Java 核心技术(高阶)

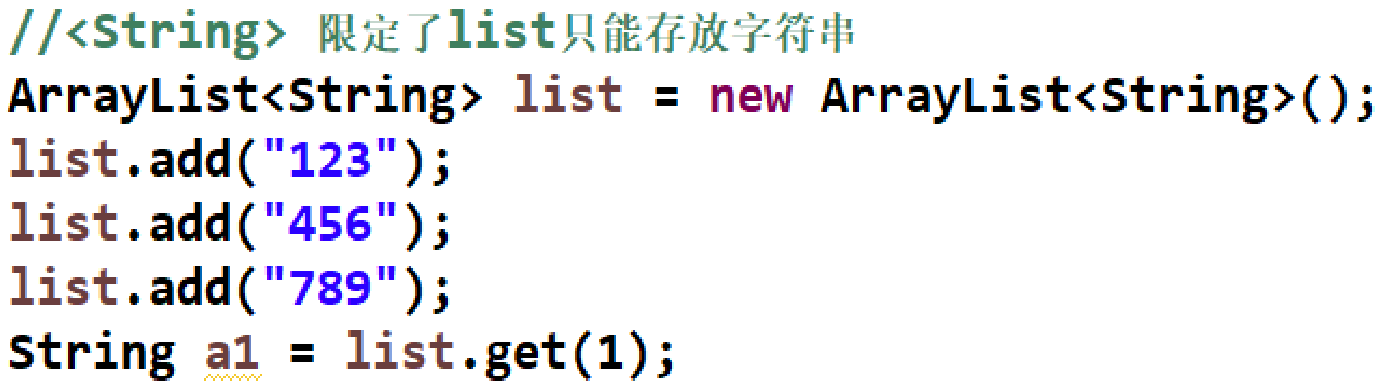
第二章 Java泛型

第二节 自定义泛型设计

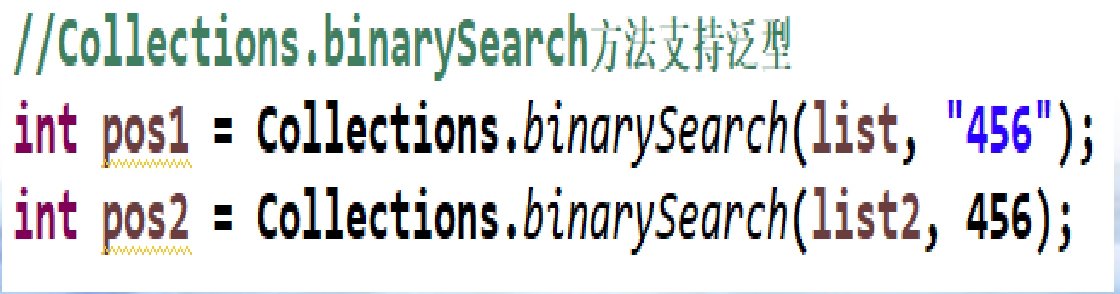
华东师范大学 陈良育

泛型

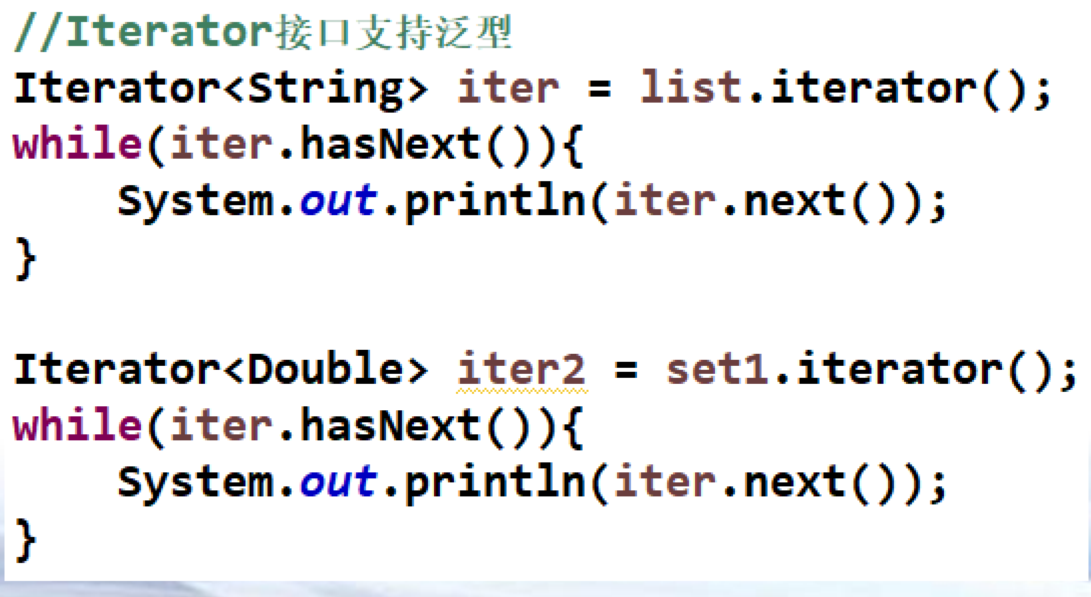
- 泛型：编写的代码可以被很多不同类型的对象所重用
  - 泛型类：ArrayList，HashSet，HashMap等
  - 泛型方法：Collections.binarySearch，Arrays.sort 等
  - 泛型接口：List, Iterator 等



```
//<String> 限定了list只能存放字符串
ArrayList<String> list = new ArrayList<String>();
list.add("123");
list.add("456");
list.add("789");
String a1 = list.get(1);
```



