Увод

в обектно ориентираното програмиране

Задачка

- Имплементирайте List интерфейсът, който да работи с колекция от String-ове
- Имплементирайте методите:
  - String get(int index);
  - void add(String value);
  - int size();
  - void remove(int index);
  - void remove(String value);
  - int indexOf(String value);

Generics & Lambdas

Съдържание

- Generics Overview
- Generic classes
- Generic methods
- Limit possible generic types
- Lambdas Overview

що е то

Защо се използват?

Generics терминът се използва, когато в клас, метод или интерфейс използваме ТИП НА ОБЕКТ като параметър.

Чрез създаването на Generic класове, можем да използваме типове (други класове или интерфейси) като параметри на Generic класа. Това позволява един и същи код да бъде изпълняван върху различни типове обекти.

Пример за такъв клас: ArrayList

ArrayList<String>, ArrayList<Cat>, ArrayList<DatabaseRecord>, ArrayList<ArrayList<ArrayList<String>>>

Защо е хубаво?

- Проверка на типа по време на компилирането предотвратява грешки по време на изпълнението
- Премахва нуждата от кастване. Представете си, ако ArrayList приемаше само обекти:
  - List list = new ArrayList();
    list.add("hello");
    String s = (String) list.get(0);
- Позволява имплементирането на generic алгоритми, които работят върху различни типове и са типово-безопасни

имплементация

Да направим кутия

Без Generic:

```
public class Box {
    private Object object;

    public void set(Object object) {
            this.object = object;
       }
    public Object get() { return object; }
}
```

Да направим кутия

C Generic:

```
public class Box<T> {
    // T stands for "Type"
    private T t;

    public void set(T t) { this.t = t; }
    public T get() { return t; }
}
```

Да подобрим нашият List

Нека листът от първата задача да работи с различни типове данни.

Направете вашият List generic.

multi-type generics

Generic класове с повече типове

```
public interface Pair<K, V> {
    public K getKey();
    public V getValue();
}
```

```
public class OrderedPair<K, V> implements Pair<K, V> {
    private K key;
    private V value;

    public OrderedPair(K key, V value) {
        this.key = key;
        this.value = value;
    }
    public K getKey() { return key; }
    public V getValue() { return value; }
}
```

raw types

Generics без конкретен тип

Raw тип се получава когато извикаме generic клас без да указваме конкретен тип за параметър

Пример:

```
Box rawBox = new Box();
```

В този случай когато извикваме put и get ще работим с типа Object.

Пробвайте да използвате ArrayList и вашият List без да конкретизирате типа на данните.

generic методи

Generics методи

Възможно е само един от методите на клас да е generic, вместо целия клас.

```
public static <K, V> boolean compare(Pair<K, V> p1, Pair<K, V> p2) {
        return p1.getKey().equals(p2.getKey()) &&  p1.getValue().equals(p2.getValue());
 }

Pair<Integer, String> p1 = new Pair<>(1, "apple");
Pair<Integer, String> p2 = new Pair<>(2, "pear");
boolean same = Util.<Integer, String>compare(p1, p2);
```

ограничаване на типовете

Използване на полиморфизъм

За да ограничите какви типове могат да бъдат задавани на вашите класове и методи, специфицирайте ги чрез интерфейси и наследяване.

Как?

```
public class Box<T extends Number> {

    private T t;

    public void set(T t) {
        this.t = t;
    }

    public T get() {
        return t;
    }
}
```

```
Class A { /* ... */ }

interface B { /* ... */ }

interface C { /* ... */ }

class D <T extends A & B & C> { /*
... */ }
```

що е то Lambda

Защо се използват?

Ламбдата е анонимна функция. Малка, стегната, анонимна функция.

Позволява ни да не пишем излишен код и да навързваме няколко операции една след друга четимо.

Пример без лямбди

```
Thread th;
th = new Thread(new Runnable() {
        public void run() {
                //in another thread
        }
});
th.start();
```

Пример с ламбди

```
Thread th;

th = new Thread(() ->  //in another thread);

th.start();
```

Пример с итерация

```
List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9);

//External iterator

//External iterator

//Internal iterator

//Internal iterator with lambdas
```

Подобрения

Type Inference:

- не трябва да дефинирате типа
- скобите около параметъра са задължителни само при повече от 1
- при просто предаване на параметъра, той може да се пропусне, но точката се замества с четири точки

Забележка

- Дръжте ламбдите си кратки
- Те могат да бъдат блокове код, но не трябва да бъдат
- Те са просто връзката между кода
- 1 ред ламбда е повече от достатъчен

домашно

Задача

- Напишете generic метод, който да принтира масив от обекти от различен тип. Метода да се казва printArray и да работи със следния код:

```
public static void main( String args[] ) {
        // Create arrays of Integer, Double and Character
        Integer[] intArray = { 1, 2, 3, 4, 5 };
        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
        Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };

        System.out.println( "Array integerArray contains:" );
        printArray( intArray  ); // pass an Integer array

        System.out.println( "\nArray doubleArray contains:" );
        printArray( doubleArray ); // pass a Double array

        System.out.println( "\nArray characterArray contains:" );
        printArray( charArray ); // pass a Character array
}
```

Задача

- Напишете generic метод, който да връща най-големия от три сравними елемента. Метода да се казва maximum и да работи със следния код:

```
public static void main( String args[] ) {
      System.out.printf( "Max of %d, %d and %d is %d\n\n",
                  3, 4, 5, maximum( 3, 4, 5 ) );

      System.out.printf( "Maxm of %.1f,%.1f and %.1f is %.1f\n\n",
                  6.6, 8.8, 7.7, maximum( 6.6, 8.8, 7.7 ) );

      System.out.printf( "Max of %s, %s and %s is %s\n","pear",
         "apple", "orange", maximum( "pear", "apple", "orange" ) );

}
```

Задача

- Напишете generic клас, който да симулира записване в база данни. Класът да работи само с типове, които имат метод getId() и getValue(). Класът да има метод addToTable, който приема обекти от съответния тип и ги записва в подходяща структура. Да има и метод writeToDatabase, който да принтира ид и стойност от всички добавени обекти.

```
public static void main( String args[] ) {
      Database<User> db = new Database<>();

       User u1 = new User(1, “pesho”);

       User u2 = new User(2, “nepesho”);

       db.addToTable(u1);

       db.addToTable(u2);

       db.writeToDatabase();
}
```

Ресурси

- https://www.youtube.com/watch?v=1OpAgZvYXLQ
- https://docs.oracle.com/javase/tutorial/java/generics