

ИЗКЛЮЧЕНИЯ И ВЛОЖЕНИ КЛАСОВЕ

Ненко Табаков

Пламен Танов

Технологическо училище “Електронни системи”

Технически университет – София

23 септември 2008



ИЗКЛЮЧЕНИЯ И ВЛОЖЕНИ КЛАСОВЕ

Забележка: Тази лекция е адаптация на лекция от курса:

• 6.092 Java Preparation for 6.170, Януари 2006

- Lucy Mendel
- Corey McCaffrey
- Rob Toscano
- Justin Mazolla Paluska
- Scott Osler
- Ray He

Интернет адрес:

<http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-092January--IAP--2006/CourseHome/index.htm>

Лиценз: Creative Commons – BY – NC – SA

СЪДЪРЖАНИЕ

- Изключения и вложени (*nested*) класове
 - Изключения
 - Същност
 - Прихващане и обработка
 - Генериране на изключения
 - Вложени класове
 - Статични
 - Вътрешни
 - Локални
 - Анонимни

ИЗКЛЮЧЕНИЯ

- Помагат за обработката на възникнали изключителни ситуации
- Изключението не може просто да бъде изпуснато
- Последователност при възникване на изключение
 1. Прекъсва се нормалното изпълнение на програмата
 2. Търси се код за обработка на възникналото изключение

```
try {  
    // statement(s) that might throw exception  
} catch (ExceptionTypeA name) {  
    // handle or report exceptionTypeA  
} catch (ExceptionTypeB name) {  
    // handle or report exceptionTypeB  
} finally {  
    // clean-up statement(s)  
}
```

ПРИМЕР

Обработка на изключения

```
class Editor {
    boolean fileOpen = false;
    public boolean openFile(String filename) {
        try {
            fileOpen = true;
            File f = new File(filename);
            // действия с f
            return true;
        } catch (FileNotFoundException e) {
            // изпълнява се само при съответното изключение:
            e.printStackTrace();
            return false;
        } finally { // изпълнява се винаги:
            fileOpen = false;
        }
    }
}
```

ПРИМЕР

Генериране на изключения

- Използва се ключовата дума *throw*, следвана от обект, който да съхранява информация за причината за възникване
- При дефиницията на конструктор или метод се описват изключенията, които могат да възникнат при изпълнение посредством ключовата дума *throws*

```
public class File {  
    public File(String filename) throws FileNotFoundException {  
        // ...  
        if ( /* файлът не съществува */ ) {  
            throw new FileNotFoundException();  
        }  
    }  
}
```

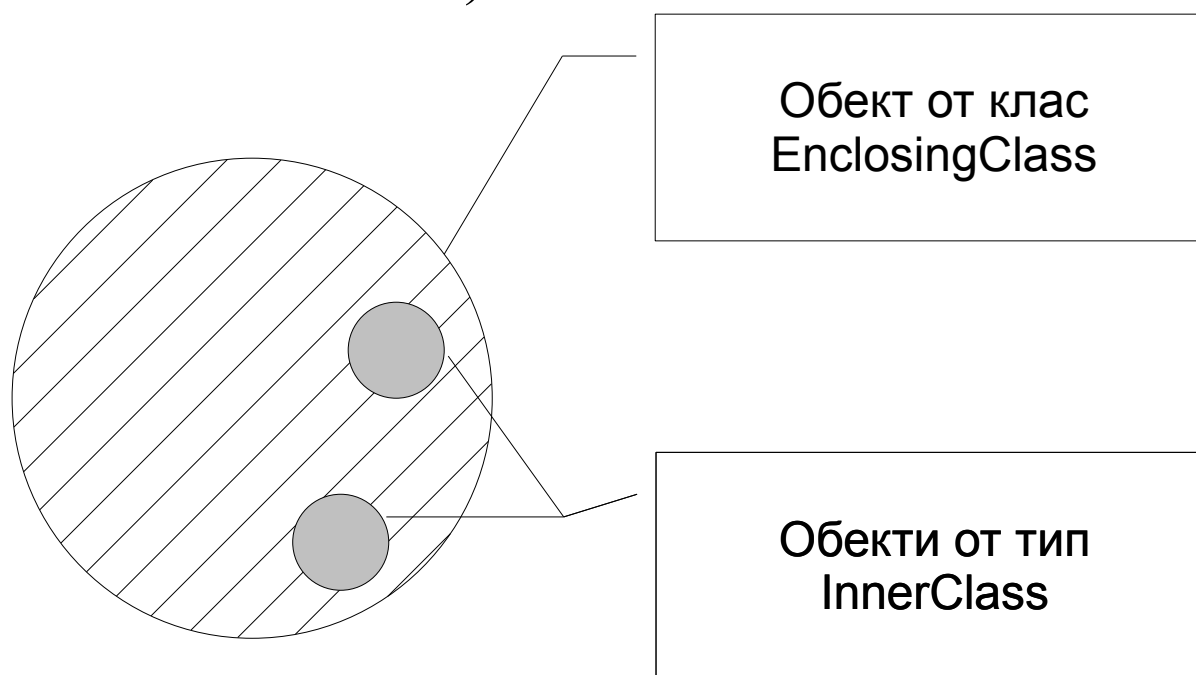
ВЛОЖЕНИ (*NESTED*) КЛАСОВЕ

- Имат достъп до всички полета на съдържащия клас (дори *private* полетата)
- Могат да бъдат статични (също така *final*, *abstract*) – има достъп само до статичните полета
- Не-статичните вложени класове се наричат още и вътрешни класове (*inner classes*)

```
class EnclosingClass {  
    static class StaticNestedClass {  
        // ...  
    }  
    class InnerClass {  
        // ...  
    }  
}
```

ВЪТРЕШНИ КЛАСОВЕ (не-статични вложени класове)

- Има достъп до всичките полета на съдържащия го клас
- Не могат да имат статични полета
- Не може да съществува без съдържащия го клас (първо той трябва да се създаде)



ЛОКАЛНИ ВЪТРСШЕНИ КЛАСОВЕ

```
import java.util.ArrayList;
import java.util.Iterator;
public class Stack {
    private ArrayList items;
    public Iterator iterator() {
        // клас дефиниран в тялото на функцията:
        class StackIterator implements Iterator {
            int currentItem = items.size() - 1;
            public boolean hasNext() { /* ... */ }
            public ArrayList<Object> next() { /* ... */ }
            public void remove() { /* ... */ }
        }
        // видим е само в границите на самата функция
        return new StackIterator();
    }
}
```

АНОНИМНИ ВЪТРСНИ КЛАСОВЕ

```
import java.util.ArrayList;
import java.util.Iterator;
public class Stack {
    private ArrayList items;

    public Iterator iterator() {
        // нямат име и се създават по този начин:
        return new Iterator() { // реализира интерфейса Iterator
            int currentItem = items.size() - 1;
            public boolean hasNext() { /* ... */ }
            public ArrayList<Object> next() { /* ... */ }
            public void remove() { /* ... */ }
        };
    }
}
```