SWING

IS A JAVA FRAMEWORK USED FOR BUILDING
GUIS (GRAPHICAL USER INTERFACES)

SWING HAS BOTH ADVANTAGES..

IT FOLLOWS SOMETHING CALLED THE MVC PARADIGM,
WHICH MEANS THAT DATA AND ITS GUI REPRESENTATION
ARE DECOUPLED

SWING IS PLATFORM-INDEPENDENT SO A SWING
GUI WILL WORK ON ANY KIND OF MACHINE

..AND DISADVANTAGES

SWING IS SINGLE-THREADED, I.E. ALL
UI IS DRAWN ON A SINGLE THREAD, SO
SWING PROGRAMMERS NEED TO DO
SOME THREADING TO KEEP THE UI
ZIPPY AND RESPONSIVE

BECAUSE SWING IS WRITTEN IN JAVA AND
INTENTIONALLY DOES NOT USE NATIVE
OPERATING SYSTEM FUNCTIONALITY, SWING
USER INTERFACES HAVE A CHARACTERISTIC
"JAVA-Y" LOOK

LIKE ANY FRAMEWORK, SWING IS A COMPLICATED COLLECTION OF INTERCONNECTED CLASSES

JFrame

THIS CLASS CONTAINS THE WINDOW INSIDE WHICH ALL OF THE USER-INTERFACE APPEARS

THE USER KICKS OF THE APPLICATION BY INSTANTIATING JFRAME AND ADDING A FEW WIDGETS (JCOMPONENT OBJECTS) TO THE JFRAME

SWING TAKES CARE OF DISPLAYING THE JFRAME TO SCREEN, OF MAKING SURE THAT MOUSE CLICKS AND KEYSTROKES GET MAPPED TO THE CORRECT JCOMPONENT ETC

JComponent

IS A COMMON BASE CLASS USED FOR MOST UI ELEMENTS - BUTTONS, LABELS, PANELS ETC

THE JCOMPONENT OBJECTS ARE CREATED BY THE PROGRAMMER AND ADDED TO THE JFRAME (I.E. THE JFRAME OBJECT CONTAINS MANY JCOMPONENT OBJECTS)

MENUBARS, BUTTONS, TREEVIEWS, FILE CHOOSERS, ...

LIKE ANY FRAMEWORK, SWING GIVES THE PROGRAMMER GREAT POWER, BUT TAKES AWAY SOME CONTROL

SWING UI-WIDGETS (BUTTONS, LABELS, FILE CHOOSERS, TREE-VIEWS ETC) MAKE IT SURPRISINGLY SIMPLE TO DEVELOP A SOPHISTICATED USER INTERFACE

BUT GETTING ABSOLUTE CONTROL OVER LITTLE THINGS - FOR INSTANCE THE EXACT LOCATION OF A GIVEN UI WIDGET - IS DIFFICULT TO ACHIEVE.

TO USE SWING, A PROGRAMMER MUST DO THE USUAL STUFF

DO THE LITTLE BOILERPLATE STUFF

CREATE A JFRAME OBJECT, DISPLAY IT, ADD COMPONENTS

DRESS THE PART

EXTEND THE BASE JCOMPONENT CLASSES

OVERRIDE THE PAINTCOMPONENT METHOD OF THE JCOMPONENT CLASSES TO DO LOW-LEVEL DRAWING

LISTEN TO THE RIGHT STUFF

SET UP EVENT LISTENERS WITH CODE TO HANDLE BUTTON CLICKS, MOUSE CLICKS, MENU CHOICES

DO THE LITTLE BOILERPLATE STUFF

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

public class Main {

    public static void main(String[] args) {
        JFrame frame = new JFrame("My First Swing App");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Heloo world!");
        frame.getContentPane().add(label);

        frame.setSize(500,500);
        frame.setVisible(true);
    }
}
```

CREATE A JFRAME OBJECT

CREATE A JCOMPONENT

ADD THE JCOMPONENT TO JFRAME

MAKE THE FRAME VISIBLE

NOTICE ALREADY HOW MUCH SWING HAS DONE FOR US

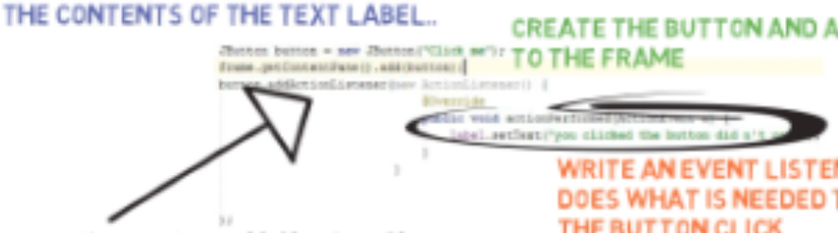
IF THAT DOES NOT SEEM LIKE MUCH, TRY GETTING THIS FAR FROM SCRATCH (OR CHAT UP A 1990S UI PROGRAMMER)

THE WINDOW APPEARS, WELL-FORMED AND CONTAINING THE LABEL

THE WINDOW SITS THERE WAITING UNTIL WE CLOSE IT, AT WHICH POINT OUR JAVA PROGRAM NEATLY EXITS

LISTEN TO THE RIGHT STUFF

TO ADD A BUTTON WHICH CHANGES
THE CONTENTS OF THE TEXT LABEL..



CREATE THE BUTTON AND ADD IT
TO THE FRAME

WRITE AN EVENT LISTENER THAT
DOES WHAT IS NEEDED TO HANDLE
THE BUTTON CLICK

REGISTER YOUR EVENT LISTENER
SO THAT IT LISTENS TO THE BUTTON
AND IS NOTIFIED WHEN THE BUTTON
IS CLICKED

DRESS THE PART

SWING, LIKE ALL FRAMEWORKS,
NEEDS A LOT INTERFACES TO BE
IMPLEMENTED, AND PRE-EXISTING
CLASS METHODS TO BE OVERRIDDEN.

NOTICE HOW WE HAD TO ELABORATELY
WRAP UP OUR ONE LINE EVENT LISTENER
CODE INTO ALL THIS BOILERPLATE, SO
THAT THE BUTTON WAS SATISFIED THAT
OUR EVENTLISTENER IMPLEMENTS THE
"ACTIONLISTENER" INTERFACE

NOTICE HOW WE HAD TO ELABORATELY WRAP UP OUR ONE LINE EVENT LISTENER CODE INTO ALL THIS BOILERPLATE, SO THAT THE BUTTON WAS SATISFIED THAT OUR EVENTLISTENER IMPLEMENTS THE "ACTIONLISTENER" INTERFACE