

מה עושים היום?

- hashCode ו- equals
- ממשק משתמש גרפי

2

ממשק משתמש גרפי בעזרת SWT

תוכנה 1 בשפת Java
ניר אטיאס ואלכסיי זגלסקי

1

מה יודפס?

```
public class Name {
    private String first, last;
    ...

    public static void main(String[] args) {
        Name name1 = new Name("Mickey", "Mouse");
        Name name2 = new Name("Mickey", "Mouse");
        System.out.println(name1.equals(name2)); 

        List<Name> names = new ArrayList<Name>();
        names.add(name1);
        System.out.println(names.contains(name2)); 
    }
}
```

4

תזכורת: המחלקה Object

```
package java.lang;

public class Object {
    public final native Class<?> getClass();

    public native int hashCode();

    public boolean equals(Object obj) {
        return (this == obj);
    }

    protected native Object clone() throws CloneNotSupportedException;

    public String toString() {
        return getClass().getName() + "@" +
            Integer.toHexString(hashCode());
    }
    ...
}
```

3

החזרה של equals

- רפלקסיבי
 - `true` `x.equals(x)` יחזיר
- סימטרי
 - `x.equals(y)` יחזיר `true` אם `y.equals(x)` יחזיר `true`
- טרנזיטיבי
 - אם `x.equals(y)` מחזיר `true` וגם `y.equals(z)` מחזיר `true` אז `x.equals(z)`
- עקבי
 - סדרת קריאות ל `x.equals(y)` תחזיר `true` (או `false`) באופן עקבי אם מידע שדרוש לצורך ההשוואה לא השתנה
- השוואה ל `null`
 - `x.equals(null)` תמיד תחזיר `false`

6

הבעיה

- רצינו השוואה לפי תוכן אבל לא דרסנו את `equals`
- מימוש ברירת המחדל הוא השוואה של מצביעים

```
public class Object {
    ...
    public boolean equals(Object obj) {
        return (this == obj);
    }
    ...
}
```

5

טעות נפוצה

- להגדיר את הפונקציה equals כך:

```
public boolean equals(Name name) {
    return first.equals(other.first) &&
        last.equals(other.last);
}
```

- זו אינה דריסה (overriding) אלא העמסה (overloading)
- שימוש ב @Override יפתור את הבעיה

8

מתכון ל equals

```
public boolean equals(Object obj) {
    if (this == obj) // 1. ודאו כי הארגומנט אינו מצביע לאובייקט הנכחי
        return true;
    if (obj == null) // 2. ודאו כי הארגומנט אינו null
        return false;
    if (getClass() != obj.getClass()) // 3. ודאו כי הארגומנט הוא הטיפוס המתאים להשוואה
        return false;
    Name other = (Name) obj; // 4. המירו את הארגומנט לטיפוס הנכון
    return first.equals(other.first) &&
        last.equals(other.last); // 5. לכל שדה "משמעותי", בדיקו ששדה זה בארגומנט תואם לשדה באובייקט הנכחי
}
```

7

כמעט

```
public class Name {
    ...
    @Override public equals(Object obj) {
        ...
    }

    public static void main(String[] args) {
        Name name1 = new Name("Mickey", "Mouse");
        Name name2 = new Name("Mickey", "Mouse");
        System.out.println(name1.equals(name2)); // יודפס true

        Set<Name> names = new HashSet<Name>();
        names.add(name1);
        System.out.println(names.contains(name2)); // יודפס false
    }
}
```

10

אז הכל בסדר?

```
public class Name {
    ...
    @Override public equals(Object obj) {
        ...
    }

    public static void main(String[] args) {
        Name name1 = new Name("Mickey", "Mouse");
        Name name2 = new Name("Mickey", "Mouse");
        System.out.println(name1.equals(name2)); // יודפס true

        List<Name> names = new ArrayList<Name>();
        names.add(name1);
        System.out.println(names.contains(name2)); // יודפס true
    }
}
```

9

החזרה של hashCode

- עקביות**
 - מחזירה אותו ערך עבור כל הקריאות באותה ריצה, אלא אם השתנה מידע שבשימוש בהשוואת equals של המחלקה
- שוויון**
 - אם שני אובייקטים שווים לפי הגדרת equals אזי hashCode תחזיר ערך זהה עבורם
- חוסר שוויון**
 - אם שני אובייקטים אינם שווים לפי equals לא מובטח ש hashCode תחזיר ערכים שונים
 - החזרת ערכים שונים יכולה לשפר ביצועים של מבני נתונים המבוססים על hashing (לדוגמה, HashMap ו HashSet)

12

hashCode ו equals

חובה לדרוס את hashCode בכל מחלקה שדורסת את equals!

11

תמיכה באקליפס

- אקליפס תומך ביצירה אוטומטית (ומשולבת) של hashCode ו equals
- בתפריט Source ניתן למוצא Generate hashCode() and equals()

14

מימוש hashCode

```
@Override public int hashCode() {  
    return 31 * first.hashCode() + last.hashCode();  
}
```

- השתדלו לייצר hash כך שלאובייקטים שונים יהיה ערך hash שונה
- המימוש החוקי הגרוע ביותר (לעולם לא לממש כך!)

```
@Override public int hashCode() {  
    return 42;  
}
```

13

SWT Widgets

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



16

SWT

- בניה 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 כולוי באפליקציה

15

הוספת טיפול בארועים

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

18

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 ();  
    }  
}
```

17

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
    }
}
```

20

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 בתיעד

Events:
Selection

Method Summary

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

19

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

22

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 ();
    }
}
```

21

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()
    }
}
```

24

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 נוגשת לשידות הפרטיים של המחלקה העוטפת

23

מחלקה אנונימית

```
public class ShellWithLabelAndTextField {
    ...
    public void createShell() {
        ...
        t.addListener(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()

26

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

25

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` – ...

28

שימוש ב Adapter

```
public class ShellWithLabelAndTextField {
    ...
    public void createShell() {
        ...
        t.addListener(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()

27