    {
        resize (250, 610);

    }
    public void actionPerformed (ActionEvent e)
    {

    }

}
```

Add an ‘x’ to show where you would add this widget code in the applet.


```
JButton end = new JButton("Game Over!");
end.setBackground(Color.red);
end.setForeground(Color.blue);
add(end);
```

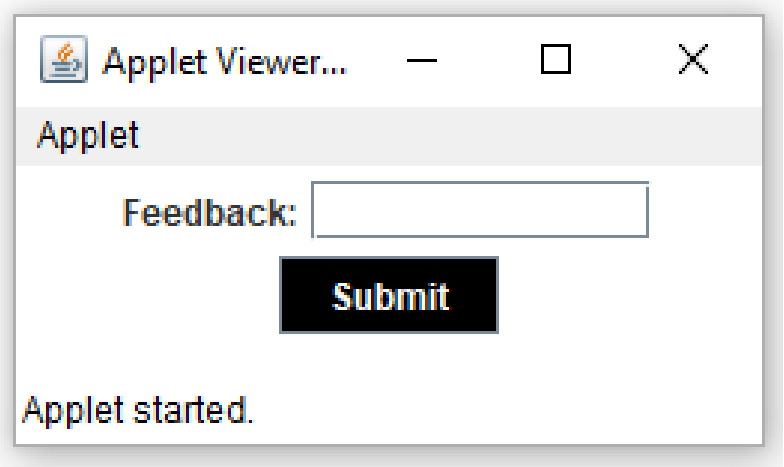
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class Example1 extends Applet implements ActionListener
{
    public void init ()
    {
        resize (250, 610);
    }
    public void actionPerformed (ActionEvent e)
    {

    }

}
```

X

Add it here.

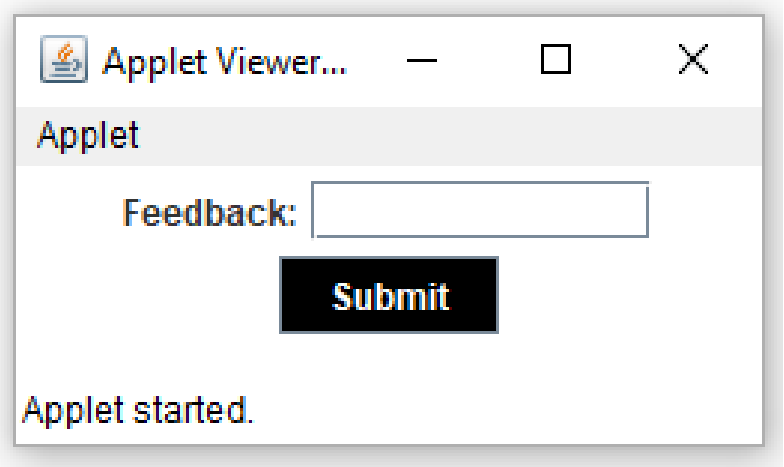

```
JButton end = new JButton("Game Over!");
end.setBackground(Color.red);
end.setForeground(Color.blue);
add(end);
```

Fill in the blanks to make this applet.




```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class Example1 extends Applet implements ActionListener
{
    public void init ()
    {
        resize (250, 70);
        JLabel fb = ______ JLabel("_______________");
        add(_____);
        JTextField tf = new _______________(10);
        add(____);
        JButton b = new ____________("_______________");
        b.setBackground(Color.black);
        b.setForeground(Color._________);
        add(_____);
    }
    public void actionPerformed (ActionEvent e)
    {
    }
}
```

The solution.




```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class Example1 extends Applet implements ActionListener
{
    public void init ()
    {
        resize (250, 300);
        JLabel fb = new JLabel("Feedback:");
        add(fb);
        JTextField tf = new JTextField(10);
        add(tf);
        JButton b = new JButton("Submit");
        b.setBackground(Color.black);
        b.setForeground(Color.white);
        add(b);
    }
    public void actionPerformed (ActionEvent e)
    {

    }
}
```

Fill in the blanks to make this applet.


```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class Halloween extends Applet implements ActionListener
{
     public void init ()
     {
          resize (350, 480);
          JLabel title = new JLabel ("_____________");
          title.setFont(new Font("Showcard Gothic", Font.PLAIN, 24));
          title.setForeground(Color._________);
          add(_____________);
          JLabel pic = new JLabel(____________________("zombie.png"));
          add(_____________);
     }
     public void actionPerformed (ActionEvent e)
     {
     }
     protected static ImageIcon createImageIcon (String path)
     {
          java.net.URL imgURL = _____________.class.getResource (path);
          if (imgURL != null)
               return new ImageIcon (imgURL);
          else
               return null;
     }
}
```

The solution.


```
import javax.swing.*; import java.awt.*;
import java.awt.event.*; import java.applet.Applet;
public class Halloween extends Applet implements ActionListener
{
    public void init ()
    {
        resize (350, 480);
        JLabel title = new JLabel ("Run!");
        title.setFont(new Font("Showcard Gothic", Font.PLAIN, 24));
        title.setForeground(Color.red);
        add(title);
        JLabel pic = new JLabel(createImageIcon("zombie.png"));
        add(pic);
    }
    public void actionPerformed (ActionEvent e)
    {
    }
    protected static ImageIcon createImageIcon (String path)
    {
        java.net.URL imgURL = Halloween.class.getResource (path);
        if (imgURL != null)
            return new ImageIcon (imgURL);
        else
            return null;
    }
}
```

Notes that I would Consider Important

And, you know, I am writing the test.

