

Software 1

תרגול 10
Nested Classes
Object Serialization (IO)

1

מחלקות מקוננות

NESTED CLASSES

2

מחלקה מקוננת

■ מחלקה שמוגדרת בתוך מחלקה אחרת

- 1. סטטית (static member)
- 2. לא סטטית (nonstatic member)
- 3. אנונימית (anonymous)
- 4. מקומית (local)

מחלקות פנימיות
(inner)

```
class Outer {
    static class NestedButNotInner {
        ...
    }
    class Inner {
        ...
    }
}
```

3

בשביל מה זה טוב

■ קיבוץ לוגי

אם עושים שימוש בטיפוס רק בהקשר של טיפוס מסוים נטמיע את הטיפוס כדי לשמר את הקשר הלוגי

■ הכמסה מוגברת

על ידי הטמעת טיפוס אחד באחר אנו חושפים את המידע הפרטי רק לטיפוס המוטמע ולא לכולם

■ קריאות

מיקום הגדרת טיפוס בסמוך למקום השימוש בו

4

תכונות משותפות

- למחלקה מקוננת יש גישה לשדות הפרטיים של המחלקה העוטפת ולהפך
- הנראות של המחלקה היא עבור "צד שלישי"
- אלו מחלקות (כמעט) רגילות לכל דבר ועניין
- יכולות להיות אבסטרקטיות, לממש מנשקים, לרשת ממחלקות אחרות וכדומה

5

Static Member Class

■ מחלקה רגילה ש"במקרה" מוגדרת בתוך מחלקה אחרת

■ החוקים החלים על איברים סטטיים אחרים חלים גם על מחלקות סטטיות

■ גישה לשדות / פונקציות סטטיים בלבד

■ גישה לאיברים לא סטטיים רק בעזרת הפניה לאובייקט

■ גישה לטיפוס בעזרת שם המחלקה העוטפת

```
OuterClass.StaticNestedClass
```

■ יצירת אובייקט

```
OuterClass.StaticNestedClass nested =
    new
    OuterClass.StaticNestedClass();
```

6

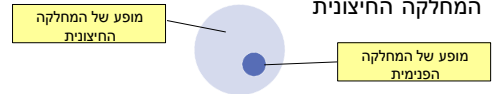
AbstractMap

```
public abstract class AbstractMap<K,V> implements Map<K,V> {  
  
    public static class SimpleEntry<K,V>  
    implements Entry<K,V>, java.io.Serializable {  
        private final K key;  
        private V value;  
  
        ...  
  
        ...  
    }  
}
```

7

Non-static Member Class

- כל מופע של המחלקה הפנימית משייך למופע של המחלקה החיצונית



- השייך מבוצע בזמן יצירת האובייקט ואינו ניתן לשינוי
- באובייקט הפנימי קיימת הפניה לאובייקט החיצוני (qualified this)

8

House Example

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

גישה למשתנה פרטי לא סטטי

9

Inner Classes

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

Height of House

Height of Room

Height of Room Same as this.height

10

AbstractList

```
public abstract class AbstractList<E> extends  
    AbstractCollection<E> implements List<E> {  
    public Iterator<E> iterator() {  
        return new Itr();  
    }  
  
    private class Itr implements Iterator<E> {  
        ...  
    }  
  
    private class ListItr extends Itr implements  
    ListIterator<E> {  
        ...  
    }  
}
```

11

יצירת מופעים

- כאשר המחלקה העוטפת יוצרת מופע של עצם מטיפוס המחלקה הפנימית אזי העצם נוצר בהקשר של העצם היוצר

- כאשר עצם מטיפוס המחלקה הפנימית נוצר מחוץ למחלקה העוטפת, יש צורך בתחביר מיוחד

```
outerObject.new InnerClassConstructor(...)
```

12

מחלקות אנונימיות

- מחלקה ללא שם
- הגדרה ויצירת מופע בנקודת השימוש
- מגבלות:
 - חייבת לרשת מטיפוס קיים (מנשק או מחלקה)
 - לא ניתן להגדיר איברים סטטיים, לא ניתן להשתמש בהקשר שדורש שם (instanceof), לא ניתן לרשת ממספר טיפוסים, לקוחות מוגבלים לממשק של טיפוס האב
- מחלקה אנונימית צריכה להיות קצרה כדי לא לפגוע בקריאות של הקוד

