מנשק משתמש גרפי ב Java
GUI with AWT

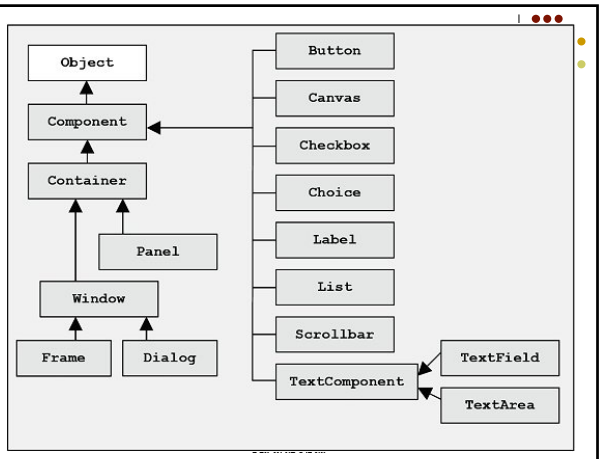
תכנות מתקדם בשפת Java

אוהד ברזילי

אוניברסיטת תל אביב

Abstract Window Toolkit (AWT)

- ספריית AWT מספקת את הרכיבים הגרפיים היסודיים שבשימוש כל יישומי ויישומוני Java
- כל הרכיבים בסיסיים וקל מאוד **להרכיב** אותם או **לרשת** מהם לצורך שימושים ספציפיים
- כל הרכיבים המוצגים על המסך יורשים מהמחלקה Component (או MenuComponent)
- מחלקה חשובה היא המחלקה המופשטת Container היכולה להכיל כמה רכיבים
  - שתי מחלקות היורשות מ Container הן Panel ו- Window



מיכלים (Container)

- ניתן להוסיף רכיב למיכל ע"י המתודה add()
- שני המכלים המרכזיים הם Panel ו- Window
  - Window: חלון עצמאי (top level)
  - Panel: חי בהקשר של מיכל אחר כגון Window או Applet
- ניתן למקם רכיבים בתוך מיכל ע"י המתודות: setLocation(), setSize() ו- setBounds()
- ואולם בדרך כלל נעדיף את הסדרן (Layout Manager) שימקם את הרכיבים במקומנו

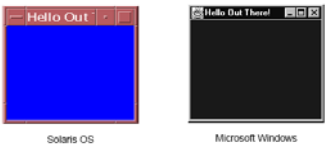
Frame

- המיכל הבסיסי איתו נעבוד יהיה Frame (שיורש מ Window)
- ל Frame המאפיינים הבאים:
  - יש לו כותרת, וניתן לשלוט בגודלו ע"י גרירת פינותיו
  - כברירת מחדל הוא מוסתר, וכדי להציגו יש לקרוא למתודה setVisible(true)
  - מגיע עם סדרן ברירת מחדל מסוג BorderLayout
  - ניתן להחליף את הסדרן ע"י שימוש במתודה setLayout

שלום עולם

```
import java.awt.*;

public class FrameExample1 {

  public static void main(String args[]) {
      Frame f =  new Frame("Hello Out There!");
      f.setSize(200, 170);
      f.setBackground(Color.blue);
      f.setVisible(true);
  }
}
```

Cross platform



- הקוד מהשקף הקודם אינו מעודד שימוש חוזר
- ניצור מחלקה בשם FrameExample2 ונתאים אותה לצרכינו


```
public class FrameExample2 {
    private Frame f;

    public FrameExample2() {
        f = new Frame("Hello Out There!");
    }

    public void launchFrame() {
        f.setSize(170, 170);
        f.setBackground(Color.blue);
        f.setVisible(true);
    }
}
```

```
public class Client {
    public static void main(String args[]) {
        FrameExample2 guiWindow = new FrameExample2();
        guiWindow.launchFrame();
    }
}
```



```
public class FrameExample3 extends Frame {

    public FrameExample3() {
        super("Hello Out There!");
    }

    public void launchFrame() {
        setSize(170, 170);
        setBackground(Color.blue);
        setVisible(true);
    }
}
```

```
public class Client {

    public static void main(String args[]) {
        FrameExample3 guiWindow = new FrameExample3();
        guiWindow.launchFrame();
    }
}
```


Panel

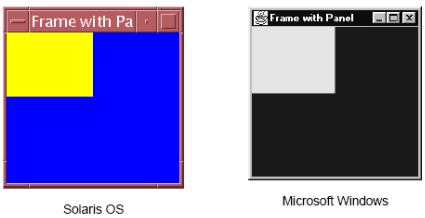
- המחלקה Panel מהווה גם היא מיכל/משטח לרכיבים
- בעזרת שילוב של Panel ו- Frame ניתן לשלוט ביתר עדינות בסידור הרכיבים על המסך

```
public class FrameWithPanel {
    private Frame f;
    private Panel pan;

    public FrameWithPanel(String title) {
        f = new Frame(title);
        pan = new Panel();
    }
```


