

泛型

泛型

- 定义类、接口、方法时，同时声明了一个或者多个类型变量（如：<E>），称为泛型类、泛型接口、泛型方法、它们统称为泛型。

```
public class ArrayList<E>{  
    ...  
}
```

- 作用：泛型提供了在编译阶段约束所能操作的数据类型，并自动进行检查的能力！这样可以避免强制类型转换，及其可能出现的异常。
- 泛型的本质：把具体的数据类型作为参数传给类型变量。

自定义泛型类 自定义泛型接口 自定义泛型方法

一、泛型类

泛型类

```
修饰符 class 类名<类型变量, 类型变量, ...> {  
  
}
```

```
public class ArrayList<E>{  
    ...  
}
```

- 注意：类型变量建议用大写的英文字母，常用的有：E、T、K、V 等

```
public class w {  
  
    public static void main(String[] args) {  
        MyArrayList<String> l=new MyArrayList<>();  
        l.add("Java1");  
        l.add("Java2");  
        l.add("Java3");  
  
        system.out.println(l.get(2));  
  
    }  
}
```

```

public class MyArrayList<E> {
    private Object[] arr=new Object[10];
    private int size=0;
    public boolean add(E e){
        arr[size++]=e;
        return true;
    }

    public E get (int index){
        return (E)arr[index];
    }
}

```

二、泛型接口

泛型接口

修饰符 **interface** 接口名<类型变量, 类型变量, ...> {
 }

```

public interface A<E>{
    ...
}

```

- 注意：类型变量建议用大写的英文字母，常用的有：**E、T、K、V** 等

```

public interface Data<E> {

    void add (E e);

    ArrayList<E> get(String name);
}

```

```

public class MyArrayList<E> implements Data<E>{

    @Override
    public void add(E e) {
        //具体实现
    }

    @Override
    public ArrayList<E> get(String name) {
        //具体实现
    }
}

```

```

        return null;
    }

}

```


三、泛型方法

修饰符 <类型变量, 类型变量, ...> 返回值类型 方法名(形参列表) {
}

```
public static <T> void test(T t){
}
```



```
public E get(int index){
    return (E) arr[index];
}
```



通配符

- 就是 “?”，可以在“使用泛型”的时候代表一切类型；E T K V 是在定义泛型的时候使用。

泛型的上下限：

- 泛型上限：? **extends** Car：? 能接收的必须是Car或者其子类。
- 泛型下限：? **super** Car：? 能接收的必须是Car或者其父类。

```

class car{

}

class c1 extends car{

}

class c2 extends car{

}

public class w {

    public static void main(String[] args) {
        ArrayList<car> ca=new ArrayList<>();
        ca.add(new c1());
        ca.add(new c2());
        go(ca);

        ArrayList<c1> cr=new ArrayList<>();
        ca.add(new c1());
        ca.add(new c1());
        go(cr);
    }
}

```

```
public static <T> T test(T t){  
    return t;  
}  
  
public static <T extends car> void go(ArrayList<T> cars){  
  
}  
  
}
```