**GUI01**

**Directions**

To this point, you have written programs that take input from the keyboard, do some processing, and then display the results on a console screen. Most modern programs, however, don't work this way. This lesson will introduce you to another type of program called **graphical user interface** or **GUI**. In a GUI program you start with a window called a frame and you add graphical components like labels, buttons, textfield, combo boxes, menus, and others. A GUI program is called an event-driven program because it responds to events initiated by the user like pressing a button, typing in a text field, or selecting a menu option.

The code below creates a Java GUI application program and displays its frame window. This code will be the template we will use to create all of our GUI programs.

import java.util.\*;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class GUI01 extends JFrame

{

// constructor

public GUI01()

{

// set frame attributes

setLayout(null); // no layout will be used

setSize(500, 500); // sets size of frame window

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // activate close icon

getContentPane().setBackground(Color.white); // set background color

setVisible(true); // show frame

}

// main method

public static void main(String[] args)

{

GUI01 app = new GUI01(); // run program

}

}

Since we will be using this code in all of our GUI programs lets break it apart and learn what each part does.

import Statement

Let's start with the import command. The import command allows your program access to classes that are not in your project. Java developers have written hundreds of classes to inable you to create GUI programs. Most of the classes fall under what Java developers call the Swing classes and they can be accessed by including the following imports at the top of your program.

import java.util.\*;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

Inheritance

The next line of code is the program's class declaration with the keyword **extends** and the class name JFrame appended to the end. The JFrame class is one of the swing classes and its purpose is to create a GUI program's frame window. By adding the code **extends JFrame** to the end of the class declaration we are utilizing a powerful programming mechanism called **inheritance**. (Inheritance is a topic we will cover in much greater detail in later unit.)

public class GUI01 extends JFrame

The idea of inheritance is simple but powerful: When you want to create a new class and there is already a class that includes some of the code that you want, you can derive your new class from the existing class. In doing this, you can use the methods of the existing class without having to write them yourself.

Writing the code for your own frame window would be a very complex process. However, through inheritance, it becomes fairly trival. All you have to do is extend or inherit the JFrame class and your program automatically has the ability to create a frame window and you also have access to a whole list of methods that can be used to manipulate the attributes of the window. In essence, your class becomes a JFrame class.

Constructor

A **constructor** is a special method that has the same name as the class it is defined in. Below is the declaration for our constructor. Notice the class is named GUI01 and the name of the constructor is also GUI01.

public class GUI01 extends JFrame

{

// constructor

public GUI01()

{

Every GUI program we write will include a constructor. The constructor is important because it will contain the majority of the code we write for our GUI programs.

JFrame Methods

All of the lines of code contained within the constructor reference methods of the JFrame class that became accessible through inheritance.

The first JFrame method is called **setLayout**. Layouts are used to automatically position components within a frame. We will not be using layouts so the line of code below is setting the layout of the frame to null.

setLayout(null);

The second JFrame method called within the contructor is **setSize**. This method sets the initial size of frame when it is displayed on the screen. The line of code below sets the dimension of the frame to 500 pixels across by 500 pixels down. The default size of a frame is (0,0) so you need to use this method to give your frame an initial size.

setSize(500, 500);

The next JFrame method is **setDefaultCloseOperation**. This method is used to determine what happens to the prograrm when a user presses the "x" icon in the upper right hand corner of your frame window. The line of code below causes the program to terminate when the "x" is pressed.

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

The next JFrame method is **setBackground**. This method sets the background color for your frame window. The default frame color is gray. Java has preset colors that can be accessed using the Color class.

getContentPane().setBackground(Color.white);

Notice the reference to the method **getContentPane** in the code above. The frame window within a Java application is actually made up of several layers or panes. To see these layers click on the Expand button below.



In our GUI programs we are only concerned with the content pane layer. To gain access to this pane you use the method getContentPane, which returns a reference to the current frame's content pane object. The line of code

getContentPane().setBackground(Color.white);

is actually changing the background color of the content pane not the frame itself. Later on when we want to add components like buttons or textfields to a frame we will actually be adding them to content pane.

The last JFrame method called within the constructor is setVisible. This method determines whether the frame window is visible when the program is executed.. By default a frame is invisible so you will need to call this method with an argument of true.

setVisible(true);

main Method

Java Application programs must contain a main method. The code below is the main method for our template class.

// main method

public static void main(String[] args)

{

GUI01 app = new GUI01(); // run program

}

The code highlighted in red is a call made to the contructor method discussed above. When the program is executed the constructor is called which builds the frame window and sets all of its attributes like size, color, etc.

Every GUI program will have a main method similar to this one with the appropriate name.

**Exercise**

1. Open your Computer Science workspace.
2. Right-click on the Computer Science workspace icon and select **Add New Project**.
3. Select **Empty Project** and click **Next**.
4. Name the project: **GUI** and select **Finish**.
5. Right-click on your new GUI project icon and select **Add-> New File**.
6. Name the file: **GUI01** and select **Finish**.
7. Copy the code located at the top of the page into the new file.

**Modifications**

Make the following modifications to the GUI01 class.

1. Change the size of the frame to 400 x 300.
2. Change the color of the frame's background to one of the preset colors.

Here is the list of the preset colors.

|  |  |
| --- | --- |
| **Color** | **Command** |
| black | Color.black |
| blue | Color.blue |
| cyan | Color.cyan |
| gray | Color.gray |
| green | Color.green |
| magenta | Color.magenta |
| orange | Color.orange |
| white | Color.white |
| yellow | Color.yellow |

1. Change the color of the frame's background to a custom color of your choosing.

To create a custom color use the following code:

getContentPane().setBackground(new Color(50, 120, 200));

The three numbers represent the RGB value of the color. They must be between 0 and 255 inclusive.

**Source File**

GUI01.java

**Sample Run**

