
Variables and Methods

Chapter 3 – Lecture Slides

Variables

- Declaration

- Initialization –

- Assignment

- Variable Reference

```
String myName;
```

```
myName = "Java Guru";
```



Scope: Instance Variable vs. Local Variable

Instance Variable

vs.

Local Variable

```
import javax.swing.*;
import java.awt.*;
public class ColorEx extends JApplet
{
    String favColor;
    String ski;
    public void paint( Graphics g )
    {
        favColor = "Favorite color";
        ski = "Love 2 ski";
        g.setColor( Color.RED );
        g.drawString( favColor, 30, 45 );
        g.setColor( new Color( 12,34,52) );
        g.drawString( ski, 30, 53 );
    }
}
```

```
import javax.swing.*;
import java.awt.*;
public class ColorEx extends JApplet
{
    public void paint( Graphics g )
    {
        String favColor = "Favorite color";
        String ski = "Love 2 ski";
        g.setColor( Color.RED );
        g.drawString( favColor, 30, 45 );
        g.setColor( new Color( 12,34,52) );
        g.drawString( ski, 30, 53 );
    }
}
```

Methods

- Example

```
public void paint( Graphics g )
```

- Within the class squigglys { }
 - Has it's own scope squigglys { }
 - Statements in method only run when the method is **called/invoked**
 - Parameters (none or multiple) listed in parenthesis
 -
 -
 -
-

Method call that returns a value

```
import java.awt.*;
import javax.swing.*;
public class Calculate extends JApplet
{
    public void paint (Graphics g )
    {
        int addition = getAdd( 2, 7 );
        String added = "2 + 7 = " + addition;
        g.drawString ( added, 0, 12 );
        String subtracted = "2 - 7 = " + getSubtract( 2, 7 );
        g.drawString ( subtracted, 0, 24 );
    }
    public int getAdd( int num1, int num2 )
    {
        return num1 + num2;
    }
    public int getSubtract( int n1, int n2 )
    {
        return n1 - n2;
    }
}
```

3 Types of Method calling

1. Within a class
2. In another class (type 1)
Example: in the Graphics class

(where g is from the Graphics class,
e.g., `public void paint(Graphics g)`)

3. In another class (type 2)
-

Why Have Methods?

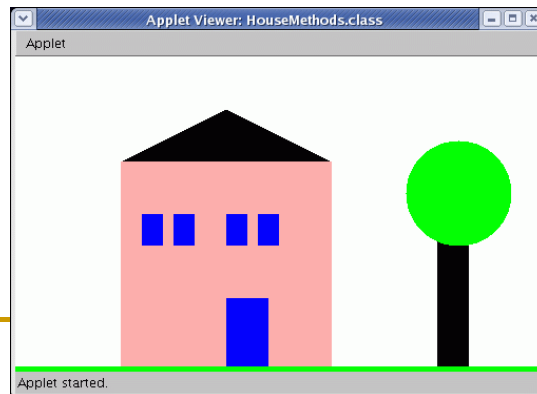
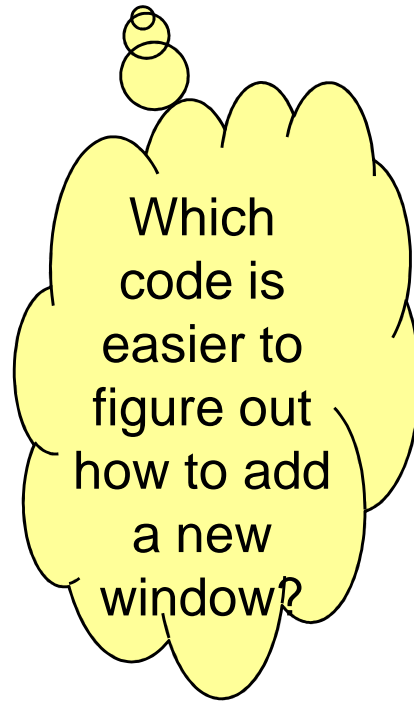
- -
 -
 - Events: (we'll discuss later)
-

Why Have Methods?

```
import java.awt.*;
import javax.swing.*;
public class House extends JApplet
{
    public void paint (Graphics g )
    {
        g.setColor( Color.pink );
        g.fillRect ( 100,100,200,200 );
        g.setColor( Color.black );
        Polygon poly = new Polygon( );
        poly.addPoint(100,100);
        poly.addPoint(200,50);
        poly.addPoint(300,100);
        g.fillPolygon(poly);

        g.setColor( Color.blue );
        g.fillRect ( 200,230,40,70);
        g.fillRect ( 120,150,20,30);
        g.fillRect ( 150,150,20,30);
        g.fillRect ( 200,150,20,30);
        g.fillRect ( 230,150,20,30);

        g.setColor( Color.black );
        g.fillRect ( 400,130,30,170 );
        g.setColor( Color.green );
        g.fillOval( 370,80,100,100 );
        g.fillRect ( 0,295,500,5 );
    }
}
```



