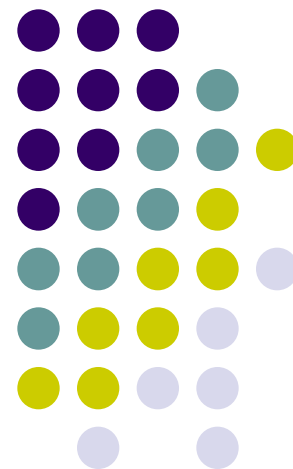


ממשק משתמש גרפי בעזרת SWT

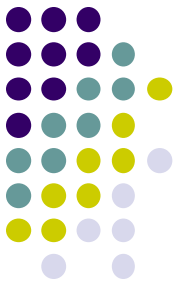
תוכנה 1 בשפת Java



SWT



- בנויה על העיקרון של publish/subscribe
- אלמנטים בסיסיים (Widgets) מייצרים אירועים (Events) שאליהם נרשמים מאזינים (Listener)
- דוגמא 1: משתמש לוחץ על כפתור, שמייצר אירוע לחיצה. מאזין שנרשם לאירוע הלחיצה של הכפתור יכול לשנות את כותרת החלון
- דוגמא 2: משתמש סוגר את החלון, שמייצר אירוע סגירת חלון. מאזין שנרשם לאירוע סגירת החלון פותח חלון ששואל את המשתמש אם הוא רוצה לשמור את השינויים לפני שיצא מהתכנית.
- ה Widgets וה- Events מוגדרים ע"י כותבי הספרייה
- מאזינים נכתבים ע"י כותבי האפליקציה
- מאפשר תגובות שונות לאירועים כתלות באפליקציה



SWT Widgets

- אבני הבניין של ממשקים גרפיים
- מוגדרים ב org.eclipse.swt.widgets
- תת-טיפוסים של המחלקה האבסטרקטית Widget ([קישור לתיעוד](#))
- האתר של SWT מכיל דוגמאות קצרות (snippets) לשימוש בכל Widget <http://www.eclipse.org/swt/widgets/>



Shell

Jack and Jill went up
the hill to fetch a pail
of water, Jack fell
down and broke his
crown and Jill came
tumbling after!

Label

The quick brown fox jum

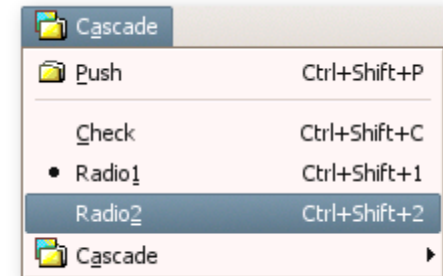
Text

Name	Type	Size
<input type="checkbox"/> Index:0	classes	0
<input checked="" type="checkbox"/> Index:1	databases	2556
<input type="checkbox"/> Index:2	images	9157
<input checked="" type="checkbox"/> Index:3	classes	0
<input type="checkbox"/> Index:4	databases	2556

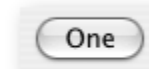
Table

- Apples
- Oranges
- Bananas
- Grapefruit**
- Peaches
- Kiwi
- Apricots
- Strawberries
- The Longest String

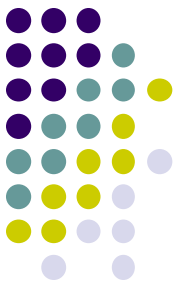
List



Menu



Button



עוד על Widgets

- ביצירת Widget נגדיר

- את ה"הורה" שלו

- את הסגנון שלו

- ההורה הוא Widget היורש מ-Composite, מה שאומר שניתן להוסיף אליו Widgets אחרים

- לדוגמא, כפתור שההורה שלו הוא טאב שההורה שלו הוא חלון – יופיע כפתור בתוך הטאב שבחלון

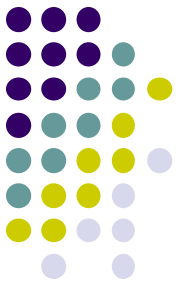
- ה-Widget מתווסף להורה בזמן הקריאה לבנאי

