Graphical User Interfaces 1

- In this section:
  - Building a program with a window…
  - … containing GUI "widgets"
  - Linking the widgets to event handlers
  - Customizing the appearance

GUIs in Java

- Most Java GUI designers use Java's "Swing" libraries
  - These followed the original basic AWT libraries
  - AWT = Abstract Window Toolkit
  - Some parts of the AWT are still important
- There are many ways to build/organize GUIs in Java
  - Even just using Swing
  - Some ways very complex (but good in large applications)
  - We will use a simple approach (not so good in large applications) based on *Java for Students, by D Bell & M Parr, Prentice Hall*

GUIs in Java

- Our GUI applications will need to import a range of libraries to include both AWT and Swing components

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

  - Library components are called "classes" – e.g. **`JButton`**
  - Many provide the GUI widgets
- Each class contains (many) methods that we can call to control the widget
  - For example to change the font it uses
  - Frequent reference to official documentation is vital!

The Car Park application

Summary:

- There are two GUI buttons
  - Java **`JButton`**s
  - Called **`enter`** and **`exit`**
- And one Java **`JTextField`** called **`text`**
- Whichever button is clicked, the event handler method **`actionPerformed`** is called
  - It needs to check the *source* of the event…
  - … then carries out the correct action
  - Either incrementing or decrementing the counter and updating the display

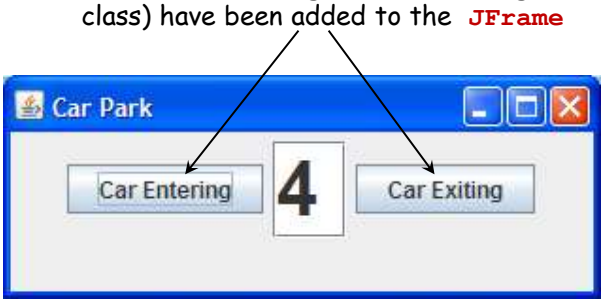
The GUI structure

The whole window/program is a specialized **`JFrame`** (a Swing class)

- **`JFrame`**s have a title, position, size
- They can have other components added to them
  - With automatic layout rules

The GUI structure

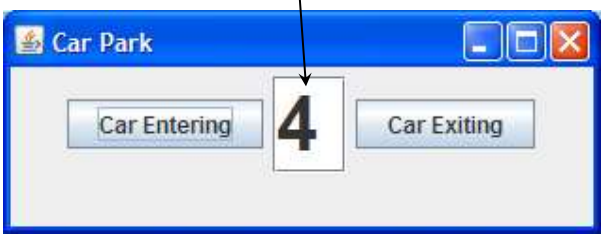
Two **`JButton`** widgets (also a Swing class) have been added to the **`JFrame`**



- **`JButton`**s have a text caption, or image
- The font can be set: face, style, size, colour
- They can be the source of **`ActionEvent`**s
  - Causing **`actionPerformed`** to be called

The GUI structure

One `JTextField` widget (also a Swing class) has been added to the `JFrame`



- `JTextField`s have text that they display
  - Which can be changed at any time by the program or user
  - And inspected at any time by the program
- The width is usually set at launch time
- The font can be set: face, style, size, colour
- `JTextField`s can be the source of `ActionEvent`s

The `JFrame`

- The `main` method launches the program
- It coordinates:
  - Creation of the frame
  - Adding the widgets
  - Position, size and visibility of the frame
- For the Car Park:

```
public static void main(String[] args) {
   CarPark frame = new CarPark();
   frame.setSize(300, 200);
   frame.setLocation(150, 150);
   frame.setTitle("Car Park");
   frame.createGUI();
   frame.setVisible(true);
}
```

`JFrame` library `set` methods

Helper method in *this* program

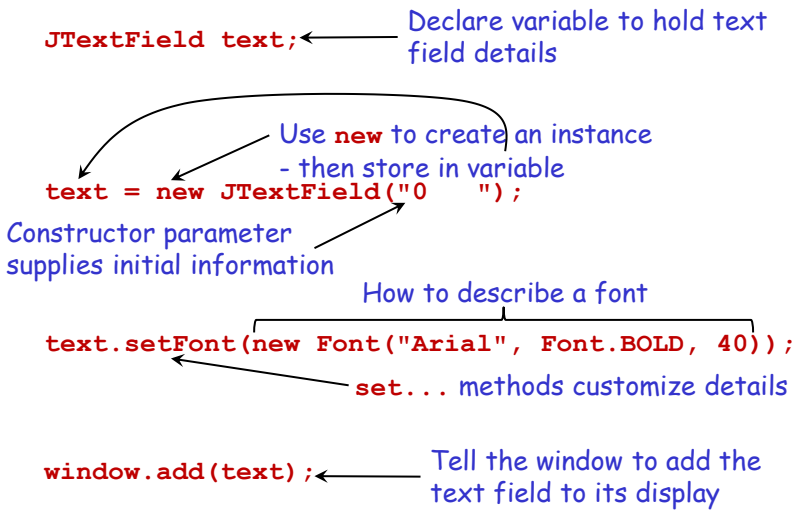
Creating widgets: `new`

- The library *classes* `JFrame`, `JButton`, `JTextField` all contain "template" code for the appearance and behaviour of the widget
- In order to actually *use* a widget, we must create an "instance" using the keyword `new`
  - The same as for arrays
  - This is an object orientation concept ("instance" = "object")
- … and *save/store* the details for the widget instance in a suitable variable
- Typically we instantiate, customize then display:

```
JTextField text;
text = new JTextField("0    ");
text.setFont(new Font("Arial", Font.BOLD, 40));
window.add(text);
```


In more detail

- Typically we instantiate, customize then display:



`createGUI` - details

```
private void createGUI() {
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    Container window = getContentPane();
    window.setLayout(new FlowLayout());

    enter = new JButton("Car Entering");
    window.add(enter);
    enter.addActionListener(this);

    text = new JTextField("0   ");
    text.setFont(new Font("Arial", Font.BOLD, 40));
    window.add(text);

    exit = new JButton("Car Exiting");
    window.add(exit);
    exit.addActionListener(this);
}
```

Overall window set up

Connect the event handling

The interactive event handling: `actionPerformed`

`main` and `createGUI` launch the program
Then the JVM waits for a button click event
It calls `actionPerformed` *every time* a button is clicked:

```
public void actionPerformed(ActionEvent event)
{
    if (event.getSource() == enter)
    {
        carCount = carCount + 1;
    }
    if (event.getSource() == exit)
    {
        carCount = carCount-1;
    }
    text.setText(Integer.toString(carCount));
}
```

The button variable identifiers

Finally, the whole program (on one slide)

```
import ...

public class CarPark extends JFrame
                     implements ActionListener
{
    private int carCount = 0;
    private JButton enter, exit;
    private JTextField text;

    public static void main(String[] args)
    ...

    private void createGUI()
    ...

    public void actionPerformed(ActionEvent event)
    ...

}
```

Required for event handling

"Global" decls

Global so persistent across & between method calls

About `FlowLayout`

- When we `add` widgets, the JVM decides where they go
- The `createGUI` method sets "`FlowLayout`"

```
window.setLayout(new FlowLayout());
```

  - Widgets are placed centrally
  - From left to right
  - And in rows from the top
  - And *will be rearranged automatically if the window size changes*
- Three lines: `window.add(enter)` … `text` … `exit`
  - These determine the *order* in which the widgets appear in the window's layout:

enter	text	exit

- There is a wide range of other layout managers…

End of section