一、介绍

1、简介

```
泛型 => Integer,String,Dog
```

2、特点

```
1）泛型又称参数化类型，是JDK5.0出现的新特性，解决数据类型的安全性问题
2）在类声明或实例化时只要指定好需要的类型即可
3）Java泛型可以保证如果程序在编译时没有发出警告，运行时就不会产生ClassCastException异常。同时，代码更加简介、健壮
4）泛型的作用：可以在类声明时通过一个标识表达类中某个属性的类型，或者是某个方法的返回值的类型，或者是参数类型
```

二、基本语法

1、声明

```
interface 接口<T> {} 和 class 类 <K,V> {}
1）其中T,K,V不代表值，而是表示类型
2）任意字母都可以。常用T表示，是Tyep的缩写
```

2、实例化

```
1）List<String> strList = new ArrayList<String> ();
2）Iterator <Customer> iterator = customers.iterator();
```

3、细节

```
1）给泛型指向数据类型时，必须是引用类型，不能是基本数据类型
2）在给泛型指定具体类型后，可以传入该类型或者其子类类型
3）在实际开发中，常使用简写：List<Integer> list = new ArrayList<>()
4）不指定泛型时，默认为Object
```

三、自定义泛型

1、语法

```
class 类名 <T,R...> {
    成员
}
```

2、细节

```
1) 普通成员可以使用泛型(属性、方法)
2) 使用泛型的数组，不能初始化
3) 静态方法中不能使用类的泛型
4) 泛型类的类型，是在创建对象时确定的（因为创建对象时，需要指定确定类型）
5) 如果在创建对象时，没有指定类型，默认为Object
```

3、自定义泛型接口-语法

```
interface 接口名 <T, R...> {
}
```

4、自定义泛型接口-细节

```
1) 接口中，静态成员不能使用泛型（接口默认是静态）
2) 泛型接口的类型，在继承接口或者实现接口时确定
3) 没有指定类型时，默认为Object
```

5、自定义泛型方法-语法

```
修饰符 <T,R...> 返回类型 方法名(参数列表) {
}
```

6、自定义泛型方法-细节

```
1) 泛型方法，可以定义在普通类中，也可以定义在泛型类中
2) 当泛型方法被调用时，类型会确定
3) public void eat(E e) {}，修饰符后没有<T,R...> eat，那么方法不是泛型方法，而是使用了泛型
```

四、继承和通配符

1、特点

```
1) 泛型不具备继承性
2) <?>：支持任意泛型类型
3) <? extends A>：支持A类以及A类的子类，用于规定泛型的上限
4) <? super A>：支持A类以及A类的父类，不限于直接父类，用于规定泛型的下限
```

2、JUnit-简介

```
JUnit是一个Java语言的单元测试框架，方便了多个功能代码测试。多数Java的开发环境都集成了JUnit作为单元测试的工具。
```