- עבור סגנונות קיימים קבועים במחלקה SWT

- נראה בהמשך



כפתור



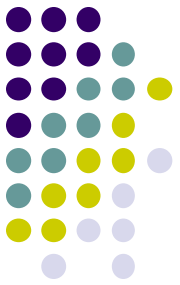
```
public class ShellWithButton1 {
    public static void main(String[] args) {
        Display display = Display.getDefault();
        Shell shell = new Shell(display);
        shell.setLayout(new FillLayout(SWT.VERTICAL));
        shell.setText("example1");

        Button ok = new Button(shell, SWT.PUSH);
        ok.setText("Push Me!");

        shell.pack();
        shell.open();
        while (!shell.isDisposed()) {
            if (!display.readAndDispatch())
                display.sleep();
        }
        display.dispose();
    }
}
```



אז מה היה לנו כאן?



```
Display display = Display.getDefault();
```

• [Display](#) - מקשר בין SWT לתצוגת מערכת ההפעלה (למשל, המסך)

```
Shell shell = new Shell(display);
```

• [Shell](#) - חלון. שימו לב שיצירת חלון לא פותחת אותו עדיין.

```
shell.setLayout(new FillLayout(SWT.VERTICAL));
```

• לכל Composite ניתן להוסיף layout שיגדיר כיצד Widgets מסודרים בתוכו

• [FillLayout](#) - ה- Widgets ממלאים את ה-Composite

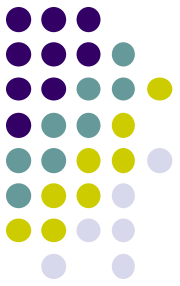
• [SWT.VERTICAL](#) - ה- Widgets מסודרים בצורה אנכית לפי סדר הוספתם ל-Composite

• [RowLayout](#) - דומה ל- [FillLayout](#), אבל ה- Widgets שומרים על גודל קבוע

• [GridLayout](#) - מסדר את ה- Widgets בגריד (לפי עמודות ושורות)



אז מה היה לנו כאן?



```
shell.setText("example1");
```

```
Button ok = new Button(shell, SWT.PUSH);  
ok.setText("Push Me!");
```

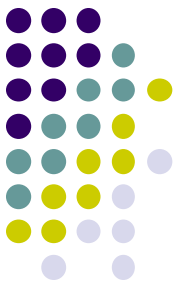
- `setText` – משתנה בהתאם לטיפוס ה-`Widget`. בחלון – קובע את הכותרת, בכפתור לחיצה (`PUSH`) – קובע מה כתוב על הכפתור.

```
shell.pack();
```

- `pack` – גורם לאובייקט גרפי לחשב ולהתאים את גודלו, למשל בהתאם ל-`layout`, ל-`Widgets` שבתוכו וכו'.

```
shell.open();
```

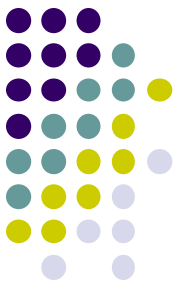
- פתיחת החלון



לולאת האירועים (Event loop)

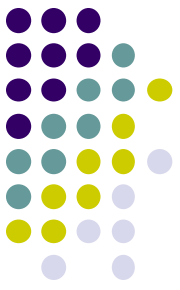
```
while (!shell.isDisposed ()) {  
    if (!display.readAndDispatch())  
        display.sleep();  
}  
display.dispose();
```

- קוד סטנדרטי שמופיע כמעט בכל תכנית SWT
- כל עוד החלון הראשי של התכנית לא נסגר – טפל באירוע הבא בתור ובדוק האם קיים אירוע נוסף
- אם לא קיים – היכנס למצב שינה עד לקבלת אירוע נוסף
- בסוף משחררים את כל המשאבים שהוקצו ע"י קריאה ל- `dispose`
 - כאשר משחררים אלמנט גרפי, כל הצאצאים שלו גם משתחררים
 - כאשר משחררים את `Display`, כל המשאבים הגרפיים משתחררים



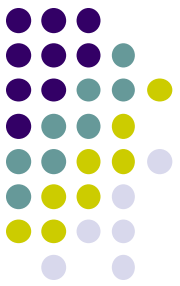
לולאת האירועים (Event loop)