FrameWithPanel

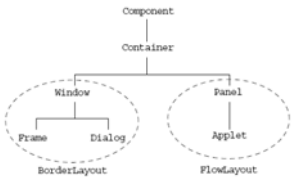
```
    public void launchFrame() {
        f.setSize(200, 200);
        f.setBackground(Color.blue);
        f.setLayout(null); // Use default layout

        pan.setSize(100, 100);
        pan.setBackground(Color.yellow);
        f.add(pan);
        f.setVisible(true);
    }

    public static void main(String args[]) {
        FrameWithPanel guiWindow = new FrameWithPanel("Frame with Panel");
        guiWindow.launchFrame();
    }
}
```


FrameWithPanel



סדרנים

- ב AWT 5 סדרנים:
  - FlowLayout, BorderLayout, GridLayout, CardLayout, GridBagLayout
- FlowLayout, BorderLayout משמשים כסדרני ברירת מחדל למיכלים השונים

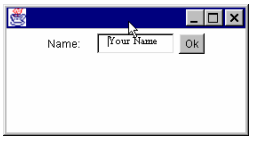


סדרנים

- הסדרנים מסדרים את הרכיבים לפי אסטרטגיה קבועה מראש
- נעזרים במתודות Component:
  - `Dimension getPreferredSize()`
  - `Dimension getMinimumSize()`
  - `Dimension getMaximumSize()`

FlowLayout

- הפשוט ביותר
- מסדר את הרכיבים משמאל לימין, שורה אחר שורה לפי גודלם המועדף
- הרכיבים ממורכזים

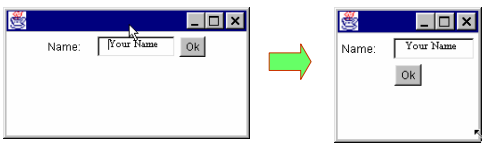



```
public class FlowTest {

   private Frame f;

   public FlowTest() {
        f = new Frame();
   }

   public void draw(){
        f.setLayout(new FlowLayout());
        f.add(new Label("Name:"));
        f.add(new TextField(10));
        f.add(new Button("Ok"));
        f.pack();
        f.setVisible(true);
   }
}
```

```
public class Client {
   public static void main(String[] args) {
        FlowTest f = new FlowTest();
        f.draw();
   }
}
```


FlowLayout

- מיקום הרכיבים מתעדכן עם השתנות גודל החלון
- ניתן לכוונן את הסדרן ע"י שינוי תכונות היישור (שמאל, ימין, מרכז) והריווח



GridLayout

- סדרן טבלאי
- מסדר את הרכיבים משמאל לימין ומלמעלה למטה בצורת טבלה (grid)
- כל הכניסות בטבלה שוות גודל

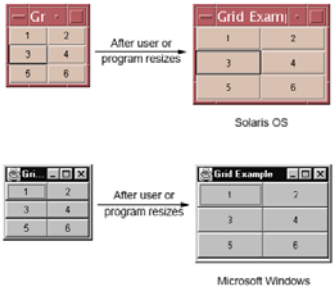
GridLayout

```
public class GridExample {

  private Frame f;
  private Button b1, b2, b3, b4, b5, b6;

  public GridExample() {
      f = new Frame("Grid Example");
      b1 = new Button("1");
      b2 = new Button("2");
      b3 = new Button("3");
      b4 = new Button("4");
      b5 = new Button("5");
      b6 = new Button("6");
  }
```


GridLayout

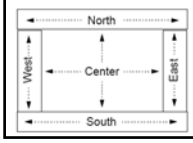
```
  public void launchFrame() {
      f.setLayout(new GridLayout(3, 2));
      f.add(b1);
      f.add(b2);
      f.add(b3);
      f.add(b4);
      f.add(b5);
      f.add(b6);
      f.pack();
      f.setVisible(true);
  }

  public static void main(String args[]) {
      GridExample grid = new GridExample();
      grid.launchFrame();
  }
}
```


GridLayout



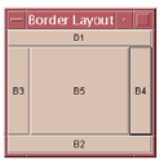
BorderLayout

- סדרן ברירת המחדל של Frame
- כל רכיב מתווסף לאזור (region) מסוים:
  - צפון, דרום, מזרח, מערב או מרכז
- הרכיבים תופסים את כל גודל האזור שלהם:
  - צפון, דרום, מרכז: מתפשטים אופקית
  - מזרח, מערב, מרכז: מתפשטים אנכית



