

# פיתוח מערכות תוכנה

## MVC

אוהד ברזילי

[ohadbr@mta.ac.il](mailto:ohadbr@mta.ac.il)

המכללה האקדמית תל אביב – יפו

# MVC

## Model View Controller Design Pattern (MVC) ■

רעיון ההפרדה בין מודל לתצוגה אינו ייחודי רק ל GUI ■  
לדוגמא: ■

מה תדפיס השורה: `System.out.println(new Date());`  
איך מיוצגת המחלקה Date בזיכרון?

נדון בנושא בכמה הקשרים: ■

■ עיצוב ספריות GUI

■ רמות הפשטה שונות של רכיבי GUI

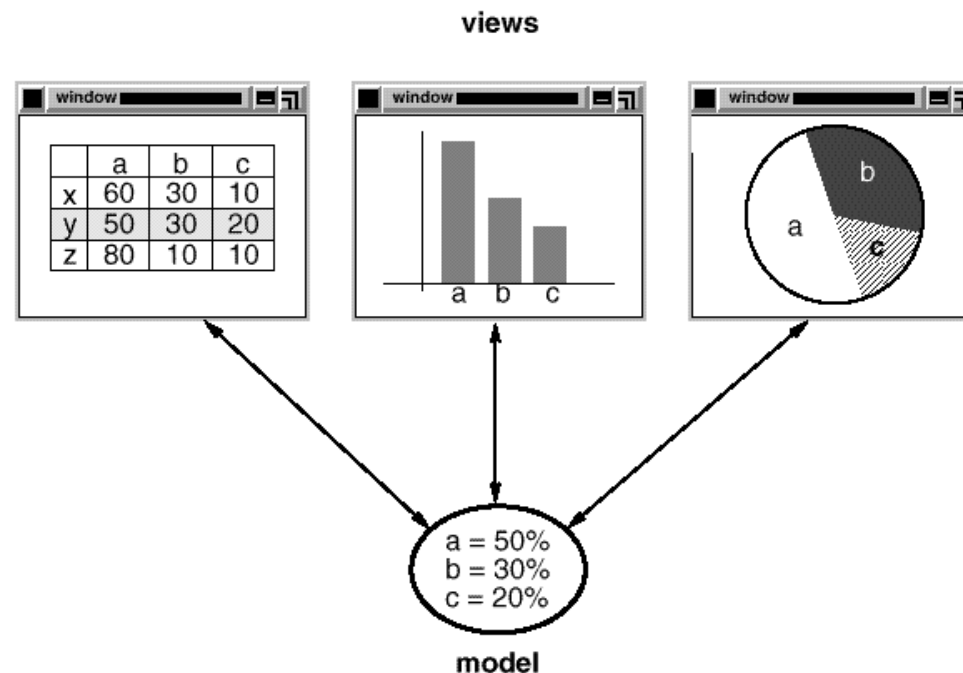
■ Builder Design Pattern

■ CSS / HTML

■ שפות וסביבות מבוססות MVC

# הפרדה בין מודל והצגה

- המודל (הנתונים והלוגיקה של התוכנית) אמור להיות אדיש לשינויים בהצגה (ואולי לאפשר ריבוי הצגות במקביל)
- ספריות GUI רבות נכתבו מתוך הבנת חשיבות ההפרדה





# חזרה על העקרונות המרכזיים ספריות GUI

# הנדסת מנשקי אנוש

---

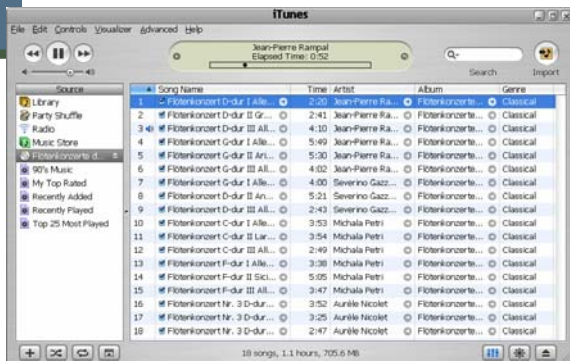
- אינטואיטיביות
- המשתמש/ת בשליטה, לא המחשב
- יעילות של המשתמש, לא של המחשב
- התאמה לתכיפות השימוש וללימוד התוכנה
- שיהיה יפה

