



# Software 1

Recitation No. 11:  
SWT GUI Package

# Events

```
public static void main(String[] args) {  
    Display display = new Display ();  
    Shell shell = new Shell (display);  
    Button ok = new Button (shell, SWT.PUSH);  
    ok.setText ("Push Me!");  
    ok.setLocation(0,0);  
    ok.setSize(100,30);  
    shell.pack ();  
    shell.open ();  
    while (!shell.isDisposed ()) {  
        if (!display.readAndDispatch ()) display.sleep ();  
    }  
    display.dispose ();  
}
```



# Events

הכפתור לא מגיב ללחיצות.  
איך אפשר לטפל בלחיצות:

- הגדרת מחלקה שתירש מכפתור
- מחלקה שתכיל כפתור כאחד משדותיה
- יצירת מחלקה עצמאית שתטפל באירועי הלחיצה ([SelectionEvent](#)):
- על המחלקה המטפלת לממש את הממשק [SelectionListener](#)
- לצורך נוחות: המחלקה [SelectionAdopter](#) מספקת מימוש ברירת מחדל

# Observer Design Pattern

- דרך הטיפול בארועי GUI היא מקרה פרטי של תבנית עיצוב יסודית בתכנות מונחה העצמים
- הבעיה הכללית מאפיינת: Subject אשר מחולל ארועים לוגים (לא בהכרח גרפיים) וישויות אחרות, Observers, אשר מעוניינות לקבל חיווי על כך
- Observers נרשמים כמנויים (subscribers) על הארוע הלוגי אצל ה-Subject
- ה Subject מיידע את כל מנוייו (notify) כל אימת שמתרחש ארוע שיש לו מנויים

# טיפול בארועים במחלקה נפרדת

יתרונות: ■

- הלקוח עובד עם כפתור סטנדרטי ולכן אין צורך לחשוף מבנה פנימי ללקוח
- הלקוח עובד עם כפתור סטנדרטי ולכן אין צורך לבצע האצלה לשרותי המחלקה
- מודולריות – הלוגיקה (טיפול בארועים) מופרדת מהצורניות (מיקום, גודל, סגנון)

# טיפול בארועים במחלקה נפרדת

```
public static void main(String[] args) {
    Display display = new Display ();
    Shell shell = new Shell (display);
    Button ok = new Button(shell, SWT.PUSH);
    ok.addSelectionListener(new ButtonHandler());
    ok.setText ("Push Me!");
    ok.setLocation(0,0);
    ok.setSize(100,30);
    shell.pack ();
    shell.open ();
    while (!shell.isDisposed ()) {
        if (!display.readAndDispatch ())
            display.sleep ();
    }

    display.dispose ();
}
```

# טיפול בארועים במחלקה נפרדת

```
public class ButtonHandler implements SelectionListener {  
    public void widgetSelected(SelectionEvent e) {  
        if (e.getSource() instanceof Button) {  
            Button b = (Button) e.getSource();  
            b.setText("Thanks!");  
        }  
    }  
}
```

# טיפול בארועים במחלקה נפרדת

■ חסרון המימוש הקודם:

■ לעיתים הטיפול באירוע דורש הכרות אינטימית עם המקור שיצר את האירוע (כדי להמנע מחשיפת המבנה הפנימי של המקור)

■ שימוש במחלקה פנימית יוצר את האינטימיות הדרושה

■ בדוגמא הבאה מחלקה המכילה שדה טקסט ותווית

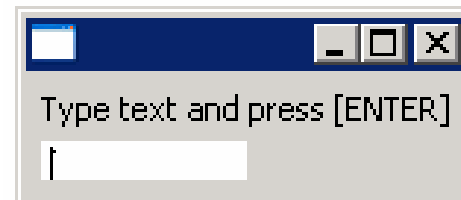
תעדכן את התווית לפי הנכתב בשדה הטקסט ע"י

שימוש במחלקה פנימית



# מחלקה פנימית

```
public class ShellWithLabelAndTextField {  
  
    private Label l;  
    private Text t;  
  
    public static void main(String[] args) {  
        ShellWithLabelAndTextField shell = new ShellWithLabelAndTextField();  
        shell.createShell();  
    }  
  
    public void createShell() {  
        Display display = new Display ();  
        Shell shell = new Shell (display);  
  
        GridLayout gl = new GridLayout();  
        shell.setLayout(gl);  
  
        l = new Label (shell, SWT.CENTER);  
        l.setText ("Type text and press [ENTER]");  
  
        t = new Text(shell, SWT.LEFT);  
        t.addKeyListener(new InnerHnadler());  
  
        // pack(), open(), while ... Dispose()  
    }  
}
```



# מחלקה פנימית

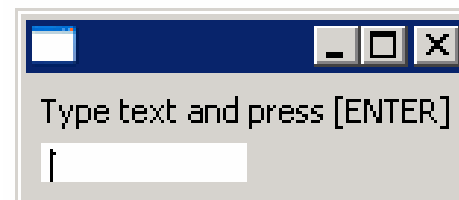
```
public class ShellWithLabelAndTextField {
    private Label l;
    private Text t;

    public static void main(String[] args) { ...}
    public void createShell() {...}

    public class InnerHnadler implements KeyListener {
        public void keyPressed(KeyEvent e) {
            if(e.character == NEW_LINE_CHAR){
                l.setText(t.getText());
                t.setText("");
            }
        }

        public void keyReleased(KeyEvent e) {
            // TODO Auto-generated method stub
        }
    }
}
```

המחלקה הפנימית ניגשת לשדות הפרטיים של המחלקה העוטפת



# מחלקה פנימית אנונימית

```
public class ShellWithLabelAndTextField {  
  
    ...  
    public void createShell() {  
        ...  
        t.addKeyListener(new KeyListener() {  
            public void keyPressed(KeyEvent e) {  
                if (e.character == NEW_LINE_CHAR) {  
                    l.setText(t.getText());  
                    t.setText("");  
                }  
            }  
  
            public void keyReleased(KeyEvent e) {  
                // TODO Auto-generated method stub  
            }  
        });  
  
        // pack(), open(), while ... Dispose()  
    }  
}
```

סוגר סוגריים של  
addKeyListener()

# מחלקות פנימיות - דיון

- הסתרת מידע
- האם המחלקה הפנימית רלוונטית רק בהקשר של המחלקה העוטפת?
- אינה מעודדת שימוש חוזר – מחלקות פנימיות ובפרט מחלקות פנימיות אנונימיות עשויות לשכפל קוד
- קריאות קוד: שימוש במחלקות Adapter