

## מחשבים בחו"ל



יצאת לחו"ל (ברצלונה) עם המחשב הנייד



- זכרת להביא כבל חשמל
- תואם בהספק (220v)
- תואם בתדר (50hz)



- אבל כל זה לא מספיק,
- אם לא הבאת מתאם

## מתאמים ומחלקות פנימיות

תרגול 10  
ליאור שפירא, מתי שמרת

## דוגמא

```
public interface RandomNumberGenerator {
    public int random();
}
```

אפשר היה לממש את השירות בעצמנו, למשל כך:

```
public class FastRandom implements RandomNumberGenerator {
    public int last;

    public FastRandom() {
        last = (int) System.nanoTime();
    }

    public int random() {
        last = last * 1103515245 + 12345;
        return (short) (last & 0xFFFF);
    }
}
```

## בעיית המתאם



- הבעיה היא טכנית
- הלקוח (laptop) זקוק ל 220v ב- 50hz
- הספק (רשת החשמל) מספק 220v ב- 50hz
- הבעיה היא בממשק בין השניים
- בעיה דומה עשויה לקרות גם בתוכנה
- במערכת חדשה או זקוקים ליכולות מסוימות (למחלקה שממשת משהו)
- קיים מודול (מחלקה) שנכתב למערכת קודמת המספק את היכולות חשובות
- אולם בשלב התכנון של המערכת החדשה הוגדר מנשק למחלקה זו השונה מהמנשק שאותו ממש המודול הקיים

## המתאם

- הבעיה היא טכנית והיא נעוצה בממשק:
- המחלקה המבוקשת נדרשת לממש את random()
- נולד java.util.Random, אף על פי שהוא ממש אקראיות, הוא ששה זאת בעזרת פונקציות (nextXXX())

הפתרון: מחלקת מתאם (Adapter)

```
public class RandomAdapter implements RandomNumberGenerator {
    java.util.Random r;

    public RandomAdapter() {
        r = new java.util.Random();
    }

    public int random() {
        return r.nextInt();
    }
}
```

המחלקה מתאמת (מתרגמת) בין המנשק של המחלקה Random למנשק RandomNumberGenerator

## שימוש חוזר

- אולם הרבה יותר פשוט להשתמש במחלקה קיימת:
- java.util.Random
- בעיה:
- המחלקה אינה מממשת את המנשק הדרוש במערכת: RandomNumberGenerator
- ובפרט היא אינה מממשת את השירות הדרוש random()

Method Summary
Random() Constructs the next pseudorandom number
nextInt() Returns the next pseudorandom, uniformly distributed <code>int</code> value from the random number generator's sequence
nextInt(int) Returns the next pseudorandom, uniformly distributed <code>int</code> value between <code>0</code> and <code>value</code> from the random number generator's sequence
nextDouble() Returns the next pseudorandom, uniformly distributed <code>double</code> value between <code>0.0</code> and <code>1.0</code> from the random number generator's sequence
nextGaussian() Returns the next pseudorandom, Gaussian (normal) distributed <code>double</code> value with mean <code>0.0</code> and standard deviation <code>1.0</code> from the random number generator's sequence
nextLong() Returns the next pseudorandom, uniformly distributed <code>long</code> value from the random number generator's sequence
nextLong(long) Returns the next pseudorandom, uniformly distributed <code>long</code> value between <code>0</code> (inclusive) and the specified value (exclusive), drawn from the random number generator's sequence
nextFloat() Returns the next pseudorandom, uniformly distributed <code>float</code> value from the random number generator's sequence
nextBoolean() Returns the result of the random number generator using a single coin toss

```

import java.util.Set;

public interface RandomSetGenerator {
    public Set<Integer> random();
}

import java.util.Set;
import java.util.TreeSet;

public class RandomSetAdapter implements RandomSetGenerator {

    java.util.Random r;

    public RandomSetAdapter() {
        r = new java.util.Random();
    }

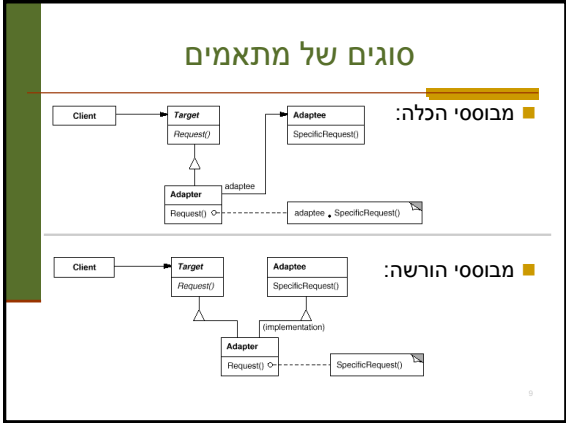
    public Set<Integer> random() {
        Set<Integer> result = new TreeSet<Integer>();
        result.add(r.nextInt());
        result.add(r.nextInt());
        return result;
    }
}