- מערכת ההפעלה מודיעה לתוכנית על אירועים: הקשות על המקלדת, הזזת עכבר והקלקה, בחירת אלמנטים, ועוד.
- ההודעה מתקבלת על ידי עצם יחיד (singleton) מהמחלקה Display, שמייצג את מערכת ההפעלה,
- קבלת אירוע מעירה את התוכנית מהשינה ב- sleep
- כאשר קוראים ל- readAndDispatch ה- display מברר לאיזה רכיב צריך להודיע על האירוע, ומודיע לו
- הרכיב מפעיל את העצמים מהטיפוס המתאים לסוג האירוע שנרשמו להפעלה על ידי קריאה ל- addListener



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 באירוע
- כמה גישות אפשריות:
 - הגדרת מחלקה שיורשת מכפתור
 - מחלקה שמכילה כפתור כאחד משדותיה
 - הוספת מאזין שיטפל באירועי הלחיצה בעת יצירת הכפתור
- המימושים שנראה היום משתייכים לגישה השלישית



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 Widget מסוים? איזה מנשק עלינו לממש?
 - נסתכל בתיעוד
 - התיעוד של Button:

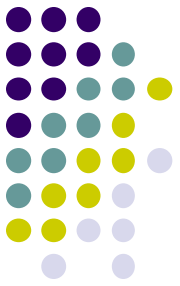
Events: Selection

Method Summary

void	<u>addSelectionListener</u> (<u>SelectionListener</u> 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 ButtonHandler
```

```
    implements SelectionListener {
```

```
        public void widgetSelected(SelectionEvent e) {
```

```
            if (e.getSource() instanceof Button) {
```

```
                Button b = (Button) e.getSource();
```

```
                b.setText("Thanks!");
```

```
            }
```

```
        }
```

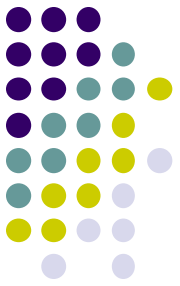
```
        public void widgetDefaultSelected(SelectionEvent e){
```

```
            // TODO Auto-generated method stub
```

```
        }
```

```
    }
```

טיפול בארועים במחלקה נפרדת

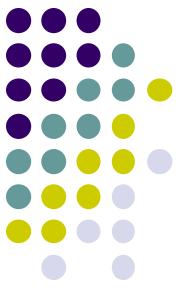


```
Display display = Display.getDefault();
Shell shell = new Shell(display);
shell.setLayout(new FillLayout(SWT.VERTICAL));
shell.setText("example2");
```

```
Button ok = new Button(shell, SWT.PUSH);
ok.setText("Push Me!");
```

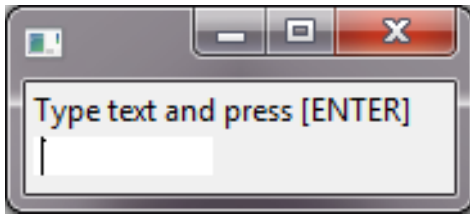
```
ok.addSelectionListener(new ButtonHandler());
```

```
shell.pack();
shell.open();
while (!shell.isDisposed()) {
    if (!display.readAndDispatch())
        display.sleep();
}
display.dispose();
```

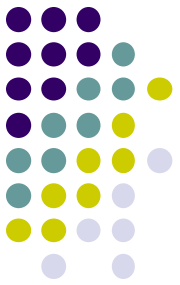


טיפול בארועים במחלקה נפרדת

- 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

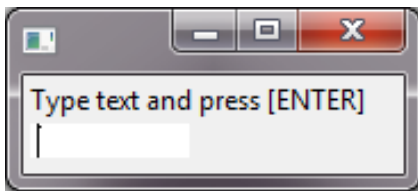


```
public class ShellWithLabelAndTextField1 {  
    private Label label;  
    private Text text;  
  
    public static void main(String[] args) {...}  
  
    public void createShell() {...}
```

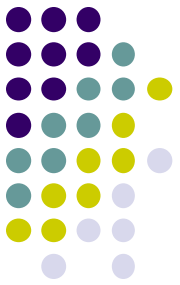
המחלקה הפנימית ניגשת לשדות הפרטיים של המחלקה העוטפת

