Поиск

Карта квестов Лекции CS50 Android

Generics: super, extends, list

Java Collections 5 уровень, 5 лекция

ОТКРЫТА

— Новая интересная тема — wildcards.

По смыслу, это что-то вроде шаблона «*», который совпадает с чем угодно.

Но давай начнем издалека.

Представь, что у тебя есть класс «Warrior(Воин)» и метод, который вычисляет, какой из двух воинов сильнее. Вот как, например, это могло бы выглядеть:

```
Пример вызова

1 MagicWarrior mag = new MagicWarrior();
2 ArcherWarrior archer = new ArcherWarrior();
3
4 boolean isMagicCooler = WarriorManager.fight(mag, archer);
```

MagicWarrior и ArcherWarrior – это классы-наследники от Warrior.

Немного простовато, но для примера сойдет.

— Ок.

— И вот допустим, ты решил сделать аналогичный метод, где уже дерутся воины стенка на стенку.

```
Пример 1
      class WarriorManager
 1
 2
      {
       public static boolean fight(Warrior w1, Warrior w2)
 3
 4
       {
 5
        return w1.power > w2.power;
 6
       }
 7
       public static boolean fight(List<Warrior> w1, List<Warrior> w2)
 8
 9
10
        return ...
```

Пример вызова ArrayList<MagicWarrior> magi = new ArrayList<MagicWarrior>(); 1 2 for(int i=0;i<10;i++)</pre> 3 magi.add(new MagicWarrior()); 4 ArrayList<ArcherWarrior> archers = new ArrayList<ArcherWarrior>(); 5 for(int i=0;i<10;i++)</pre> 6 7 archers.add(new ArcherWarrior()); 8 9 boolean isMagicCooler = WarriorManager.fight(magi, archers); //ошибка компиляции!

И вот тут ты встречаешь ошибку компиляции и недоумеваешь, а что не так-то?

Все дело в том, что MagicWarrior – это наследник Warrior и его объекты можно передавать в метод fight(Warrior, Warrior)

A вот List<MagicWarrior> — это не наследник List<Warrior>. И передавать его туда нельзя!

- Как это не наследник?
- А вот так. То List и то List. Пусть и с параметрами.
- Да, а я как-то сразу не обратил внимание. И что, есть уже решение этой проблемы?
- Ага. Для этого используется более сложная конструкция. Выглядит это так:

```
Пример 1
      class WarriorManager
 1
 2
       public static boolean fight(Warrior w1, Warrior w2)
 3
       {
 4
 5
        return w1.power > w2.power;
       }
 6
 7
 8
       public static boolean fight(List<? extends Warrior> w1, List<? extends Warrior> w2)
       {
 9
        return ...
10
11
       }
12
      }
```

«? extends Warrior» обозначает «любой тип, унаследованный от Warrior ».

T.e. туда можно передать список List<MagicWarrior> и список List<ArcherWarrior>, и все будет работать.

- Так это же отлично. Чем меньше всяких таких проблем, тем лучше.
- Так и я о том же.
- А если мне не нужен наследник Warrior? Что если я хочу, чтобы в метод можно было передать любой List с любым параметром? Так можно?
- Да, для описания этой ситуации существует две записи:

```
1 List<? extends Object>
```