```

## לא רק תאומים

- ומה אם נרצה לצאת עם המחשב הנייד לארה"ב?
- כעת הבעיה היא לא רק בממשק אלא גם בהפרשי המתחים
- ניתן לפתור בעיה זו באותה שיטה:
  - המתאם (adapter) יבצע גם את המרת הזרם (transformator)
  - ע"י שימוש בספק המקורי
- בדוגמא שלנו: איך נממש מנשק חדש שבו נדרשים להחזיר זוג (Set) מספרים אקראיים?

## מחלקות פנימיות (מקוננות) Inner (Nested) Classes



## מחלקות פנימיות

- הגדרת מחלקה כפנימית מרמזת על היחס בין המחלקה הפנימית והמחלקה העוטפת:
  - למחלקה הפנימית יש משמעות רק בהקשר של המחלקה החיצונית
  - למחלקה הפנימית יש הכרות אינטימית עם המחלקה החיצונית
  - המחלקה הפנימית היא מחלקת עזר של המחלקה החיצונית
- דוגמאות:
  - Collection -> Iterator
  - Body -> Brain
  - מבני נתונים המוגדרים ברקורסיה: List -> Cell

## Inner Classes

- מחלקה פנימית היא מחלקה שהוגדרה בתחום (Scope – בין המוסולסיים) של מחלקה אחרת
- דוגמא:
 

```

public class House {
    private String address;
    public class Room {
        private double width;
        private double height;
    }
}

```

**שימוש לבי!**  
Room אינה שדה של המחלקה House

## Inner Classes

```
public class House {
    private String address;
    public class Room {
        // hidden reference to a House
        private double width;
        private double height;
        public String toString(){
            return "Room inside: " + address;
        }
    }
}
```

14

## Inner Classes

- ב Java כל מופע של עצם מטיפוס המחלקה הפנימית צריך להיות משויך לעצם מטיפוס המחלקה העוטפת

### השלכות

- תחביר מיוחד לבנאי
- לעצם מטיפוס המחלקה הפנימית יש שדה הפנייה שמוצא אוטומטית לעצם מהמחלקה העוטפת
- כתוצאה לכך יש למחלה הפנימית גישה לשדות ולשדותים (אפילו פרטיים) של המחלקה העוטפת ולהיפך

13

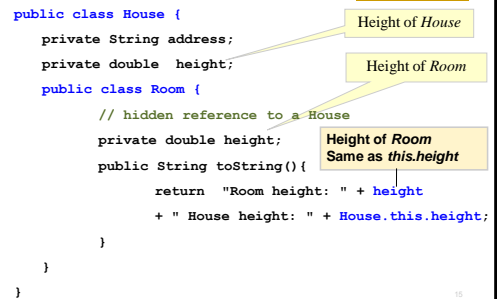
## יצירת מופעים

- כאשר המחלקה החיצונית יוצרת מופע של עצם מטיפוס המחלקה הפנימית אזי העצם יוצר בהקשר של העצם היוצר
- כאשר עצם מטיפוס המחלקה הפנימית נוצר מחוץ למחלקה העוטפת, יש צורך בתחביר מיוחד