# עיצוב מנשקים

- קונסיסטנטיות
- קונטרסט
- ארגון ברור של המסך
- כיוון וסדר ברורים
- העיצוב צריך להתאים את עצמו לסביבה

# שלושת הצירים של תוכנה גרפית

- אלמנטים מסוגים שונים על המסך (היררכיה של טיפוסים)
- הארגון הדו-מימדי של האלמנטים, בדרך כלל בעזרת מיכלים וסדרנים
- ההתנהגות הדינמית של האלמנטים בתגובה לפעולות של המשתמש/ת ("ארועים": הקלדה, הקלקה, גרירה). לרוב ע"י מימוש תבנית ה Observer (נקרא בדרך כלל Listener)



# ספריות גרפיות ב Java

■ כחלק מהספרייה הסטנדרטית מכילה הפצת Java את החבילה `java.awt` המספקת שרותי GUI בסיסיים:

■ **Abstract Windowing Toolkit**

■ בעיית המכנה המשותף הנמוך ביותר

■ יעיל, יביל, מכוער

■ בגרסאות מאוחרות של Java התווספה ספריית `javax.swing` המספקת שרותי GUI מתקדמים:

■ JFC/Swing

■ Look & Feel

■ עשיר, איטי, כבד, מכוער (שנוי במחלוקת)

■ ספריית SWT של IBM מנסה לרקוד על שתי החתונות

■ גם יפה גם אופה

■ המנשק הגרפי של Eclipse מבוסס SWT

■ אינו סטנדרטי (יש להוריד כ zip נפרד)



# Look and Feel

- מערכות הפעלה עם ממשק גרפי מספקות שירותי ממשק (למשל, Windows ו-MacOS; אבל לא לינוקס ויוניקס)
- שימוש בממשקים של מערכת ההפעלה תורם למראה אחיד ולקונסיסטנטיות עם ציפיות המשתמש ועם קביעת התצורה שלו (אם יש דרך לשלוט על מראה הרכיבים, כמו בחלונות)
- ספריות ממשקים משתמשות באחת משתי דרכים על מנת להשיג **אחידות** עם הממשקים של מערכת ההפעלה
  - שימוש ישיר ברכיבי ממשק של מערכת ההפעלה; AWT, SWT
  - אמולציה של התנהגות מערכת ההפעלה אבל כמעט ללא שימוש ברכיבי הממשק שלה (פרט לחלונות); למשל Swing, JFace, Qt; זה מאפשר להחליף מראה, pluggable look & feel

# אלגוריתמי אריזה ב SWT

- **FillLayout**: רכיבים בשורה/עמודה, גודל אחיד לכולם
- **RowLayout**: רכיבים בשורה/עמודה, עם אפשרות שבירה למספר שורות/עמודות, ועם יכולת לקבוע רוחב/גובה לרכיבים
- **GridLayout**: כפי שראינו, סריג שניתן לקבוע בו איזה שורות ועמודות ימתחו ואיזה לא, ולקבוע רוחב/גובה לרכיבים
- **FormLayout**: מיקום בעזרת אילוצים על ארבעת הקצוות (או חלקם) של הרכיבים; אילוצים יחסיים או אבסולוטיים ביחס למיכל (למשל, באמצע רוחבו ועוד 4 פיקסלים) או אילוצים אבסולוטיים ביחס לנקודת קצה של רכיב אחר (דבוק לרכיב אחר או דבוק עם הפרדה של מספר פיקסלים נתון)
- **StackLayout**: ערימה של מיכלים בגודל זהה אבל רק העליון נראה; שימושי להחלפה של תוכן מיכל או חלון

# הרכבה של Composites

Instructions:

1. Fill in your name
2. Fill in your age
3. Fill in your gender
4. Check the box for employment
5. Click on OK

Name:

Age:

Gender:

Have you been employed in the past six months?

