СБЕРБАНК
ТЕХНОЛОГИИ

Лекция №4

Дата

Цели

- Ознакомиться с Java Generics
- Изучить правила использования
- Использование масок (Wildcards)
- Применить на практике свои знания о Java Generics

Обобщённое программирование — это такой подход к описанию данных и алгоритмов, который позволяет их использовать с различными типами данных без изменения их описания.

Generics введены с версии Java 1.5

generics (дженерики) или <<контейнеры типа T>> — подмножество обобщённого программирования.

До generics

```
List strList = new ArrayList();
strList.add("some text");

//ОК, хотя коллекция предназначалась для хранения строк!
strList.add(new Integer(0));
String str = (String)strList.get(0);

//Ошибка приведения типов во время выполнения
(ClassCastException) Integer i = (Integer)strList.get(0);
```

С использованием generics

```
List<String> strList = new ArrayList<String>();
strList.add("some text");
strList.add(new Integer()); // сообщение об ошибке компилятора
String str = strList.get(0);
Integer i = strList.get(0); // сообщение об ошибке компилятора
```

- Проверка типов – на этапе компиляции
- Отсутствие приведения типов

«Алмазный» синтаксис Diamonds

```
Pair<Integer, String> pair = new Pair<Integer, String>(6, " Ap
r");
```

В Java 1.7 введен “diamond” operator <>, с помощью которого можно опустить параметры типа:

```
Pair<Integer, String> pair = new Pair<>(6, " Apr");
```

Параметризация класса

```
public class Test <T>  {

    private T param;

    public Test(T param) {
        this.param = param;
    }
    public T getParam() {
        return this.param;
    }
}
```

Параметризация класса

```
class Pair<T1, T2> {
    T1 object1;
    T2 object2;

    Pair(T1 one, T2 two) {
        object1 = one;
        object2 = two;
    }

    public T1 getFirst() {
        return object1;
    }

    public T2 getSecond() {
        return object2;
    }
}
```

Параметризация класса

```
class PairOfT<T> {
    T object1;
    T object2;

    PairOfT(T one, T two) {
        object1 = one;
        object2 = two;
    }

    public T getFirst() {
        return object1;
    }

    public T getSecond() {
        return object2;
    }
}
```

Параметризация методов **(Generic methods)**

Универсальными могут быть кроме классов также и методы

```
public static <T> void fill(List<T> list, T val) {…}

public <T> T getData(T data) { return data; }
```

Пример

```
package ru.sbertech.lession4;

import java.util.ArrayList;
import java.util.List;
class Utilities {
    public static <T> void fill(List<T> list, T val) {
        for (int i = 0; i < list.size(); i++)
            list.set(i, val);
    }
}
class Test {
    public static void main(String[] args) {
        List<Integer> intList = new ArrayList<>();
        intList.add(1);
        intList.add(2);
        System.out.println("Список до обработки дженерик-методом: " + intList);
        Utilities.fill(intList, 0);
        System.out.println("Список после обработки дженерик-методом: "
                + intList);
    }
}
```

Вопрос по использованию generics

какая (-ие) из нижеприведённых строк откомпилируется без проблем?

1. List<Integer> list = **new** List<Integer>();
2. List<Integer> list = **new** ArrayList<Integer>();
3. List<Number> list = **new** ArrayList<Integer>();
4. List<Integer> list = **new** ArrayList<Number>();

Wildcards (Маски)

```
//Ошибка компиляции (нарушение типобезопасности)
List<Number> intList = new ArrayList<Integer>();
```

Wildcards (Маски)

```
//Этот код скомпилируется
List<?> intList = new ArrayList<Integer>();
```

Однако при попытке

```
intList.add(new Integer(10));
```

Получим ошибку компиляции

При использовании маски <?> мы не знаем, какого типа аргумент может быть передан.

```
List<? super Integer> intList = new ArrayList<Integer>();
intList.add(new Integer(10));
```

(PECS Producer Extends Consumer Super)

<? extends T>

```
public static Double sum(List<? extends Number> numList) {
    Double result = 0.0;
    for (Number num : numList) {
        result += num.doubleValue();
    }
    return result;
}
```

<? super T>

```
class Test {
…
public static <T> void addValue(List<? super T> dest, T value) {
      list.add(value);
}
…
}
```

Использование

```
List<Number> list = new ArrayList<>();
Test.<Number> addValue(list, new Integer(1));
Test.<Number> addValue(list, new Double(2));
```

Пример <? super T> и <? extends T>

```
public class CollectionsUtil {
      public static <T> void copy(List<? super T> dest, List<? extends
T> src) {
            for (int i=0; i<src.size(); i++)
                  dest.set(i,src.get(i));
      }
}
```

Вопросы для самоконтроля

- Как параметризировать класс
- Как параметризировать метод
- Использование <? super T>
- Использование <? extends T>

Задание для самоконтроля

Необходимо написать свой LinkedList
Методы:

- add(E e)
- add(int index, E element)
- E get(int index)
- E remove(int index)
- Iterator<E> iterator()

с использованием wildcards:

- boolean addAll(Collection c)
- boolean copy(Collection c)

ПРИЛОЖЕНИЯ