

### Вопросы по теме Дженерики:

1. Что такое дженерики?
2. Какие преимущества дает использование дженериков?
3. Что такое стирание типов?
4. Расскажите про extends и super в дженериках.
5. Что такое wildcard и как его использовать?
6. Можете ли вы передать List <String> методу, который принимает List <Object>?

### Дженерики.

Дженерики – набор свойств языка, позволяющих определять и использовать обобщенные типы (классы и интерфейсы) и методы. Обобщенные типы / методы отличаются от обычных тем, что имеют типизированные параметры. Параметром типа может быть только ссылочный тип.

Говоря более простым языком:

**Дженерик (generic types)** – это класс или интерфейс с одним или несколькими параметрами типа в заголовке. Параметр типа (type parameter) – тип объекта, который принимает дженерик, может быть только ссылочным типом данных.

```
public interface List<E> extends Collection<E> {}  
List<String> list = new ArrayList<>();
```

ТИП

ПАРАМЕТР ТИПА

АРГУМЕНТ ТИПА

ПОДТИП (относительно List)

ТИП (относительно ArrayList это супертип)

аргумент типа берется из супертипа

**Аргумент типа (type argument)** – тип объекта, который используется вместо параметра типа.

**Сырой тип (raw type)** – дженерик-тип без указания параметра типа (new ArrayList<>()).

**Неизвестный тип (wildcard)** – тип, обозначенный символом «?» (new MyObject<?>()). Вместо данного параметра типа может быть подставлен любой тип.

**Ограничение сверху** – ограничение типа дженерика, при котором вместо параметра типа можно подставить указанный класс или его класс-наследник ( <E extends MyClass> ).

**Ограничение снизу** – ограничение типа дженерика, при котором вместо параметра типа можно подставить указанный класс или его класс-предок ( <E super MyClass> ).

### Преимущества использования дженериков:

1. упрощение тестирования кода (компилятор применяет строгую проверку типов, что позволяет выявлять их несоответствие в момент компиляции кода);
2. упрощение написания кода (во многих моментах пропадает необходимость явного приведения типов);
3. делает код универсальным (дает возможность использовать разные типы данных одному классу / интерфейсу).

При обозначении параметра типа можно использовать любую букву. Однако существуют рекомендации от Oracle:

Е – элемент (обычно для коллекций), К – ключ (для Map), V – значение (для Map), N – число, Т – тип (в остальных случаях), S, U, V и т.д. – для 2-го, 3-го, 4-го типа и т.д.

Во время компиляции класса-джереника происходит **стирание информации о типе**. Все параметры без ограничений заменяются типом Object, а с ограничениями – ограничивающим типом. В некоторых случаях компилятор может применить приведение к нужному типу.

Пример (до / после компиляции) стирания типа для дженеика без ограничений:

<pre>public class Main&lt;City&gt; {     private City city;     public City getCity() {         return city; }     public void setCity(City city) {         this.city = city; } }</pre>	<pre>public class Main {     private Object city;     public Object getCity() {         return city; }     public void setCity(Object city) {         this.city = city; } }</pre>
---	---

Пример (до / после компиляции) стирания типа для дженеика с ограничением:

<pre>public class Main&lt;K extends City&gt; {     private T city;     private Main&lt;T&gt; main;     public Main(T city, Main&lt;T&gt; main) {         this.city = city;         this.main = main; } }</pre>	<pre>public class Main {     private City city;     private Main main;     public Main(City city, Main main) {         this.city = city;         this.main = main; } }</pre>
--	--

Пример (до / после компиляции) приведения типа к нужному:

<pre>List&lt;String&gt; list = new ArrayList&lt;&gt;(); list.add("London"); String city = list.get(0);</pre>	<pre>List list = new ArrayList&lt;&gt;(); list.add((String) "London"); String city = (String) list.get(0);</pre>
--	--