

Generics: Class<T>

Java Collections
5 уровень, 7 лекция

ОТКРЫТА

— Привет! Я продолжу лекцию Элли про Generic'и. Готов слушать?

— Ага.

— Тогда начинаем.

Факт первый. У методов класса тоже могут быть свои типы-параметры.

— Да, я знаю.

— Нет, я имею ввиду именно **свои типы-параметры**:

Пример

```
1  class Calculator
2  {
3      T add(T a, T b); //сложить
4      T sub(T a, T b); //отнять
5      T mul(T a, T b); //умножить
6      T div(T a, T b); //делить
7  }
```

Это типы-параметры именно метода(ов). У класса параметров нет. Можно даже объявить методы статическими и вызывать их без использования объекта.

— Ясно. Смысл типов-параметров в методах такой же, как и в классах?

— Ага. Но есть и кое-что новое.

Как ты уже знаешь, в описании типа можно использовать wildcard. Тогда представь себе ситуацию:

Пример 1

```
1  public void doSomething(List<? extends MyClass> list)
2  {
3      for(MyClass object : list)
4      {
5          System.out.println(object.getState()); //тут все работает отлично.
6      }
7  }
```

А вот, что случится, если мы захотим добавить в коллекцию новый элемент:

Пример 2

```
1  public void doSomething(List<? extends MyClass> list)
2  {
3      list.add(new MyClass());
4  }
```

НАЧАТЬ ОБУЧЕНИЕ

Дело в том, что в общем случае в метод doSomething можно передать List с типом элементов не MyClass, а любой из наследников MyClass. А в такой список заносить объекты MyClass уже нельзя!

— Ага. И что же делать?

— Ничего. Прямо в этой ситуации – ничего не сделаешь. Но это дало разработчикам Java повод для размышлений. И они придумали новое ключевое слово – **super**.

Выглядит его использование практически так же:

1	List<? super MyClass> list
---	-----------------------------------

Но между extends и super есть существенное различие.

«? extends T» обозначает, что класс должен быть наследником T.

«? super T» обозначает, что класс должен быть предком T.

— Ух ты. А где это используется?

— «? super T» используется, когда метод собирается добавлять в коллекцию объект типа T. Тогда это может быть коллекция типа T или любого типа-предка T.

— Ага. Ссылку на объект типа T можно же присвоить любому родительскому типу для T.

— Честно говоря – этот подход используется не очень часто. Тем более, что у него есть и обратная сторона. Пример:

Примеры	
1	public void doSomething(List<? super MyClass> list)
2	{
3	for(MyClass object : list) //ошибка!
4	{
5	System.out.println(object.getState());
6	}
7	}

1	public void doSomething(List<? super MyClass> list)
2	{
3	list.add(new MyClass()); //тут все работает отлично.
4	}

Теперь не работает первый пример.

Т.к. коллекция list может быть даже List<Object> (Object самый верхний родитель MyClass), то фактически мы пишем такой код, а так писать нельзя:

Пример 1	
1	List<Object> list;
2	
3	for(MyClass object : list) //ошибка!
4	{
5	System.out.println(object.getState());
6	}

— Ясно. Спасибо за интересную лекцию.

− +62 +

Комментарии (53)

популярные новые старые

JavaCoder

Введите текст комментария

Igor Petrashevsky Уровень 44

25 августа, 03:00 ...

какой набор костылей. и эти люди критикуют php и js

Ответить − 0 +

Фарид Гулиев Уровень 41, Днепр, Украина

5 июля, 19:12 ...

Для тех кто не понял, почему мы не может пройтись по такой конструкции <? super MyClass> вот таким циклом:

1

for(MyClass myClass : list){ //Произойдёт ошибка!

2

3

}

Дело в том, что когда мы пишем <? super MyClass> мы разрешаем использовать только объекты класса MyClass и всех его родителей. Значит, в таком списке будут лежать объекты класса, которые находятся выше по иерархии(вплоть до класса Object). И если мы будет проходить именно объектом MyClass(который в самом внизу заданной иерархии), то будем присваивать объекту myClass присваивать объекты классов, которые могут находиться выше. А как мы помним, в языке Java **downcast** неявно не происходит, его надо прописывать вручную(в отличии от **upcast**а), а так как это всё происходит в цикле forEach, то мы не в силе самим провести **downcast**, поэтому и будет выкинуто исключение. Не знаю, почему в лекции не написано это(мои догадки, возможно оно всё **НЕ** так), или же, могли написать что стоит проходится объектом класса Object.

Ответить − +1 +

LuneFox инженер по сопровождению в BIFIT EXPERT

31 января, 19:12 ...

А если я хочу метод, который принимает списки Animal, Dog и Cat? (Dog и Cat - наследники Animal). Если напишу <? extends Animal>, то не смогу добавлять Animal. Если напишу <? super Cat>, то не смогу добавлять собак.

Ответить − 0 +

Дмитрий Рыбин Уровень 40, Краснодар, Россия

2 февраля, 02:26 ...