```
public class InnerHandler implements KeyListener {  
    public void keyPressed(KeyEvent e) {  
        if (e.character == SWT.CR) {  
            label.setText(text.getText());  
            text.setText("");  
        }  
    }  
  
    public void keyReleased(KeyEvent e) {  
        // TODO Auto-generated method stub  
    }  
}
```

```
}
```



מחלקה פנימית



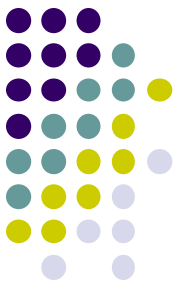
```
public class ShellWithLabelAndTextField1 {
    private Label label;
    private Text text;

    public static void main(String[] args) {
        ShellWithLabelAndTextField1 shell = new ShellWithLabelAndTextField1();
        shell.createShell();
    }

    public void createShell() {
        Display display = new Display();
        Shell shell = new Shell(display);
        shell.setLayout(new RowLayout(SWT.VERTICAL));

        label = new Label(shell, SWT.CENTER);
        label.setText("Type text and press [ENTER]");

        text = new Text(shell, SWT.LEFT);
        text.addKeyListener(new InnerHandler());
        // pack(), open(), while ... Dispose()
    }
}
```

שימוש במחלקות אנונימיות

- בדרך כלל נזדקק רק למאזין יחיד לכל אירוע
- נשתמש במחלקה לוקאלית אנונימית

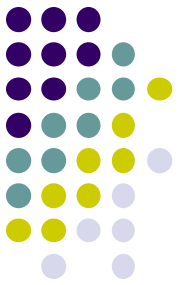
```
new className([argument-list]) {classBody}
```

● **תזכורת:**

- יצירת מופע חדש של מחלקה ללא שם, שטרם הוגדרה, שיורשת באופן אוטומטי מ `className`

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

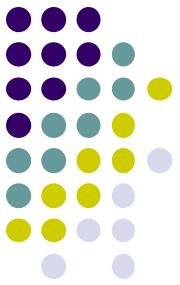
- יצירת מופע חדש של מחלקה ללא שם, שטרם הוגדרה, שממשת באופן אוטומטי את `interfaceName`



מחלקה אנונימית

```
public class ShellWithLabelAndTextField2 {  
    private Label label;  
    private Text text;  
  
    public static void main(String[] args) {...}  
  
    public void createShell() {  
        ...  
        text.addListener(new KeyListener() {  
            public void keyPressed(KeyEvent e) {  
                if (e.character == SWT.CR) {  
                    label.setText(text.getText());  
                    text.setText("");  
                }  
            }  
  
            public void keyReleased(KeyEvent e) {  
                // TODO Auto-generated method stub  
            }  
        });  
  
        // pack(), open(), while ... Dispose()  
    }  
}
```

← addKeyListener() סוגר סוגריים של המתודה



שימוש ב Adapter

```
public class ShellWithLabelAndTextField2 {
    private Label label;
    private Text text;

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

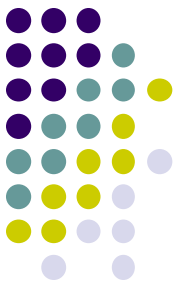
    public void createShell() {
        ...
        text.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                if (e.character == SWT.CR) {
                    label.setText(text.getText());
                    text.setText("");
                }
            }
        });

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

```
public class KeyAdapter implements
    KeyListener {

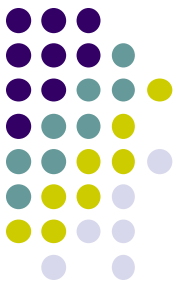
    @Override
    public void keyPressed(KeyEvent arg0) {
        //do nothing
    }

    @Override
    public void keyReleased(KeyEvent arg0) {
        // do nothing
    }
}
```



