Министерство образования и науки Украины

Одесский Политехнический Университет

Кафедра “Информационные технологии”

Отчёт по лабораторной работе №10

На тему: «**Обобщенные типы**»

Выполнил cтудент группы:

АД-201

Стыцковский Н.Ю.

Проверили:

Рудниченко Н.Д.

Оглавление

[Цель работы: 3](#_Toc87780576)

[Теоретические сведения 3](#_Toc87780577)

[Задание 7](#_Toc87780578)

[Скриншоты 7](#_Toc87780579)

[Вывод: 7](#_Toc87780580)

Цель работы:

* Ознакомиться с механизмом обобщенных типов;
* Научиться создавать обобщенные классы, методы и интерфейсы.

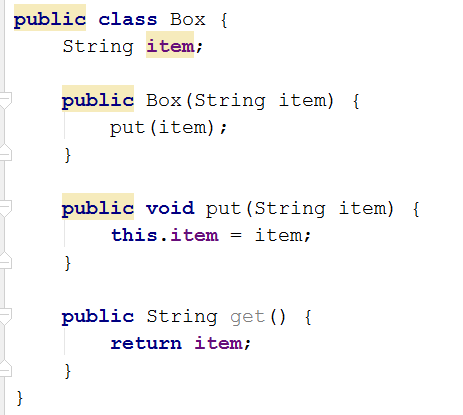
.

# Теоретические сведения

Обобщение – означает параметризированный тип. Они позволяют создавать классы, интерфейсы и методы, в которых тип данных указывается в виде параметра. С помощью обобщений можно создавать класс, который будет автоматически работать с разными типами данных. Такие классы, интерфейсы и методы называются обобщенными, как например, обобщенный класс или обобщенный метод.

Обобщенный код будет автоматический работать с типом данных, переданным ему в качестве параметра. Многие алгоритмы выполняются одинаково, независимо от того, к данным какого типа они будут применяться. Например, быстрая сортировка не зависит от типа данных, будь то String, Object или любой пользовательский класс. Используя обобщения, можно реализовать алгоритм один раз, а затем применять его без дополнительных усилий к любому типу данных.

Рассмотрим небольшой пример. Вернемся к классу Box, с которым мы работали во время выполнения первых лабораторных работ.



В данном случае наша коробка хранит объекты класса String. Но что если нам в нашей программе необходима коробка, которая бы хранила не только объекты класса String, но и любого другого класса, как стандартного из библиотек Java, так и нашего созданного.

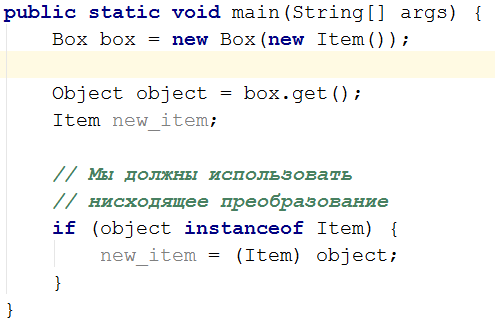
Как мы можем реализовать это с помощью уже известных нам механизмов ООП?

Мы можем использовать механизм полиморфизма. Например, мы можем использовать класс Object, который, как известно, является суперклассом для всех классов Java. Второй вариант – объявить интерфейс BoxItem и использовать его в качестве типа ссылочной переменной. Тогда в коробку можно будет положить объект класса, который реализует этот интерфейс.

|  |  |
| --- | --- |
|  |  |

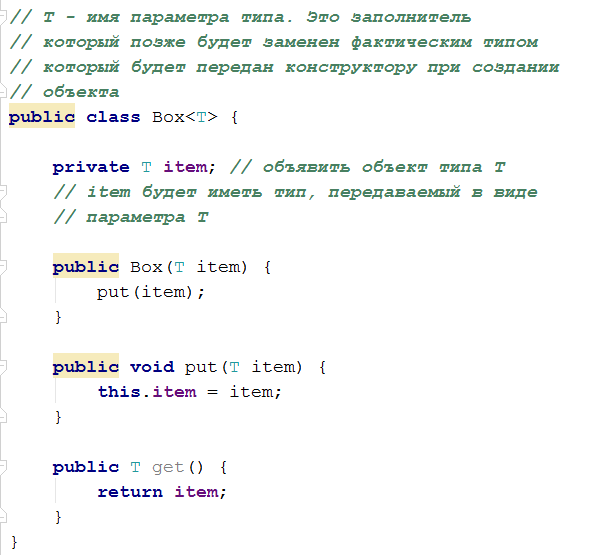
Метод стал чуть более общим и может использоваться в большем количестве мест. Однако, огромный недостаток такого подхода – необходимость нисходящего преобразования извлеченного из коробки объекта.