// Code Easier to read: with methods

```
import java.awt.*;
import javax.swing.*;
public class HouseMethods extends JApplet
{
    int WINDOW_WIDTH = 20;
    int WINDOW_HEIGHT = 30;
    public void paint (Graphics g ) {
        paintHouse( g );
        paintLandscape( g );
    }
```

public void paintHouse(Graphics grph) {

```
    grph.setColor( Color.pink );
    grph.fillRect ( 100,100,200,200 );
    grph.setColor( Color.black );
    Polygon poly = new Polygon();
    poly.addPoint(100,100);
    poly.addPoint(200,50);
    poly.addPoint(300,100);
    grph.fillPolygon(poly);
```

```
    grph.setColor( Color.blue );
    grph.fillRect ( 200,230,40,70);
    paintWindow( grph, 120, 150 );
    paintWindow( grph, 150, 150 );
    paintWindow( grph, 200, 150 );
    paintWindow( grph, 230, 150 );
```

public void paintWindow(Graphics gp, int x, int y)

```
{
    gp.setColor( Color.blue );
    gp.fillRect ( x, y, WINDOW_WIDTH, WINDOW_HEIGHT );
}
```

public void paintLandscape(Graphics g) {

```
    g.setColor( Color.black ); // tree
    g.fillRect ( 400,130,30,170 );
    g.setColor( Color.green );
    g.fillOval( 370,80,100,100 );
    g.fillRect ( 0,295,500,5 ); // grass
```

```
}
```

Why Have Methods?

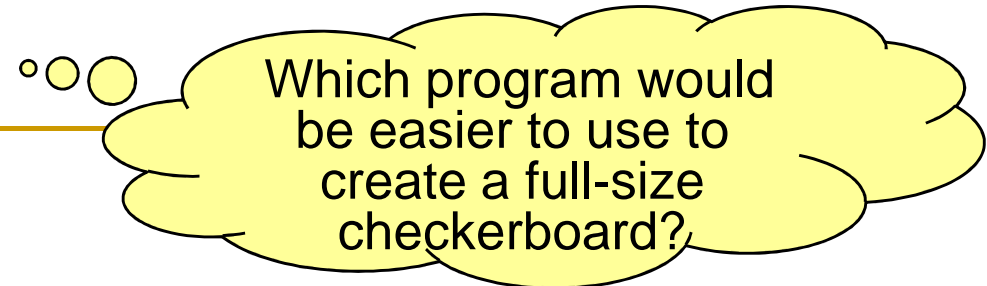
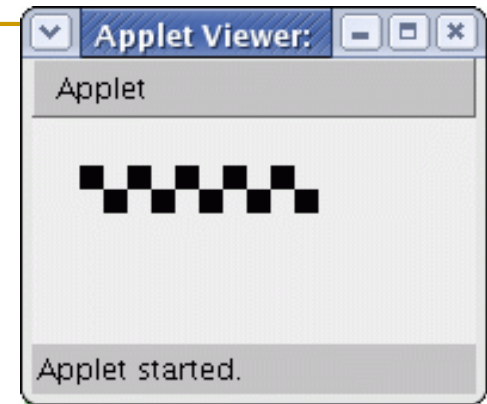
```
import javax.swing.*;  
import java.awt.*;
```

```
public class NeedMethods extends JApplet  
{  
    public void paint( Graphics g )  
    {  
        // row 1  
        g.fillRect( 20,20, 10,10 );  
        g.fillRect( 40,20, 10,10 );  
        g.fillRect( 60,20, 10,10 );  
        g.fillRect( 80,20, 10,10 );  
        g.fillRect( 100,20, 10,10 );  
  
        // row 2  
        g.fillRect( 30,30, 10,10 );  
        g.fillRect( 50,30, 10,10 );  
        g.fillRect( 70,30, 10,10 );  
        g.fillRect( 90,30, 10,10 );  
        g.fillRect( 110,30, 10,10 );  
    }  
}
```

// Repeated code using a method:

```
import javax.swing.*;  
import java.awt.*;
```

```
public class NeedMethods2 extends JApplet  
{  
    public void paint( Graphics g )  
    {  
        drawRows( g, 20, 20 );  
        drawRows( g, 30, 30 );  
    }  
    public void drawRows( Graphics graphics, int x, int y )  
    {  
        graphics.fillRect( x,y, 10,10 );  
        graphics.fillRect( x+20, y, 10, 10 );  
        graphics.fillRect( x+40, y, 10, 10 );  
        graphics.fillRect( x+60, y, 10, 10 );  
        graphics.fillRect( x+80, y, 10, 10 );  
    }  
}
```



Why Have Methods?

- For access by other objects:
 - Graphics gr
 - If these weren't methods, we wouldn't be able to draw anything!
-

Why Have Methods?

- Events
 - Events in Java are triggered when:
 - User selects a button/checkbox/item in list/etc.
 - User moves/draggs the mouse
 - User types a key
 - Timer expires
 - More...
 - Each event automatically calls a particular **method** to handle the type of event that occurred.
 - We'll discuss events in more detail later
-

Summary

- Variables
 - Scope: instance variables vs. local variables
 - Method Structure
 - Purpose of methods
-