ממשק משתמש גרפי בעזרת SWT

תוכנה 1 בשפת Java

תרגול 11

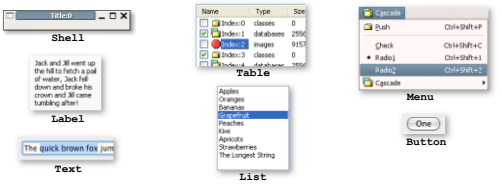
הדס צור ואסף זריצקי

SWT

- בנויה על העיקרון של publish/subscribe
  - אלמנטים בסיסיים (Widgets) מייצרים אירועים (Events) שאליהם נרשמים מאזינים (Listener)
- ה Widgets וה- Events מוגדרים ע"י כותבי הספרייה
- מאזינים נכתבים ע"י המשתמש
  - תגובות שונות לאירועים זהים כתלוי באפליקציה

SWT Widgets

- אבני הבניין של ממשקים גרפים
- מוגדרים ב org.eclipse.swt.widgets
- תת-טיפוסים של המחלקה האבסטרקטית Widget



כפתור


```
public class ShellWithButton {
   public static void main(String[] args) {
      Display display = Display.getDefault();
      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 ();
   }
}
```


הוספת טיפול בארועים

- הכפתור לא מגיב ללחיצות. יש להוסיף טיפול בארוע "לחיצה"
- על המחלקה המטפלת לממש את המנשק `SelectionListener`
- על הכפתור עצמו להגדיר מי העצם (או העצמים) שיטפלו בארוע
- כמה גישות אפשריות:
  - הגדרת מחלקה שיורשת מכפתור
  - מחלקה שמכילה כפתור כאחד משדותיה
  - יצירת מחלקה עצמאית שתטפל באירועי הלחיצה
- לכל אחת מהאפשרויות יתרונות וחסרונות שידונו בהמשך

הוספת טיפול בארועים

- הכפתור לא מגיב ללחיצות. יש להוסיף טיפול באירוע "לחיצה"
- עלינו לממש מאזין המקבל שמטפל באירוע ולהרשם על הווידג'ט המתאים.
- כיצד נדע אילו אירועים מייצר ווידג'ט? איזה מנשק עלינו לממש?
  - נסתכל בתיעוד

Events:
Selection

Method Summary

void	`addSelectionListener(SelectionListener listener)` Adds the listener to the collection of listeners who will be notified when the of the messages defined in the `SelectionListener` interface.

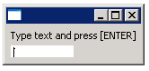
טיפול בארועים במחלקה נפרדת

```
public class ShellWithButton {
   public static void main(String[] args) {
        Display display = Display.getDefault();
        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!");
       }
   }

   public void widgetDefaultSelected(SelectionEvent e){
       // TODO Auto-generated method stub
   }
}
```


מחלקה פנימית



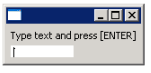

```
public class ShellWithLabelAndTextField {
   private Label l;
   private Text t;

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

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

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

המחלקה הפנימית ניגשת לשדות הפרטיים של המחלקה העוטפת

טיפול בארועים במחלקה נפרדת

- לעיתים הטיפול באירוע דורש הכרות אינטימית עם המקור (כדי להימנע מחשיפת המבנה הפנימי של המקור)
  - שימוש במחלקה פנימית יוצר את האינטימיות הדרושה
- בדוגמא הבאה נרצה לעדכן תווית על סמך קלט מהמשתמש
  - דרושה הכרות לא רק עם יוצר האירוע (Text) אלא גם עם חלקים אחרים במבנה

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 InnerHandler());

        // pack(), open(), while ... Dispose()
   }
```


שימוש במחלקות אנונימיות

- בדרך כלל נזדקק רק למאזין יחיד לכל אירוע
  - נשתמש במחלקה לוקאלית אנונימית
- תזכורת: `new className([argument-list]) {classBody}`
- יצירת מופע חדש של מחלקה ללא שם, שטרם הוגדרה, שיורשת באופן אוטומטי מ-className

```
new interfaceName() {classBody}
```

- יצירת מופע חדש של מחלקה ללא שם, שטרם הוגדרה, שמממשת באופן אוטומטי את interfaceName

מחלקה 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 המתודה addKeyListener()

שימוש ב Adapter

```
public class ShellWithLabelAndTextField {

   ...
   public void createShell() {
      ...
      t.addKeyListener(new KeyAdapter() {
         public void keyPressed(KeyEvent e) {
            if (e.character == NEW_LINE_CHAR) {
               l.setText(t.getText());
               t.setText("");
            }
         }
      });
      // pack(), open(), while ... Dispose()
   }
}
```

סוגר סוגריים של המתודה addKeyListener()

המחלקה SWT

- מוגדרת ב **org.eclipse.swt.SWT**
- אוסף של קבועים:
  - אירועים – Activate, Close, FocusIn, MouseDown, ...
  - צבעים – COLOR_BLACK, COLOR_BLUE, ...
  - תווים – CR, DEL, ESC, ...
  - אירוע מקשים – ARROW_DOWN, END, ...