1

List<Animal> arr = **new** ArrayList<>();

2

arr.add(**new** Cat ());

3

arr.add(**new** Dog ());

4

arr.add(**new** Animal ());

Ответить − 0 +

LuneFox инженер по сопровождению в BIFIT EXPERT

2 февраля, 18:43 ...

Без вайлдкардов? А если при этом мне нужно, скажем, исключить из добавления слонов?

Ответить − 0 +

Борис Уровень 33

16 апреля, 11:47 ...

Наследование от одного класса? После Pets & можно указывать только интерфейсы?

Ответить − 0 +

On1k Уровень 42, Krasnogorsk, Russian Federation

7 июля, 09:04 ...

А зачем вам добавлять Animal? Это же абстрактный класс по идее и создавать объекты именно этого типа вряд ли понадобится. Хотя мало ли что в голове у программиста)

полиморфизм)

Ответить

0

On1k

Уровень 42, Krasnogorsk, Russian Federation

7 июля, 21:51

Так в этом то и суть на мой взгляд, чтобы использовать список для хранения разных животных вы и создаете абстрактный класс Animal, а уже в списке храните всех ваших животных, которых можно привести к классу Animal.
Если конечно у вас нет для всех животных отдельного класса, тогда для таких животных можно завести класс SomeAnimal extends Animal, которых вы и будете запихивать в лист.
Либо я пока чего то не понимаю =)

Ответить

0

Ars

Уровень 41

22 ноября 2021, 14:33

Факт первый. У методов класса тоже могут быть свои типы-параметры.
— Да, я знаю.
— Нет, я имею ввиду именно свои типы-параметры:
— Да, я знаю. Я только сделал задачу где у метода были свои типы параметры!

Ответить

+5

Artamon Khakimov

Уровень 35, Казань, Россия

16 ноября 2021, 18:47

Как я это понял.
Допускаются все наследники класса MyClass. Сам же MyClass входить не будет.

1

<? extends MyClass>

Если же мы хотим допускать значение MyClass и все классы выше MyClass, то используем

1

<? super MyClass>

Ответить

+1

Е К

Уровень 41, Краснодар, Россия

22 июня 2021, 22:42

[для закрепления](#) - жмём next и впитываем официальную точку зрения на всё это блинство

Ответить

+4

Vadim

Уровень 36, США

24 мая 2021, 21:52

жаль, что к 35 уровню перестали объяснять на примере класса кошек и животных, сразу понятнее становится)

Ответить

+1

Pig Man

Главная свинья в Свинарнике

10 февраля 2021, 19:35

1

Object

2

Родитель1

3

Родитель2

4

MyClass

5

Наследник1

6

Наследник2

List<? extends MyClass> - ? в данном случае все, что ниже класса MyClass (включительно). Поэтому мы можем написать *for(MyClass object : list)*, так как объект в списке точно подойдет под MyClass. Но не можем положить MyClass в список, так как не факт, что он подойдет (как не могли бы положить в список с MyClass объект типа Object)

List<? super MyClass> - ? в данном случае все, что выше MyClass (включительно). Поэтому мы можем положить в такой список MyClass, так как он точно подойдет (как могли бы положить в список с Object наш MyClass). Но мы не можем написать *for(MyClass object : list)*, потому что не факт, что он подойдет

Ответить

+24

Илья

Backend Developer в СберТех

10 февраля 2021, 14:26

Модераторы!
Просьба обратить внимание на данный участок лекции

1

class Calculator

2

{

3

T add(T a, T b); //сложить

4

T sub(T a, T b); //отнять

5

T mul(T a, T b); //умножить

НАЧАТЬ ОБУЧЕНИЕ

перед каждым T необходимо поставить <T>:

1 {
2 <T>T add(T a, T b); //сложить
3 <T>T sub(T a, T b); //отнять
4 <T>T mul(T a, T b); //умножить
5 <T>T div(T a, T b); //делить
6 }

Ответить

+16👍👎🔥

wan-derer.ruУровень 40, Москва, Россия18 декабря 2020, 13:48⋮

Здорово, но непонятно. А что делать если надо и добавлять элементы, и проходить по списку? Объявить 2 разных метода: один для добавления, второй для итераций?

Ответить

+1👍👎

СергейУровень 382 января 2021, 18:03⋮

Прочитайте про принцип PECS. В данном случае wildcard не используется. Где-то к комментах была ссылка http://java-online.ru/java-generic.xhtml - см. п6. Но так или иначе приходим к использованию параметра типа без ограничений сверху/снизу.

Ответить

0👍👎

↻ Показать еще комментарии

ОБУЧЕНИЕ

- Курсы программирования
- Курс Java
- Помощь по задачам
- Подписки
- Задачи-игры

СООБЩЕСТВО

- Пользователи
- Статьи
- Форум
- Чат
- Истории успеха
- Активности

КОМПАНИЯ

- О нас
- Контакты
- Отзывы
- FAQ
- Поддержка



JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

 Русский

▼

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

