|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Отчет по выполнению лабораторной работы 10

**Тема. РАБОТА С ДЖЕНЕРИКАМИ. СТИРАНИЕ ТИПОВ**

Дисциплина Программирование на языке Джава

Выполнил

|  |  |
| --- | --- |
| студент | Болотов Михаил |
|  | Фамилия И.О. |
| группа | ИКБО-06-19 |
|  | Номер группы |

Москва 2020

**Содержание**

[​ Теория 2](#__RefHeading___Toc4128_811265792)

[​ Понятие дженериков. 2](#__RefHeading___Toc35850_811265792)

[​ Задание 2](#__RefHeading___Toc4136_811265792)

[​ Код 3](#__RefHeading___Toc4138_811265792)

[​ GenericArray.java 3](#__RefHeading___Toc35852_811265792)

[​ Solution.java 3](#__RefHeading___Toc35854_811265792)

[​ Скриншот 4](#__RefHeading___Toc3748_811265792)

[​ Заключение 4](#__RefHeading___Toc3750_811265792)

[​ Библиографический список 4](#__RefHeading___Toc3752_811265792)

# Теория

### Понятие дженериков.

Введение в Дженерики. В JDK представлены дженерики (перевод с англ. generics), которые поддерживают абстрагирование по типам (или параметризованным типам). В объявлении классов дженерики представлены обобщенными типами, в то время как пользователи классов могут быть конкретными типами, например во время создания объекта или вызова метода.

Вы, конечно, знакомы с передачей аргументов при вызове методов в языке С++. Когда вы передаете аргументы внутри круглых скобок () в момент вызова метода, то аргументы подставляются вместо формальных параметров с которыми объявлен и описан метод. Схожим образом в generics вместо передаваемых аргументов мы передаем информацию только о типах аргументов внутри угловых скобок <> (так называемая diamond notation или алмазная запись).

Основное назначение использования дженериков — это абстракция работы над типами при работе с коллекциями («Java Collection Framework»).

Если вы знакомы с классами- коллекциями, например, такими как ArrayList, то вы знаете, что они предназначены для хранения объектов типа java.lang.Object. Используя основной принцип ООП-полиморфизм, любой подкласс класса Object может быть заменен на Object. Поскольку Object является общим корневым классом всех классов Java, то коллекция, предназначенная для хранения Object, может содержать любые классы Java. Однако есть одна проблема. Предположим, например, что вы хотите определить ArrayList из объектов класса String. То при выполнении операций, например операции add(Object) объект класса String будет неявным образом преобразовываться в Object компилятором. Тем не менее, во время поиска ответственность программиста заключается в том, чтобы явно отказаться от Object обратно в строку. Если вы непреднамеренно добавили объект не-String в коллекцию, то компилятор не сможет обнаружить ошибку, а понижающее приведение типов (от родителя к дочернему) завершится неудачно во время выполнения (будет сгенерировано ClassCastException throw).

# Задание

1. Написать метод для конвертации массива строк/чисел в список.
2. Написать класс, который умеет хранить в себе массив любых типов данных (int, long etc.).
3. Реализовать метод, который возвращает любой элемент массива по индексу.

# Код

### GenericArray.java

package dev.ky3he4ik.lab.lab10;  
  
public class GenericArray<T> {  
 private Object[] array;  
  
 GenericArray(int size) {  
 array = new Object[size];  
 }  
  
 T get(int ind) {  
 @SuppressWarnings("unchecked") final T obj = (T) array[ind];  
 return obj;  
 }  
  
 void set(int ind, final T obj) {  
 array[ind] = obj;  
 }  
}

### Solution.java

package dev.ky3he4ik.lab.lab10;

import java.util.\*;

public class Solution {

public static <T> List<T> toArray(T[] arr) {

return Arrays.asList(arr);

}

public static <T> T getElem(T[] arr, int ind) {

assert ind >= 0 && ind < arr.length;

return arr[ind];

}

public static void main(String[] args) {

String[] strings = {"test", "asd", "gtrg"};

List<String> strList = toArray(strings);

Integer[] ints = {4, 6, 7, 8};

List<Integer> intList = toArray(ints);

GenericArray<String> array = new GenericArray<>(strings.length);

for (int i = 0; i < strings.length; i++)

array.set(i, strings[i]);

GenericArray<Integer> arrayI = new GenericArray<>(ints.length);

for (int i = 0; i < ints.length; i++)

arrayI.set(i, ints[i]);

assert getElem(strings, 2).equals(strings[2]);

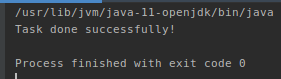
assert getElem(ints, 2).equals(ints[2]);

System.out.println("Task done successfully!");

}

}

# Скриншот



# Заключение

В данной лабораторной работе я научился работать с обобщенными типами в Java и применять их в программах.

# Библиографический список

1. Зорина Н.В. Курс лекций по Объектно-ориентированному программированию на Java, МИРЭА, Москва, 2016
2. Программирование на языке Java: работа со строками и массивами. Методические указания. [Электронный ресурс] : Учебно-методические пособия — Электрон. дан. — СПб. : ПГУПС, 2015. — 24 с.
3. Кожомбердиева, Г.И. Программирование на языке Java: создание графического интерфейса пользователя: учеб. пособие. [Электронный ресурс] : Учебные пособия / Г.И. Кожомбердиева, М.И. Гарина. — Электрон. дан. — СПб.: ПГУПС, 2012. — 67 с.
4. Вишневская, Т.И. Технология программирования. Часть 1. [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2007. — 59 с.