13

דוגמאות שימוש

Function object (functor) ■

- מיון מחרוזות לפי אורך

```
Arrays.sort(stringArray, new Comparator<String>() {  
    public int compare(String s1, String s2) {  
        return s1.length() - s2.length();  
    }  
})
```

- מימוש איטרטור

```
public Iterator<E> iterator() {  
    return new Iterator<E>() {  
        boolean hasNext() {...}  
        E next() {...}  
        void remove() {...}  
    }  
}
```

14

המרה ממערך לרשימה

```
static List<Integer> intArrayAsList(final int[] a) {  
    if (a == null)  
        throw new IllegalArgumentException();  
    return new ArrayList<Integer>() {  
        public Integer get(int i) {  
            return a[i];  
        }  
        public Integer set(int i, Integer val) {  
            int oldVal = a[i];  
            a[i] = val;  
            return oldVal;  
        }  
        public int size() {  
            return a.length;  
        }  
    };  
}
```

גישה למשתנים מקומיים שהוגדרו final

15

מחלקות מקומיות

- מוגדרות בתוך מתודות

- יש להם שם וניתן להשתמש בהם מספר פעמים, בתוך אותה מתודה
- אובייקט עוטף רק אם הוגדרו בהקשר לא סטטי; לא ניתן להגדיר משתנים סטטיים
- המחלקה הפנימית תוכל להשתמש גם במשתנים מקומיים של המתודה אבל רק אם הם הוגדרו כ-**final**

16

דוגמה - המרה ממערך לרשימה

```
static List<Integer> intArrayAsList(final int[] a) {  
    if (a == null)  
        throw new IllegalArgumentException();  
    class IntegerList extends ArrayList<Integer> {  
        public Integer get(int i) {  
            return a[i];  
        }  
        public Integer set(int i, Integer val) {  
            int oldVal = a[i];  
            a[i] = val;  
            return oldVal;  
        }  
        public int size() {  
            return a.length;  
        }  
    }  
    return new IntegerList();  
}
```

17

IO - OBJECT SERIALIZATION

18

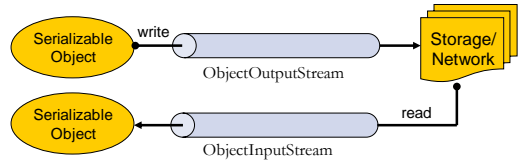
Object Serialization

- A mechanism that enables objects to be:
 - saved and restored from byte streams
 - persistent (outlive the current process)
- Useful for:
 - persistent storage
 - sending an object to a remote computer

19

The Default Mechanism

- The default mechanism includes:
 - The Serializable interface
 - The ObjectOutputStream
 - The ObjectInputStream



20

The Serializable Interface

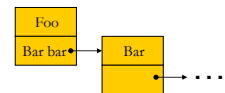
- Objects to be serialized must implement the `java.io.Serializable` interface
- An empty interface
- Most objects are `Serializable`:
 - Primitives, Strings, GUI components etc.
- Subclasses of `Serializable` classes are also `Serializable`

21

Recursive Serialization

- Can we serialize a `Foo` object?

```
public class Foo implements Serializable {  
    private Bar bar;  
    ...  
}
```



```
public class Bar {...}
```

- No, since `Bar` is not `Serializable`
- Solution:
 - Implement `Bar` as `Serializable`
 - Mark the `bar` field of `Foo` as `transient`

22

Writing Objects

- Writing a `HashMap` object (`map`) to a file*:

```
try {  
    FileOutputStream fileOut =  
        new FileOutputStream("map.s");  
  
    ObjectOutputStream out =  
        new ObjectOutputStream(fileOut);  
  
    out.writeObject(map);  
} catch (Exception e) {...}
```

* `HashMap` is `Serializable`

23

Reading Objects

```
try {  
    FileInputStream fileIn =  
        new FileInputStream("map.s");  
  
    ObjectInputStream in =  
        new ObjectInputStream(fileIn);  
  
    Map h = (Map) in.readObject();  
} catch (Exception e) {...}
```

24