Лекция 6.5: Коллекции / HashMap/HashSet, hashCode().

Приветствуем, pafnat

Настройки профиля
Выйти

Ваша успеваемость

Java Core	14.34%
1. Основы Java	88.07%
2. Базовые алгоритмы	24.13%
3. Исключения	31.21%
4. Ввод/вывод	0%
5. Многопоточность	0%
6. Коллекции	0%
7. ООП: Синтаксис	0%
8. ООП: Шаблоны	0%
9. Продвинутые возможности	0%
10. Java 8	0%

HashSet/HashMap

hashCode()

Тест

Для прохождения теста по теме Coll.HashCodeEquals (уровень сложности теста: Basic) нажмите “Start Quiz”

Тест

Для прохождения теста по теме Coll.Equals.Algebra (уровень сложности теста: Hard) нажмите “Start Quiz”

Первая попытка (нет hashCode)

Возьмем вот такую реализацию класса Человек{имя, возраст}

```java
import java.util.Objects;

public class PersonA {
    public final String name;
    public final int age;

    public PersonA(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null || this.getClass() != obj.getClass()) {
            return false;
        }
        PersonA that = (PersonA) obj;

        // 'age' comparing
        if (age != that.age) {
            return false;
        }
        // 'name' comparing
        return Objects.equals(this.name, that.name);
    }

    @Override
    public String toString() {
        return "PersonA{name='" + name + '\'' + ", age=" + age + '}';
    }
}
```

Посмотрим, как она ведет себя с List-ом (ХОРОШО ведет)

```java
import java.util.ArrayList;
import java.util.Collection;

public class App {
    public static void main(String[] args) {
        Collection<PersonA> coll = new ArrayList<>();
        coll.add(new PersonA("Mike", 45));
        System.out.println("coll = " + coll);
        System.out.println("contains: " + coll.contains(new PersonA("Mike", 45)));
        System.out.println("remove: " + coll.remove(new PersonA("Mike", 45)));
        System.out.println("coll = " + coll);
        System.out.println("contains: " + coll.contains(new PersonA("Mike", 45)));
        System.out.println("remove: " + coll.remove(new PersonA("Mike", 45)));
    }
}

>> coll = [PersonA{name='Mike', age=45}]
>> contains: true
>> remove: true
>> collt = []
>> contains: false
>> remove: false
```

Посмотрим, как она ведет себя с HashSet-ом (ПЛОХО ведет: не обнаруживает в коллекции и не удаляет)

```java
import java.util.Collection;
import java.util.HashSet;

public class App {
    public static void main(String[] args) {
        Collection<PersonA> coll = new HashSet<>();
        coll.add(new PersonA("Mike", 45));
        System.out.println("coll = " + coll);
        System.out.println("contains: " + coll.contains(new PersonA("Mike", 45)));
        System.out.println("remove: " + coll.remove(new PersonA("Mike", 45)));
        System.out.println("coll = " + coll);
    }
}

>> coll = [PersonA{name='Mike', age=45}]
>> contains: false
>> remove: false
>> coll = [PersonA{name='Mike', age=45}]
```

А вот для стандартного класса java.lang.String – все отлично

```java
import java.util.Collection;
import java.util.HashSet;

public class App {
    public static void main(String[] args) {
        Collection<String> coll = new HashSet<>();
        coll.add("Mike-45");
        System.out.println("coll = " + coll);
        System.out.println("contains: " + coll.contains("Mike-45"));
        System.out.println("remove: " + coll.remove("Mike-45"));
        System.out.println("coll = " + coll);
        System.out.println("contains: " + coll.contains("Mike-45"));
        System.out.println("remove: " + coll.remove("Mike-45"));
    }
}

>> coll = [Mike-45]
>> contains: true
>> remove: true
>> coll = []
>> contains: false
>> remove: false
```

И для java.lang.String не допускает дубликатов

```java
import java.util.Collection;
import java.util.HashSet;

public class App {
    public static void main(String[] args) {
        Collection<String> coll = new HashSet<>();
        System.out.println(coll.add("Mike-45"));
        System.out.println(coll.add("Mike-45"));
        System.out.println("coll = " + coll);
    }
}

>> true
>> false
>> coll = [Mike-45]
```

Для нашего же класса – допускает дубликаты

```java
import java.util.Collection;
import java.util.HashSet;

public class App {
    public static void main(String[] args) {
        Collection<PersonA> coll = new HashSet<>();
        System.out.println(coll.add(new PersonA("Mike", 45)));
        System.out.println(coll.add(new PersonA("Mike", 45)));
        System.out.println("coll = " + coll);
    }
}

>> true
>> true
>> coll = [PersonA{name='Mike', age=45}, PersonA{name='Mike', age=45}]
```

Итак, наш класс каким-то образом не приспособлен для работы с HashSet, однако отлично справляется с ArrayList. Разберемся в чем дело.