Address Phone Numbers Credit Cards Organizations Cancel

OK

■ כדי לבנות בצורה מודולרית מסכים מורכבים (ולפתח כל איזור בנפרד) – רצוי להשתמש במחלקה Composite (מקבילה למחלקה J/Panel ב-Swing/AWT)

■ בדוגמא שלפנינו ה Shell מכיל 3 Composites שונים, כל אחד מהם מנוהל ע"י מנהל פריסה משלו

# עשה זאת בעצמך

- ניתן לצייר על רכיבי GUI (להבדיל מלהוסיף רכיבים מוכנים)
- כדי שהציור ישמור על עיקביותו גם לאחר ארועי חשיפה (שינוי גודל החלון, הסתרת/מזעור החלון ע"י חלונות אחרים) יש לדאוג לציור מחדש לאחר כל ארוע כזה
- לשם כך נכתוב את פונקצית הציור כשגרת הטיפול בארועי ציור
- השגרה מקבלת כארגומנט ארוע ציור `PaintEvent` אשר ניתן לחלץ ממנו הפנייה להקשר הגרפי (`GC - Graphics Context`)

# עשה זאת בעצמך (SWT)

■ נצייר על GC ע"י שימוש בשרות drawXXX הכולל את (רשימה חלקית):

- void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
- void drawFocus(int x, int y, int width, int height)
- void drawImage(Image image, int x, int y)
- void drawLine(int x1, int y1, int x2, int y2)
- void drawOval(int x, int y, int width, int height)
- void drawPath(Path path)
- void drawPoint(int x, int y)
- void drawPolygon(int[] pointArray)
- void drawRectangle(int x, int y, int width, int height)
- void drawRoundRectangle(int x, int y, int width, int height, int arcWidth, int arcHeight)
- void drawString(String string, int x, int y)
- void drawText(String string, int x, int y)

## 2 חלופות

■ איך נתכנן משחק שח גרפי?

■ עשה זאת בעצמך:

■ ציור של משבצות שחור-לבן

■ לכידה של ארועי לחיצה על העכבר

■ שימוש ב widgets:

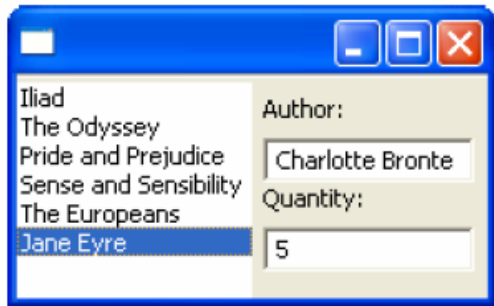
■ בניית סריג של כפתורים ריבועיים בצבעי שחור ולבן  
לסרוגין

■ לכידה של ארועי בחירת כפתור

■ מה היתרונות והחסרונות של כל אחת מהגישות?

# רכיבים המפרידים בין מודל ותצוגה

- חלק מספריות ה GUI מספקות הפרדה בין מודל ותצוגה גם ע"י רכיבים ייעודיים
- דוגמא: Java Beans הן מחלקות Java רגילות אשר יודעות לדווח ל"עולם" על ארועים שונים (fire event)
- דוגמא נוספת:
  - טבלה המצוירת על המסך מכילה מחרוזות ואולם בעצם היא מתארת עצם סמנטי שזהו רק הייצוג שלו
  - היה נחמד לו ניתן היה להציג בצורה אוטומטית Map בתור טבלה ויזואלית



