**Краткий конспект**

**Generics (a.k.a Дженерики, обобщения)**

1. ***Дженерики (обобщения)*** *-* Термин Generic по сути означает параметризированный тип. Специфика параметризированных типов состоит в том, что они позволяют создавать классы, интерфейсы и методы, в которых тип данных указывается в виде параметра. Используя обобщения, можно создать единственный класс или метод, который будет автоматически работать с различными типами данных.

2. **Преимущества использования дженериков**

2.1) автоматическая работа с типом данных, переданным ему в качестве параметра. Позволяет написать один алгоритм, который будет работать с различными типами.

2.2) Проверка типов, позволяет выявить ошибку на этапе компиляции, до запуска программы.

Пример:

Код без дженериков

List numbers = new ArrayList();

// «Случайно» вставляем строку вместо числа.

numbers.add("100");

//Необходимо привести полученное число к int

//При приведении строки в число возникнет ошибка во время

//выполнения программы

int num = (int) numbers.get(0);

Код с использованием дженериков

List<Integer> numbers = new ArrayList();

// «Случайно» вставляем строку вместо числа.

// Здесь произойдет ошибка во время компиляции

numbers.add("100");

// Обратите внимание, что больше не нужно приводить число к int

int num = numbers.get(0);

3. **Пример параметризированного класса**

public class GenericClass<T> {

private T value;

public T getValue() {

return value;

}

public void setValue(T value) {

this.value = value;

}

}

4. **Пример параметризированного метода**

public <T> T getLastValue(List<T> list) {

return list.get(list.size() - 1);

}

5. **Как нельзя использовать дженерики**

* Нельзя объявлять параметризованное поле как static.

public class Forbidden<T> {

private static T value;

}

* Нельзя инициализировать параметризованный объект

new T();

* Нельзя параметризировать примитивными типами

final List<int> ids = new ArrayList<>();

}

* Нельзя создавать параметризованные исключения

class GenericException<T> extends Exception {} ]

6. **Параметры типа, границы и символы подстановки**

|  |  |
| --- | --- |
| <T> | Неограниченный тип |
| <T, P> | Неограниченные типы |
| <T extends P> | Тип, ограниченный сверху. Особый тип T, который является подтипом типа P |
| <?> | Неограниченный символ подстановки (WildCard) |
| <? extends P> | Ограниченный символ подстановки (WildCard). Некий неизвестный тип, который является подтипом типа P |
| <? super P> | Ограниченный символ подстановки (WildCard). Некий неизвестный тип, который является супертипом типа P |

7. **Особенности наследования**