Compare and Contrast ActionPerformed and Init

	Init	ActionPerformed
How Often It Runs	Once	The user decides.
When it is run	When the applet begins	When the user clicks a button
General Purpose	To set up the screen	To respond to a user's request
Lines of Code Found there	Declaration Construction (new) Mutation (set) Add addActionListener setActionCommand	Mutation (set) Accessors (get) showStatus
Lines of Code NOT found there	Accessors (get) – you can, it is just pointless.	Construction (new) Add

Widget

- A widget is a piece of a user interface
- Users can see widgets, click on them, or type in them.
- They are more complex than simple variables (simple types are ints, doubles, chars). Because of their complex memory structures, we need to set them up in more complex ways and use special methods to edit them.
- The three most common widgets are: JLabel, JTextFields, JButtons

CLI

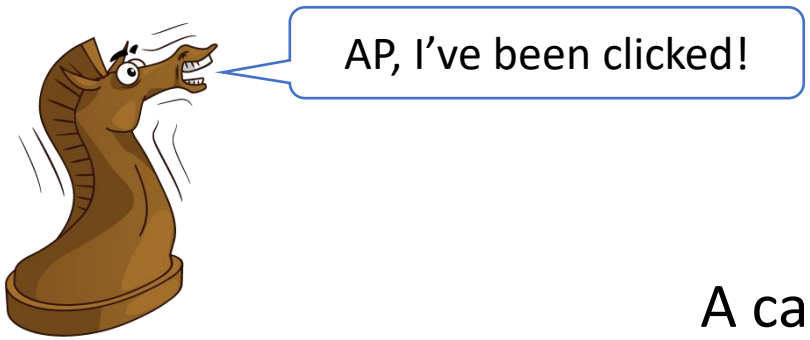
- A command line interface
- They are simple, so they don’t need libraries
- They are computer-driven; the code tells the user what they must do next.
- They input using IO (Keyboards).
- Their output is uses System.out.println and it can’t have colour or real pictures.
- The two main methods are the main method and the constructor.

GUI

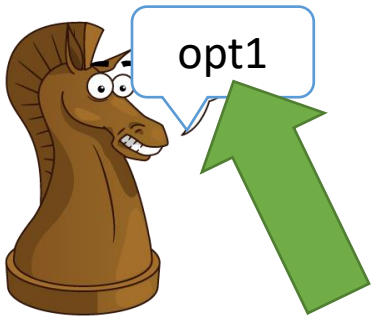
- A graphic user interface
- They are complex, so we draw on libraries for additional code
- They are user-driven; the user can decide what they want to do next.
- They input using JTextFields (Keyboards) and JButtons (mouse)
- Their output is mostly on JLabels and it can have colour and pictures
- The two main methods are init and actionPerformed.

ActionListener

- Is a piece of code set up to watch a button
- It has an actionCommand.
- When a button is pressed, it calls the ActionPerformed method and passes it the ActionCommand



A card for you to write.



ActionCommand

- Is a STRING
- It is a word that is passed from the ActionListener to the ActionPerformed
- It is used by the AP to determine which button was pressed.
- It is like a secret code word that is passed between the AL and the AP.

A card for you to write.

ActionPerformed

- Is a method
- It is called by the ActionListener when a button is pressed.
- It has code to respond to the user's button click and it is used to update the screen.
- It uses the ActionCommand to figure out which button was pressed to run the appropriate code.

A card for you to write.

User Interface Design Summary

5. Write the characteristic of good user interface design in front of the matching description.

(Word Bank: Visually Appealing, Restricts Input, Widget Arrangement, Clear Instructions, Error Handling)

Error Handling		a) Have a reset/undo button to allow the user to start over if they make a mistake. Code if statements to handle invalid data entry.
Widget Arrangement		b) Like items are grouped with like items. Widgets appear in the order the user will need them. This makes it easy to find what you need. Like items should have similar formatting.
Clear Instructions		c) Provide clear titles. Put prompts in front of JTextfields. Have useful information on JButtons. Provide instructions in JLabels.
Visually Appealing		d) Use colour and pictures. Change the fonts. Resize the applet to one that displays the content well. Similar widgets should be the same size.
Restricts Input		e) Uses JButtons instead of JTextfields when possible – users are more likely to mistype than to misclick. Disables buttons when they should not be clicked.

Denise Melanson

- Tragically died in 2006 after receiving an overdose when her chemotherapy pump was mis-programmed to deliver 4 days of medication in 4 hours.
- Hospital inquest decided programmer at fault because a simple if was required to prevent overdose.
- By fixing the device, not blaming the nurse, no more overdoses have been delivered.
- User-centric design solved the problem.

Why User-Centric Design is Best:

- less training needed
- less checks needed downstream (which are not effective anyway)
- accelerates adoption
- increases safety
- improves efficiency

User-centric design = safer, better equipment = happy people = rich designers. :D