# JFace Viewers

■ החבילה JFace מציעה מגוון מחלקות המציעות שרותי GUI מתקדמים הכתובים מעל (בעזרת) הספרייה SWT

■ אחת המשפחות בחבילה מכילה הצגות למבני נתונים שימושיים כגון: `CheckboxTableViewer`, `CheckboxTreeViewer`, `ListViewer`, `TableTreeViewer`, `TableViewer`, `TreeViewer`

■ למשל, אם ברצוננו להציג למשתמש רשימה של ספרים נרצה לקשור בין רשימת הספרים (עצמים מטיפוס `Book`) ובין רכיב הרשימה היוזואלית

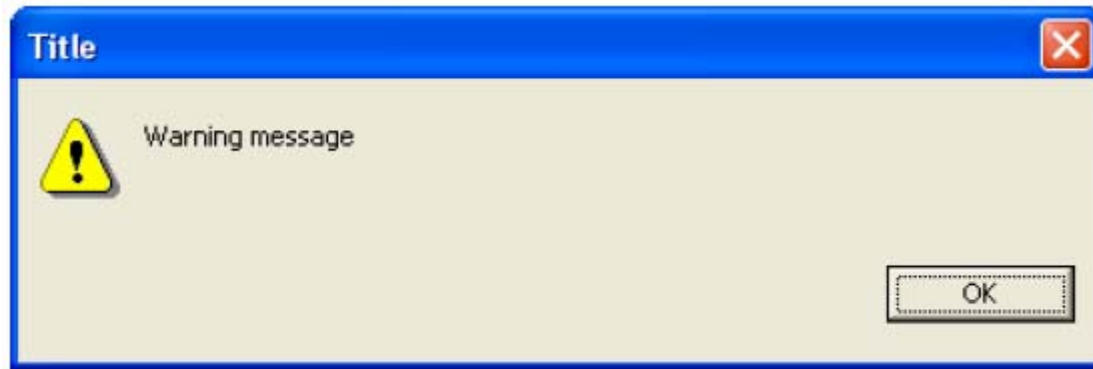
■ לצורך כך יש להגדיר לרשימת הספרים: `LabelProvider` ו- `StructuredContentProvider` (בספרית `Swing` `Renderer`)



# JFace Dialogs

- בחבילה JFace ניתן גם למצוא מגוון תיבות דו-שיח לתקשורת עם המשתמש:

```
MessageDialog.openWarning(shell, "Title", "Warning message");
```





# JFace Dialogs

```
String[] buttonText =  
    new String[] { "Button1", "Button2", "Button3" };  
  
MessageDialog messageBox; =  
    new MessageDialog(shell, "Title", null, "Message",  
        MessageDialog.ERROR, buttonText, 1);  
messageBox.open();
```

ניתן להגדיר מספר סוגי תיבות דו-שיח:

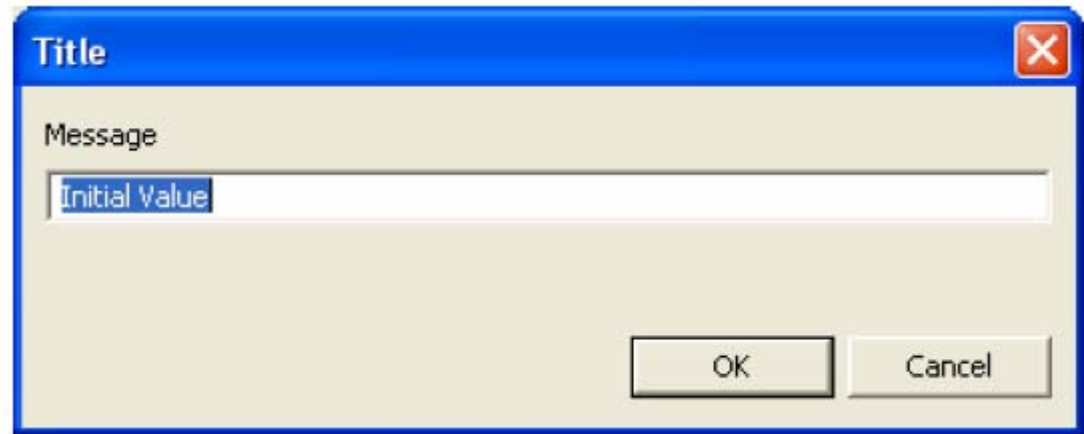
MessageDialog.NONE, MessageDialog.ERROR, MessageDialog.INFORMATION,  
MessageDialog.QUESTION, MessageDialog.WARNING