Вторая попытка (hashCode == 0)

На удивление, нас спасает тривиальнейшая реализация метода hashCode

```java
public class PersonB {
    ...
    public int hashCode() {
        return 0;
    }
    ...
}
```

```java
import java.util.Objects;

public class PersonB {
    public final String name;
    public final int age;

    public PersonB(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public int hashCode() {
        return 0;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null || this.getClass() != obj.getClass()) {
            return false;
        }
        PersonB that = (PersonB) obj;

        // 'age' comparing
        if (age != that.age) {
            return false;
        }
        // 'name' comparing
        return Objects.equals(this.name, that.name);
    }

    @Override
    public String toString() {
        return "PersonB{name='" + name + '\'' + ", age=" + age + '}';
    }
}
```

Наслаждаемся эффектом (нет дубликатов, работают contains и remove)

```java
import java.util.Collection;
import java.util.HashSet;

public class App {
    public static void main(String[] args) {
        Collection<PersonB> coll = new HashSet<>();

        System.out.println("add = " + coll.add(new PersonB("Mike", 45)));
        System.out.println("add = " + coll.add(new PersonB("Mike", 45)));

        System.out.println("coll = " + coll);
        System.out.println("contains: " + coll.contains(new PersonB("Mike", 45)));
        System.out.println("remove: " + coll.remove(new PersonB("Mike", 45)));
        System.out.println("coll = " + coll);
        System.out.println("contains: " + coll.contains(new PersonB("Mike", 45)));
        System.out.println("remove: " + coll.remove(new PersonB("Mike", 45)));
    }
}

>> add = true
>> add = false
>> coll = [PersonB{name='Mike', age=45}]
>> contains: true
>> remove: true
>> coll = []
>> contains: false
>> remove: false
```

Третья попытка (hashCode == “calculate”)

Однако, лучше бы, реализовать hashCode вот так

```java
public class PersonC {
    ...
    @Override
    public int hashCode() {
        return 31 * age + ((name == null) ? 0 : name.hashCode());
    }
    ...
}
```

или вот так (что тоже самое)

```java
import java.util.Objects;

public class PersonC {
    ...
    @Override
    public int hashCode() {
        return 31 * age + Objects.hash(name);
    }
    ...
}
```

Вот полная реализация третьей версии

```java
import java.util.Objects;

public class PersonC {
    public final String name;
    public final int age;

    public PersonC(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public int hashCode() {
        return 31 * age + Objects.hash(name);
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null || this.getClass() != obj.getClass()) {
            return false;
        }
        PersonC that = (PersonC) obj;

        // 'age' comparing
        if (age != that.age) {
            return false;
        }
        // 'name' comparing
        return Objects.equals(this.name, that.name);
    }

    @Override
    public String toString() {
        return "PersonB{name='" + name + '\'' + ", age=" + age + '}';
    }
}
```

Сравнение второй и третей попыток

Так почему же третья версия ЛУЧШЕ второй?
Обе – корректны, но третья ПОЗВОЛЯЕТ HashSet РАБОТАТЬ НАМНОГО БЫСТРЕЕ.

Скорость второй реализации

```java
import java.util.Collection;
import java.util.HashSet;

public class App {
    public static void main(String[] args) {
        Collection<PersonB> coll = new HashSet<>();
        long t0 = System.nanoTime();
        for (int k = 0; k < 30_000; k++) {
            // add
            coll.add(new PersonB("", k));
        }
        long t1 = System.nanoTime();
        System.out.println("dT(add)      = " + (t1 - t0));

        long t2 = System.nanoTime();
        // contains
        coll.contains(new PersonC("", -1));
        long t3 = System.nanoTime();
        System.out.println("dT(contains) = " + (t3 - t2));
    }
}

>> dT(add)      = 1 718 008 072
>> dT(contains) = 2 153 005
```

Скорость третей реализации

```java
import java.util.Collection;
import java.util.HashSet;

public class App {
    public static void main(String[] args) {
        Collection<PersonC> coll = new HashSet<>();
        long t0 = System.nanoTime();
        for (int k = 0; k < 30_000; k++) {
            // add
            coll.add(new PersonC("", k));
        }
        long t1 = System.nanoTime();
        System.out.println("dT(add)      = " + (t1 - t0));

        long t2 = System.nanoTime();
        // contains
        coll.contains(new PersonC("", -1));
        long t3 = System.nanoTime();
        System.out.println("dT(contains) = " + (t3 - t2));
    }
}

>> dT(add)      = 20 465 537
>> dT(contains) = 973
```

Как видим на добавлении (add) выигрыш в 80 раз, на проверке вхождения (contains) – более чем в 200 раз.

HashSet/HashMap internals

Стандартная реализация hashCode

Математические требования на реализацию hashCode

Больше информации о hash-структурах

Тесты по всей лекции

Тест, состоящий из случайных вопросов тестов этой лекции

Some issues occured with quiz randomizer :(

Видео

Набор декабрь 2013
Набор октябрь 2013
Набор июль 2013
Набор апрель 2013
Набор февраль 2013
Набор январь 2013
Набор октябрь 2012

Лабораторные

???

Литература

Хорстманн, Корнелл. “Java 2. Том II. Тонкости программирования”. Издание 7
….Глава 2. Наборы данных, стр 140-143