המחלקה SWT

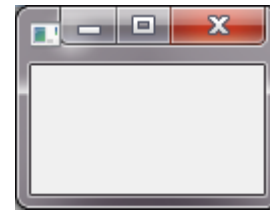
- מוגדרת ב `org.eclipse.swt.SWT`
- אוסף של קבועים:
 - אירועים – `MouseDown`, `FocusIn`, `Close`, `Activate` – ...
 - צבעים – `COLOR_BLUE`, `COLOR_BLACK` – ...
 - תווים – `ESC`, `DEL`, `CR` – ...
 - אירוע מקשים – `END`, `ARROW_DOWN` – ...
 - עיצובים
- ניתן להוסיף מס' קבועים ע"י שימוש באופרטור | (bitwise OR)
 - `SWT.V_SCROLL | SWT.H_SCROLL | SWT.BORDER`

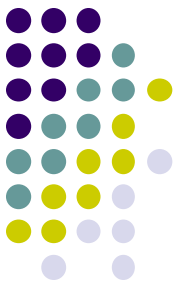


דוגמא נוספת – רשימת משימות

```
public class TaskList1 {  
    private static final Display display =  
        Display.getDefault();  
    private static Shell shell;  
  
    public static void main(String[] args) {  
        TaskList1 taskList = new TaskList1();  
        taskList.createShell();  
        taskList.runApplication();  
    }  
  
    private void createShell() {  
        shell = new Shell(display);  
        shell.setText("My Tasks");  
    }  
  
    private void runApplication() {  
        shell.pack();  
        shell.open();  
        while (!shell.isDisposed()) {  
            if (!display.readAndDispatch())  
                display.sleep();  
        }  
        display.dispose();  
    }  
}
```

- נכתוב אפליקציית SWT פשוטה המאפשרת להזין משימות בשדה טקסט, ללחוץ על כפתור ולהוסיף את הטקסט שהוזן לרשימה.
- נתחיל מבניית שלד האפליקציה





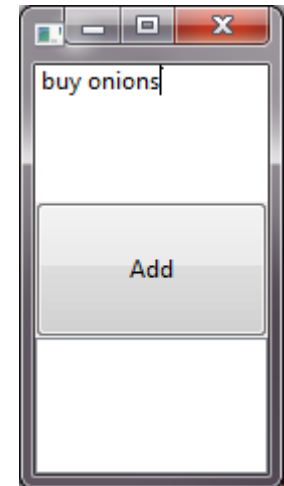
הוספת Widgets

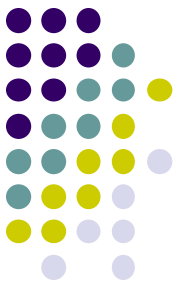
- נוסף Layout בסיסי ל-Shell, וכן שדה טקסט, כפתור ורשימה
- כרגע אין מאזינים לאף פעולה

```
private void createShell() {  
    shell = new Shell(display);  
    shell.setText("My Tasks");  
    shell.setLayout(new FillLayout(SWT.VERTICAL));
```

```
//a text field to enter a task  
Text input = new Text(shell, SWT.LEFT);  
  
//a button to add a task to the list  
Button add = new Button(shell, SWT.PUSH);  
add.setText("Add");  
  
//the list  
List list = new List(shell, SWT.BORDER);
```

```
}
```





הוספת פונקציונליות

```
//a text field to enter a task
final Text input = new Text(shell, SWT.LEFT);

//a button to add a task to the list
Button add = new Button(shell, SWT.PUSH);
add.setText("Add");

// the list
final List list = new List(shell, SWT.BORDER);

// the action to perform when pressing the button
add.addSelectionListener(new SelectionAdapter() {
    @Override
    public void widgetSelected(SelectionEvent e) {
        String text = input.getText();
        //add the task to the list, if the input text is not empty
        if (text != null && text.length() > 0) {
            list.add(text);
            input.setText("");
        }
    }
});
```

- כעת, נוסיף את הפעולה של לחיצה על כפתור – תוך שימוש במחלקה `final` אנונימית.
- הכפתור צריך להכיר את שדה הטקסט ואת הרשימה, לכן נשנה אותם ל-`final`
- עכשיו זה עובד!



