Containers: JFrame and JPanel

- **Import packages**
  - When writing **Java Graphical Applications**, you need to import these **packages**

    - `java.awt.*` : the **abstract window toolkit** - the first generation Java API
    - `javax.swing.*` : the **"swing" extension toolkit** - provides additional functionality on top of AWT

  - **Classes** that start with **J...** are **swing classes**

- **Containers: used to *organize* GUI components**
  - **Containers:**

    - **JFrame** = a **heavy weight** container used as the **top-level window**
    - **JPanel** = a **light weight** container used to ***organize* GUI components**

  - **How to** use these **containers**:

    - Various **GUI components** are **stuck (added)** on to one or more **JPanels**
    - Then the **JPanels** are **stuck (added)** onto the **JFrame**

- **The JFrame class**
  - **Creating** a **JFrame** object:

```
JFrame f = new JFrame("Title of the Window");
```

Note:

When a **JFrame** is **created**, it is ***not* painted (= visible)** !!!

  - Some **important methods** on a **JFrame**:

        - `setSize(width, height)` : set the **display size** of the **frame (= window)**
        - `setVisible(true)` : make the **frame (= window) visible**

  - **Example:** an empty **window**

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

public class Frame1
{
   public static void main(String[] args)
   {
      JFrame f = new JFrame("My First GUI");  // Create Frame

      f.setSize(400,300); // Set size of frame
      f.setVisible(true); // Show the window
   }
}
```

  - **Example Program:** (Demo above code)

    Prog file: click here

Example

- **Adding a GUI component (label) to JFrame**
  - **Label**:

        - A **label** is a **box with some text**
        - We will use a **label** to write the traditional **first** program: the **"Hello World" program**

  - **Creating** a **Label**:

```
JLabel L = new JLabel("Text");
```

  - **Adding** a **GUI object** onto a **JFrame**:

```
JLabel L = new JLabel("Text");

JFrame f = new JFrame("Window Title");

f.getContentPane().add( L );
```

    Example:

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

public class Frame2
{
   public static void main(String[] args)
   {
      JFrame f = new JFrame("My First GUI");

      f.setSize(400,300);

      JLabel L = new JLabel("Hello World !");

      f.getContentPane().add( L );

      f.setVisible(true);
   }
}
```

- **Example Program:** (Demo above code)
  Prog file: click here

Example

- **The JPanel class**
  - **JPanel:**
    - A **JPanel** object is a **lightweight (simple) container** to **hold graphical components**
    - I like to imagine a **JPanel** as a **"post-it" sticker (of *any size*).**
  - **Usage of JPanels**:
    - Hold other **"normal" graphical components** (such as labels, buttons, etc)
    - Hold ***other* JPanels** !!!
      Yep, you can **stick** a **JPanel** on to ***another* JPanel**

      (Just imagine putting a smaller post-it sticker onto a larger one....)

      It lets you **organize** other windows in the window
  - **Creating a JPanel:**

```
JPanel MyPanel =  new JPanel();
```

  - **Adding** a **graphical object** on to a **JPanel**:

```
JLabel L = new JLabel("Hello World !");

JPanel P =  new JPanel();

P.add(L);
```

Note:

> To **display** the **JPanel**, the **JPanel** must be **added** on to the **JFrame** !!!

- **Example:**

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

public class Frame4a
{
   public static void main(String[] args)
   {
  JFrame f = new JFrame("JFrame with a JPanel");

  JLabel L = new JLabel("Hello World !");   // Make a JLabel;
  JPanel P = new JPanel();                  // Make a JPanel;

  P.add(L);                    // Add lable L to JPanel P

  f.getContentPane().add(P);   // Add panel P to JFrame f

  f.setSize(400,300);
  f.setVisible(true);
   }
}
```

- **Example Program:** (Demo above code)

  Prog file: click here

  Example

  How to run the program:

  - **Right click** on link(s) and **save** in a scratch directory
  - To compile: `javac Frame4a.java`
  - To run: `java Frame4a`

- ***Why use JPanels ?***

- JPanels allow you to **group related GUI components** inside **one JPanel**
- By organizing **related components** within **one single JPanel** you can **re-arrange** the **individual JPanels** on the **JFrame** *later*.

  When you **re-position** a JPanel in the **JFrame**, you will move the **entire group of *related components***

- **Therefore:**

  The **JPanel** is **ideally suited** for designing layout of the **GUI**