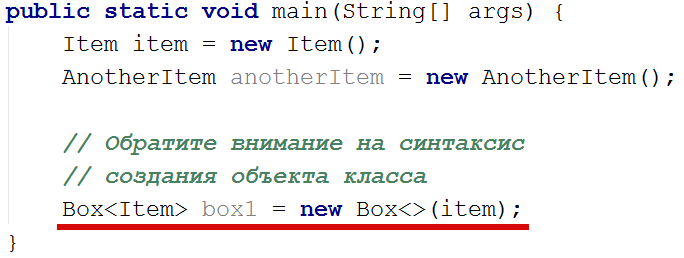
Если же мы не знаем заранее – объект какого класса будет помещен в коробку, то и нисходящее преобразование реализовать будет достаточно затруднительно.

Какой выход может быть из данной ситуации? В Java и в других ОО-языках программирования существует механизм обобщенных типов. Этот механизм позволяет нам создавать классы, методы и интерфейсы, которые будут автоматически работать с типами данных, которые будут переданы позднее, при создании объекта этого класса.

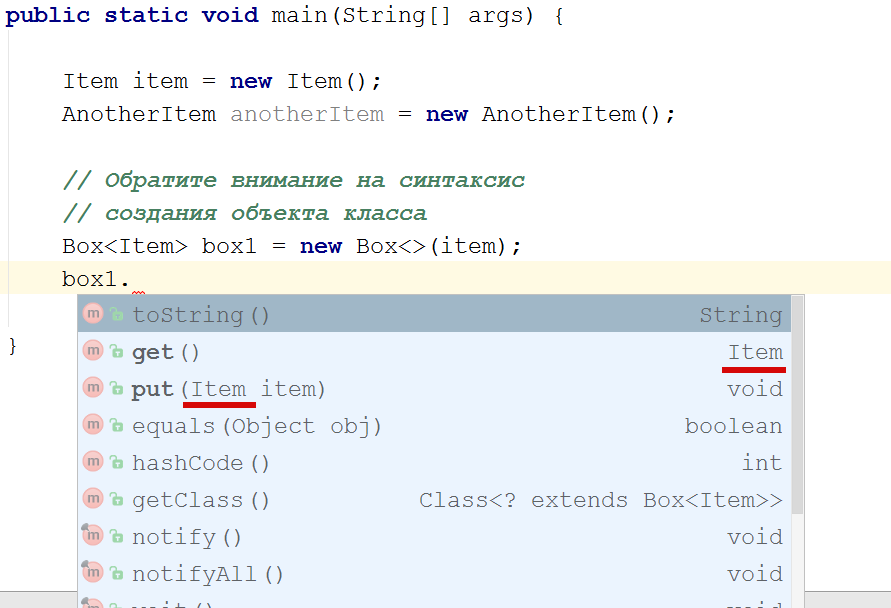


Такой класс называется обобщенным. Под термином обобщение следует понимать «применимость к большой группе классов».

Теперь создадим объект класса Box



Обратите внимание, что после создания объекта обобщенного класса, класс Item будет подставлен вместо типа T



То есть, нам нет необходимости заниматься нисходящим преобразованием и приведением типов – объект класса Box будет работать с классом Item.

Обобщения действуют только со ссылочными типами.

# Задание

3 вариант

Структура данных generic-класса односвязный список.

Generic-метод возвращает группу, в которой содержатся четные элементы из входной группы.

Листинг

package com.mycompany.lab.stytskovskyi;

import java.util.\*;

/\*\*

\*

\* @author USER

\*/

public class Main

{

public static void main(String[] args)

{

LinkedList<Integer> A = new LinkedList<> ();

LinkedList<Integer> B = new LinkedList<> ();

LinkedList<Integer> C = new LinkedList<> ();

for (int i = 0; i < 10; i++)

{

A.add((int)(0 + Math.random() \* 10));

B.add((int)(0 + Math.random() \* 10));

}

System.out.println("Group A : " + A);

System.out.println("Group B : " + B);

C.addAll(findUnique(A, B));

System.out.println("Group C : " + C);

}

public static <V extends Comparable,T extends LinkedList<V>> LinkedList findUnique(T a, T b)

{

LinkedList<V> returnList = new LinkedList<> ();

boolean unique;

for (int i = 0; i < a.size(); i++)

{

unique = true;

for (int j = 0; j < b.size(); j++)

{

if (a.get(i) == b.get(j))

unique = false;

}

if (unique == true)

returnList.add(a.get(i));

}

for (int i = 0; i < b.size(); i++)

{

unique = true;

for (int j = 0; j < a.size(); j++)

{

if (b.get(i) == a.get(j))

unique = false;

}

if (unique == true)

returnList.add(b.get(i));

}

return returnList;

}

}

# Скриншоты



# Вывод:

Мы ознакомились с механизмом обобщенных типов, научился создавать обобщенные классы, методы и интерфейсы.