נטפל בעיצוב החלון

- תחילה, נתכנן איך היינו רוצים שהחלון ייראה

Header	
Input field	Add
Main content area	

- וכיצד הוא ישתנה בשינוי גודל החלון

Header	
Input field	Add
Main content area	

Header	
Input field	Add
Main content area	

Header	
Input field	Add
Main content area	



עיצוב החלון

שני טורים ברוחב
לא אחיד

```
shell = new Shell(display);  
shell.setText("My Tasks");  
shell.setLayout(new GridLayout(2, false));
```

```
// a text field to enter a task  
final Text input = new Text(shell, SWT.LEFT);  
input.setLayoutData(new GridData(  
    GridData.FILL_HORIZONTAL));
```

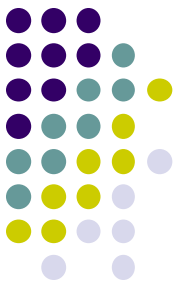
```
// a button to add a task to the list  
Button add = new Button(shell, SWT.FLAT);  
add.setText("Add");
```

```
// the list  
final List list = new List(shell, SWT.BORDER);  
GridData listGridData = new GridData(GridData.FILL_BOTH);  
listGridData.horizontalSpan = 2;  
listGridData.widthHint = 300;  
listGridData.heightHint = 300;  
list.setLayoutData(listGridData);
```

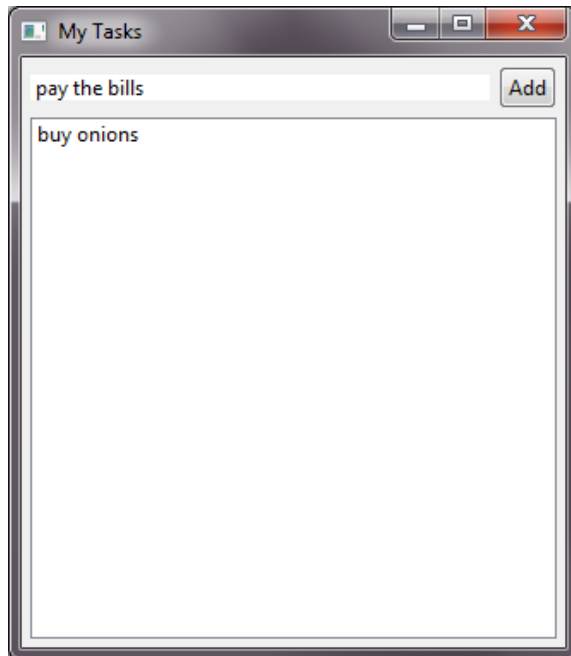
שדה הטקסט נמתח
לרוחב

הרשימה נמתחת
לרוחב ולאורך,
תופסת שני טורים,
וגודלה המומלץ
300x300
פיקסלים

- נשתמש ב-[GridLayout](#)
- מוסיף אלמנטים לגריד משמאל לימין ומלמעלה למטה לפי הסדר
- מאפשר הוספת `GridData` שמציין איך כל `Widget` יתנהג בגריד



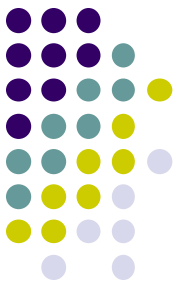
עיצוב החלון



● עכשיו האפליקציה שלנו נראית כך

● עוד תוספות?

- סימון משימה כ"הושלמה"
- הוספת תאריך יצירת המשימה
- הוספת תאריך יעד לביצוע המשימה
- עריכת משימה
- מחיקת משימה
- ...



תוספת - מחיקת משימות מהרשימה

- נוסףKeyListener לרשימה שיאזין ללחיצה על delete
- מ-Widget הרשימה ניתן לקבל את האינדקס של הפריט הנבחר ולמחוק את הפריט

```
// the action to perform when DELETE is pressed on the list
list.addKeyListener(new KeyAdapter() {
    @Override
    public void keyPressed(KeyEvent e) {
        if (e.character == SWT.DEL) {
            int selectionIndex = list.getSelectionIndex();
            //if a list item is selected, delete it from the list
            if (selectionIndex >= 0)
                list.remove(selectionIndex);
        }
    }
});
```