קריאת בחירת המשתמש ע"י:

```
messageBox.getReturnCode();
```

# JFace Dialogs

```
InputDialog inputBox =  
    new InputDialog(shell, "Title", "Message", "Initial Value", null);  
  
inputBox.open();
```



קריאת קלט משתמש ע"י:

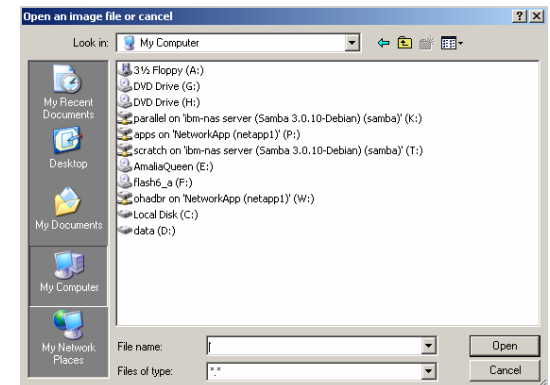
```
inputBox.getReturnCode();  
inputBox.getValue();
```

# JFace Dialogs

```
ColorDialog d = new ColorDialog(shell);  
RGB selection = d.open();
```



```
FileDialog dialog = new FileDialog (shell, SWT.OPEN);  
dialog.setText ("Open an image file or cancel");  
String string = dialog.open ();
```



פיתוח מערכות תוכנה  
אוהד ברזילי

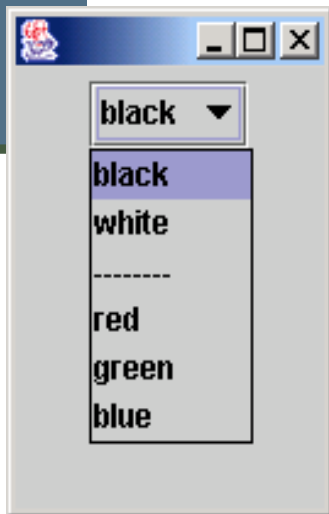
# המודל של ה View

■ לפעמים (בדר"כ) גם לתצוגה יש מודל פנימי משל עצמה

■ כאשר אנו מחליפים טכנולוגיה אנו משכפלים מודל זה שלא לצורך

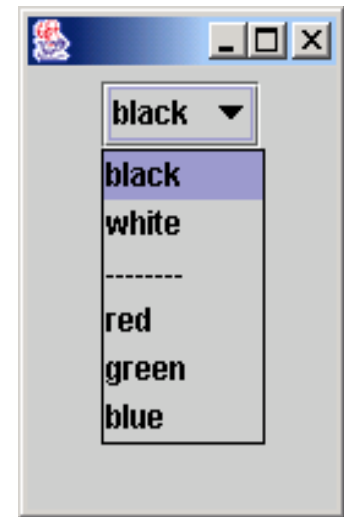
■ תבנית העיצוב Builder נותנת אפשרות להבעה מפורשת של המודל הפנימי