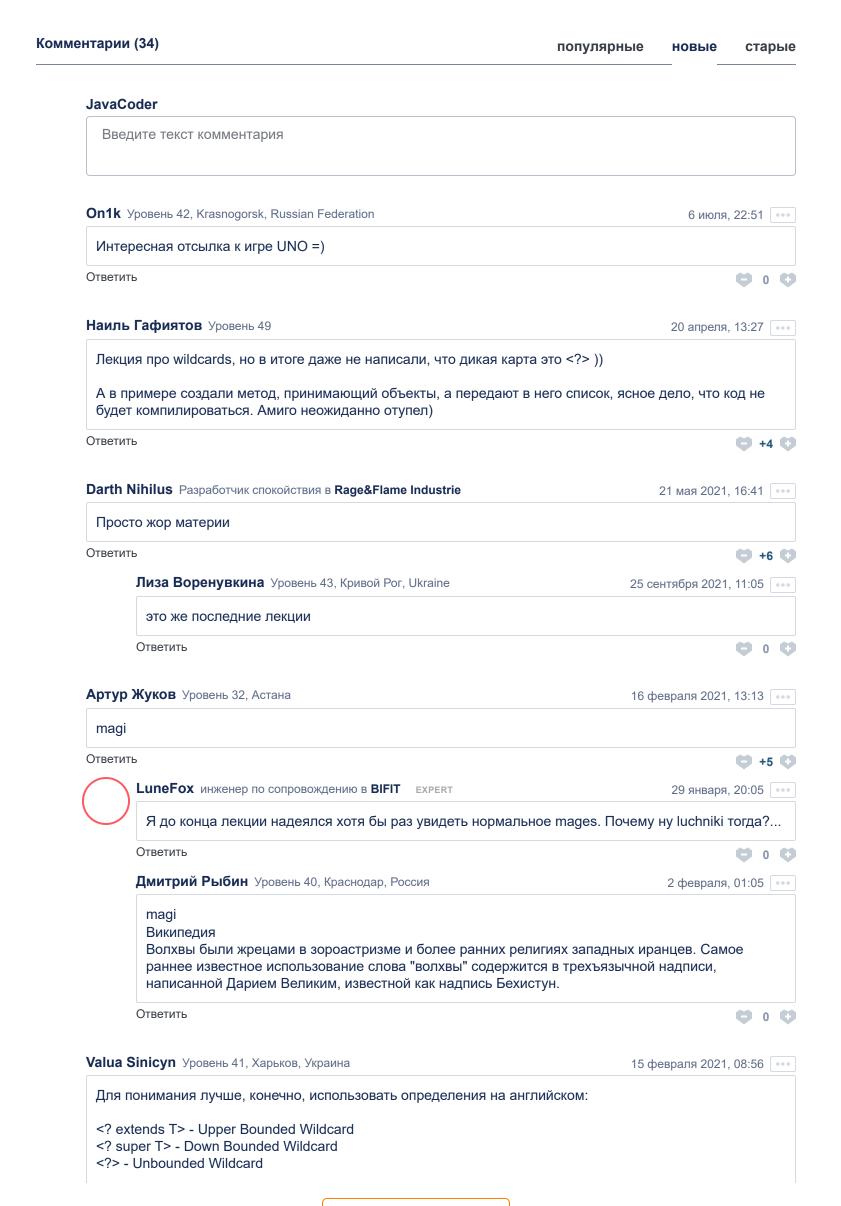
1 List<?>

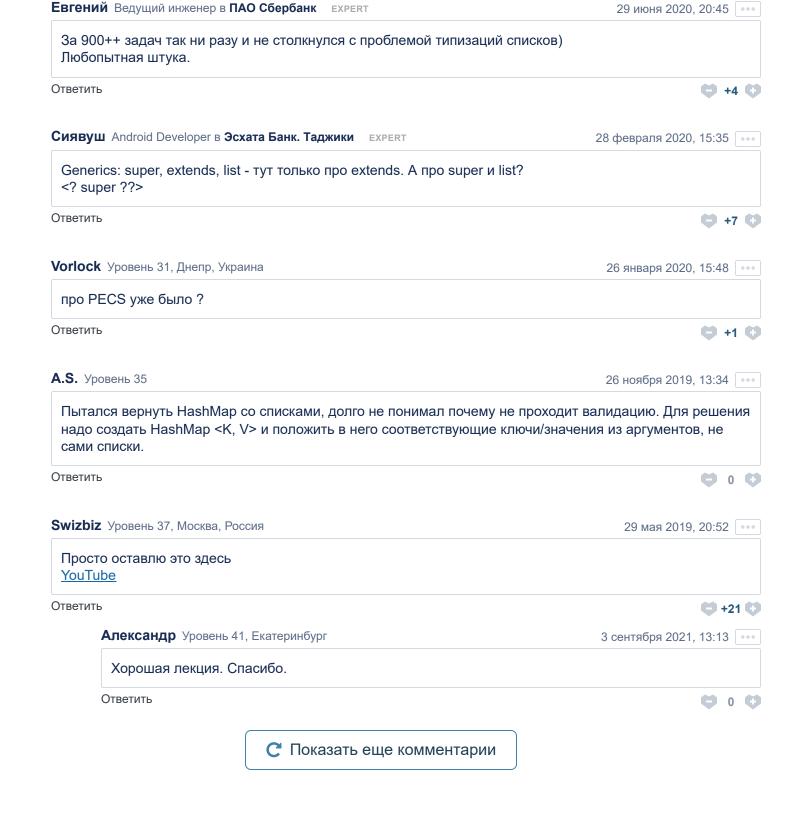
— Спасибо, Элли, действительно узнал сегодня много нового.

< Предыдущая лекция

Следующая лекция >







ОБУЧЕНИЕ СООБЩЕСТВО КОМПАНИЯ Онас Курсы программирования Пользователи Kypc Java Статьи Контакты Помощь по задачам Форум Отзывы Подписки Чат **FAQ** Задачи-игры Истории успеха Поддержка Активности



RUSH

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





"Программистами не рождаются" © 2022 JavaRush