BorderExample

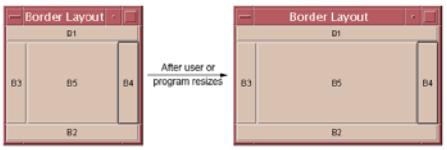
```
public class BorderExample {
  private Frame f;

  private Button bn, bs, bw, be, bc;

  public BorderExample() {
      f = new Frame("Border Layout");
      bn = new Button("B1");
      bs = new Button("B2");
      bw = new Button("B3");
      be = new Button("B4");
      bc = new Button("B5");
  }
```



BorderExample

```
  public void launchFrame() {
      f.add(bn, BorderLayout.NORTH);
      f.add(bs, BorderLayout.SOUTH);
      f.add(bw, BorderLayout.WEST);
      f.add(be, BorderLayout.EAST);
      f.add(bc, BorderLayout.CENTER);
      f.setSize(200, 200);
      f.setVisible(true);
  }

  public static void main(String args[]) {
      BorderExample guiWindow2 = new BorderExample();
      guiWindow2.launchFrame();
  }
}
```

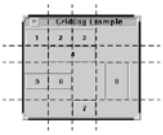
Resizing BorderExample



- בעיה: כל אחד מהאזורים יכול להכיל רק רכיב אחד (רכיב נוסף באזור מסוים דורס את הרכיב שהיה שם לפניו)
- פתרון: מיכל הוא גם רכיב בעצמו. נוסיף Panel בתור רכיב

סדרנים נוספים

- CardLayout
  - הרכיבים מסתירים זה את זה
  - ניתן לדפדף בין רכיבים (ע"י לחיצות עכבר למשל)
- GridBagLayout
  - טבלה אשר התאים בה לא בהכרח באותו הגודל
  - הרכיבים בכל תא לא חייבים למלא את שטח התא



Combination of layouts

```
setLayout(new BorderLaout());
TextField display = new TextField();
add(display, BorderLayout.NORTH);
Panel buttonsPanel = new Panel();
add(buttonsPanel , BorderLayout.CENTER);
buttonsPanel.setLayout(new GridLayout(4,4));
String[] labels = {"7", "8", "9", "+", "4", ... };
for (int i = 0; i < labels.length; i++)
    buttonsPanel.add(new Button(labels[i]));
```

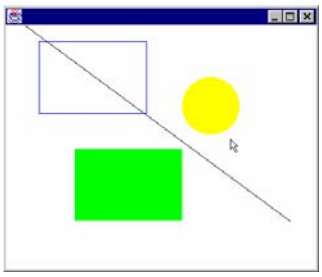

ציור על גבי רכיבי AWT

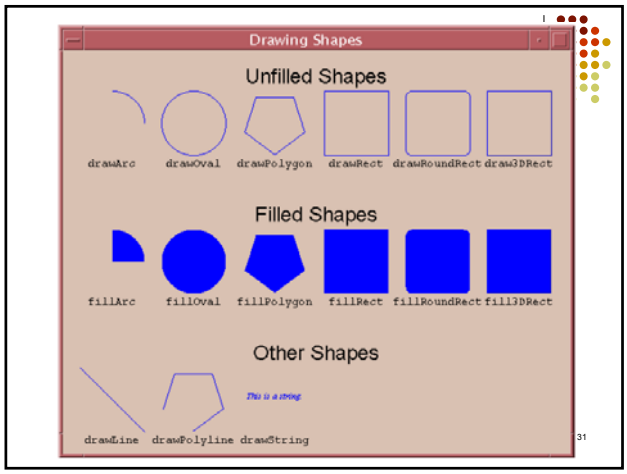
- בכל פעם שיש להציג רכיב AWT על המסך נקראת מתודת ה paint של אותו רכיב
- כדי לצייר על רכיב, בדרך כלל נירש מהמחלקה Panel או Canvas ונדרוס את המתודה ()paint
- המתודה מקבלת כארגומנט עצם מטיפוס Graphics המייצג את הציור על גבי הרכיב
- למחלקה Graphics מתודות רבות לציור

ציור בעזרת Graphics

```
import java.awt.*;

class GraphicsExample extends Frame {
      public void paint(Graphics painter) {
            painter.setColor(Color.black);
            painter.drawLine(20,20,400,300);
            painter.setColor(Color.blue);
            painter.drawRect(50,50,150,100);
            painter.setColor(Color.yellow);
            painter.fillOval(250,100,80,80);
            painter.setColor(Color.green);
            painter.fillRect(100,200,150,100);
      }
}
```


ציור בעזרת Graphics




Drawing Shapes
Unfilled Shapes
drawArc
drawOval
drawPolygon
drawRect
drawRoundRect
draw3DRect
Filled Shapes
fillArc
fillOval
fillPolygon
fillRect
fillRoundRect
fill3DRect
Other Shapes
This is a string
drawLine
drawPolyline
drawString
31