■ לדוגמא: איך נתכנן את תפריטי הבחירה הבאים:



```
abstract class Builder {  
    public abstract void addPart(String choiceStr);  
    public abstract void addSeparator();  
    public abstract Component getResult();  
}
```

```
class ComboBuilder extends Builder {  
  
    private JComboBox combo = new JComboBox();  
  
    public void addPart(String choiceStr) {  
        combo.addItem(choiceStr);  
    }  
  
    public void addSeparator() {  
        combo.addItem("-----");  
    }  
  
    public Component getResult() { // Return the Complex object  
        return combo;  
    }  
}
```



```

class RadioButtonBuilder extends Builder {

    private Box panel = Box.createVerticalBox();
    private ButtonGroup group = new ButtonGroup();

    public void addPart(String choiceStr){
        JRadioButton bt = new JRadioButton(choiceStr);
        group.add(bt);
        panel.add(bt);
    }

    public void addSeparator(){
        panel.add(new JSeparator());
    }

    public Component getResult(){ // Return the Complex object
        // select first radio button
        ((JRadioButton)group.getElements().nextElement())
            .setSelected(true);
        return panel;
    }
}

```



```
class ListDirector {  
  
    public Component create(Builder builder, String[] choiceStrings){  
        for(int i=0; i<choiceStrings.length; i++){  
            String s = choiceStrings[i];  
            if (s==null)  
                builder.addSeparator();  
            else  
                builder.addPart(s);  
        }  
        return builder.getResult();  
    }  
}
```

### Usage:

```
ListDirector director = new ListDirector();
```

```
String[] choiceStrings = {  
    "black", "white", null, "red", "green", "blue"  
};
```

```
Component comp = director.create( new RadioButtonBuilder(), choiceStrings);  
Component comp2 = director.create( new ComboBuilder(), choiceStrings);
```



# MVC - HTML

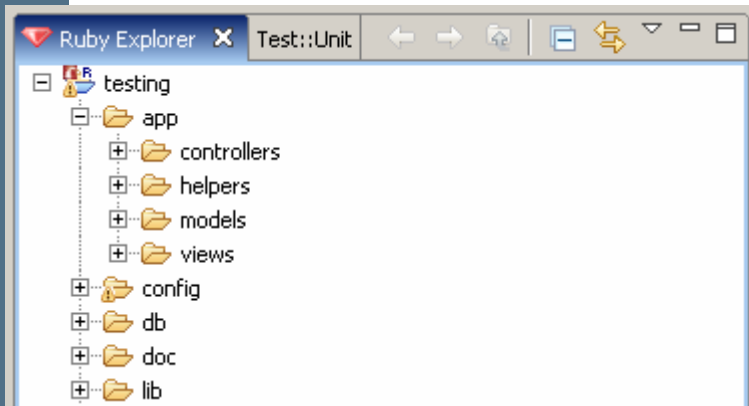
- רעיון ההפרדה בין מודל ותצוגה חלחל גם לתוך תקן ה HTML לעיצוב עמודי אינטרנט
- תגיות HTML הן בחלקן סמנטיות (`<h1>` , `<p>`) ובחלקן עיצוביות (`<font>` , `<b>`)
- יצירת עיצוב אחיד לעמוד או לאתר מצריכה מעבר על כל התגיות

# CSS

- בעזרת תקן ה Cascading Style Sheet (CSS) ניתן להחיל עיצוב חדש על עמודים קיימים ללא שינוי ה HTML
- ניתן להגדיר מספר עיצובים עבור אותו עמוד שיוחלו מן הכלל אל הפרט
  - למשל עיצוב של עמוד מסוים באתר גדול
- <http://www.csszengarden.com/>
- CSS מסוגל לעצב גם עם מסמכי XML

# MVC Frameworks

קיימות שפות תכנות וסביבות פיתוח אשר MVC  
מהווה מרכיב מרכזי בהן:



Ruby on Rails ■

MFC ■

Struts ■

לדוגמא: ■

פרוייקטים ב Ruby on Rails נוצרים אוטומטית עם  
תיקיות נפרדות ל models , Views , ו- controllers