ХЕШИРАНЕ

Ненко Табаков
Пламен Танов
Технологическо училище “Електронни системи”
Технически университет – София

18 септември 2008

ХЕШИРАНЕ

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

СЪДЪРЖАНИЕ

- Хеширане
  - Идея
  - Изисквания
  - Използване
  - Често срещани грешки

ХЕШИРАНЕ

- **Число**, което описва даден обект
- Предоставя бърз начин за проверка за еднаквост на два обекта
- Логически **еднаквите** обекти трябва да имат **еднакъв** хеш

МЕТОД ***hashCode()***

- Метод, който връща целочислена стойност (**int**)
- Дефиниран е в **java.lang.Object**
- Връща стойност, която трябва да бъде (по възможност) уникална за дадения обект
- Всеки клас притежава такъв метод (тъй като или го наследява от **Object** или го предефинира)

ИЗИСКВАНИЯ КЪМ ХЕШ КОДА НА ОБЕКТ

- Хеш кодът на обект се променя **единствено и само когато** обектът бъде **променен**
- Два **еднакви** обекта трябва да имат **еднакъв** хеш код
- Препоръчително е два **различни** обекта да имат **различни** хеш кодове

ПРИМЕР
хеш код

```
String scott = "Scotty";
String scott2 = "Scotty";
String corey = "Corey";
System.out.println(scott.hashCode());
System.out.println(scott2.hashCode());
System.out.println(corey.hashCode());
// => -1823897190, -1823897190, 65295514

Integer int1 = 123456789;
Integer int2 = 123456789;
System.out.println(int1.hashCode());
System.out.println(int2.hashCode());
// => 123456789, 123456789
```

ПРИМЕР
клас Name и equals()

```
public  class Name {
   public String  first;
   public String  last;

   public Name(String first, String last) {
      this.first = first;
      this.last = last;
   }

   public String toString() {
      return first + " " + last;
   }

   public boolean equals(Object o) {
      return (o instanceof Name &&
            ((Name) o).first.equals(this.first) &&
            ((Name) o).last.equals(this.last));
   }
}
```

ПРИМЕР
Работи ли класът Name?

```
Name kyle = new Name("Kyle", "MacLaughlin");
Name jack = new Name("Jack", "Nance");
Name jack2 = new Name("Jack", "Nance");

System.out.println(kyle.equals(jack));  // ⇒false
System.out.println(jack.equals(jack2)); // ⇒true

System.out.println(kyle.hashCode());    // ⇒6718604
System.out.println(jack.hashCode());    // ⇒7122755
System.out.println(jack2.hashCode());   // ⇒14718739

// Обектите са еднакви,
// но имат различен хеш код (това е грешно)!
```

Необходимост от *hashCode()*

- Ако **не предефинираме** този метод противоречим на изискванията към хеш код на обект
- Това може да доведе до **неочаквани** резултати:

```
Set<String> strings = new HashSet<String>();
Set<Name> names = new HashSet<Name>();

strings.add("jack");
names.add(new Name("Jack", "Nance"));

System.out.println(strings.contains("jack"));
System.out.println(names.contains(new Name("Jack",
"Nance")));
// => true, false
```

РЕШЕНИЕ НА ПРОБЛЕМА

За да се избегнат неочакваните резултати хеш кодът на даден обект (резултатът, върнат от **hashCode()**) трябва да е:

- Целочислена стойност
- Един и същ при отделните извиквания, докато обектът не бъде променен
- Еднакъв за еднакви обекти
- Различен за различни обекти

ПРИМЕР
Възможна реализация

```
public class Name {
    // ...
    public int hashCode() {
        return first.hashCode() + last.hashCode();
    }
}


// ...
Set<Name> names = new HashSet<Name>();
names.add(jack);
System.out.println(names.contains(new Name("Jack",
"Nance")));
// ⇒true
```

По-добра реализация

- Хеш кодът на обект се променя **единствено и само когато** обектът бъде **променен**
- Два **еднакви** обекта трябва да имат **еднакъв** хеш код
- Препоръчително е два **различни** обекта да имат **различни** хеш кодове
  - Пример: ***Jack Nance*** да има различен хеш код от ***Nance Jack***

```
public class Name {
    // ...
    public int hashCode() {
        return first.hashCode() * 37 + last.hashCode();
    }
}
```