16

## Inner Classes

```
public class House {
    private String address;
    private double height;
    public class Room {
        // hidden reference to a House
        private double height;
        public String toString(){
            return "Room height: " + height
                + " House height: " + House.this.height;
        }
    }
}
```



15

## יצירת מופע שלא ע"י המחלקה החיצונית

```
public class Test {
    public static void main(String[] args){
        House h = new House();
        House.Room r = h.new Room();
    }
}
```

`outerObject.new InnerClassName`

18

## יצירת מופע ע"י המחלקה החיצונית

```
public class House {
    private String address;
    public void test() {
        Room r = new Room();
        System.out.println( r );
    }

    public class Room {
        ...
    }
}
```

17

```

public class House {
    private String address;
    public static class Room {
        public String toString(){
            return "Room " + address;
        }
    }
}

public class Test {
    public static void main(String[] args){
        House.Room r = new House.Room();
        ...
    }
}

```

Error: this room is not related to any house

Not related to any house

`new OuterClassName.InnerClassName`

## Static Nested Classes

- ניתן להגדיר מחלקה פנימית כ `static` ובכך ליצין שהיא אינה קשורה למופיע מסוים של המחלקה העוטפת
- הדבר אנלוגי למחלקה שכל שרתיה הוגדרו כ `static` והיא משמשת כספרייה עבור מחלקה מסוימת
- בשפת C++ יחס זה מושג ע"י הגדרת `friend` יחס

## מחלקות מקומיות - מחלוקת פנימיות בתוך מתודות

- ניתן להגדיר מחלקה פנימית בתוך מתודה של המחלקה החיצונית
- הדבר מגביל את תחום ההכרה של אותה מחלקה לתחום אותה המתודה בלבד
- המחלקה הפנימית תוכל להשתמש במשתנים מקומיים של המתודה רק אם הם הוגדרו כ `final` (מדוע?)

## הגנה על מחלקות פנימיות סטטיות

- אם המחלקה הפנימית אינה ציבורית (אינה מוגדרת `public`), הטיפוס שלה מוסתר, אבל עצמים מהמחלקה אינם מוסתרים אם יש התייחסות אליהם

```

public class Outer ... {
    private static class Inner implement Inter {...}
    public static Inter getInner() {
        return new Inner ();
    }
}

Inter i = new Outer.Inner(); //error
Inter i = Outer.getInner(); // ok

```

## שימוש במשתנים מקומיים

```

public class Test {
    public void test (int x) {
        final int y = x+3;
        class Info {
            public String toString(){
                return "***" + y + "****";
            }
        };
        System.out.println( new Info());
    }
}

```

## מחלקות מקומיות

```

public class Test {
    ...
    public void test () {
        class Info {
            private int x;
            public Info(int x) {this.x=x;}
            public String toString() {
                return "***" + x + "****";
            }
        };
        Info inf1 = new Info(0);
        System.out.println(inf1);
    }
}

```

## הידור של מחלקות פנימיות

- המהדר (קומפיילר) יוצר קובץ `.class`. עבור כל מחלקה. מחלקה פנימית אינה שונה במובן זה ממחלקה רגילה
- שם המחלקה הפנימית יהיה `Outer$Inner.class`
- אם המחלקה הפנימית אנונימית, שם המחלקה שיוצר הקומפיילר יהיה `Outer$1.class`

26

## מחלקות אנונימיות

- בעזרת מחלקות פנימיות ניתן להגדיר מחלקות אנונימיות – מחלקות ללא שם
- מחלקות אנונימיות שימושיות מאוד במערכות מונחות ארועים (כגון GUI) וילמדו בהמשך הקורס

25