Вопросы по теме Дженерики:

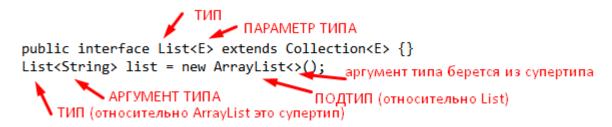
- 1. Что такое дженерики?
- 2. Какие преимущества дает использование дженериков?
- 3. Что такое стирание типов?
- 4. Расскажите про extends и super в дженериках.
- 5. Что такое wildcard и как его использовать?
- 6. Можете ли вы передать List <String> методу, который принимает List <Object>?

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.



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:

E – элемент (обычно для коллекций), K – ключ (для Мар), V – значение (для Мар), N – число, T – 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-дженерика 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: public class Main<City> { public class Main { private City city; private Object city; public City getCity() { public Object getCity() { return city; } return city; } public void setCity(City city) { public void setCity(Object city) { this.city = city; } this.city = city; } } } Пример (до / после компиляции) стирания типа для дженеика с ограничением: public class Main<K extends City> { public class Main { private T city; private City city; private Main<T> main; private Main main; public Main(T city, Main<T> main) { public Main(City city, Main main) { this.city = city; this.city = city; this.main = main; } this.main = main; } } } Пример (до / после компиляции) приведения типа к нужному: List<String> list = new ArrayList<>(); List list = new ArrayList<>(); list.add("London"); list.add((String) "London");

String city = list.get(0);

String city = (String) list.get(0);