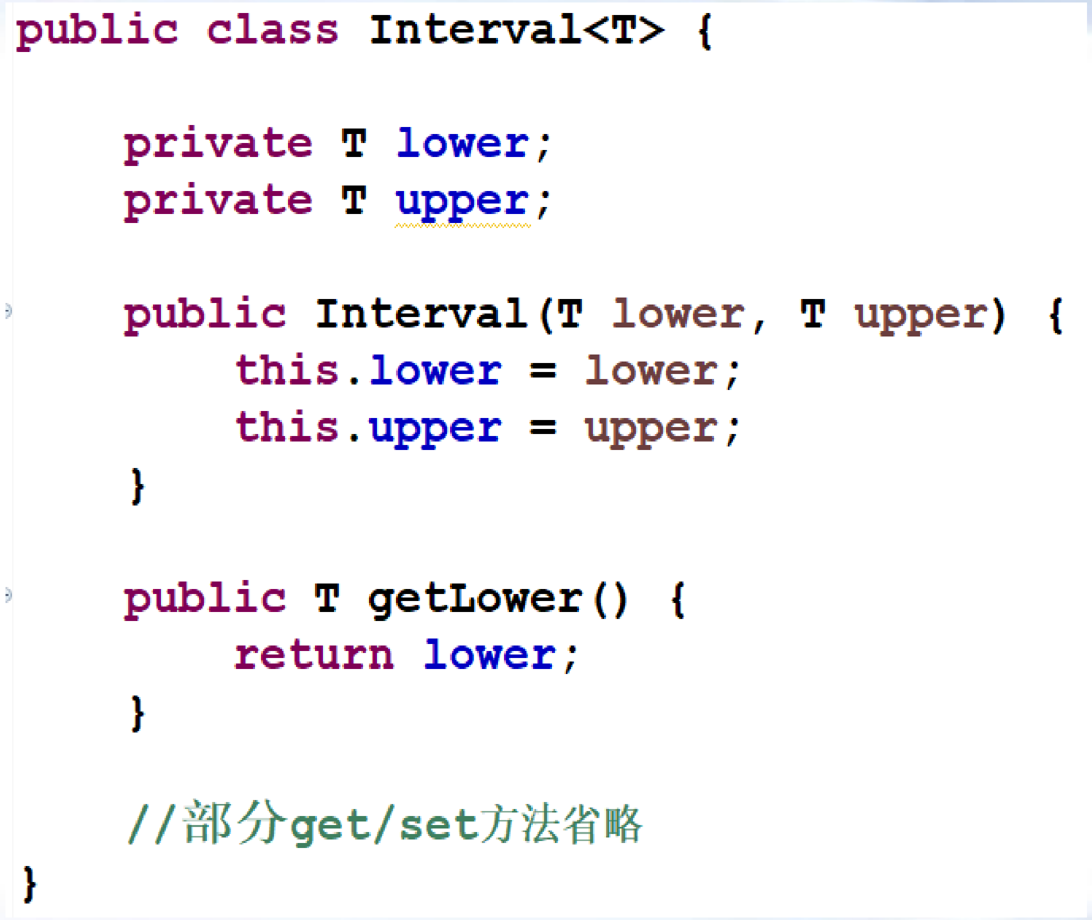
```
//Collections.binarySearch方法支持泛型
int pos1 = Collections.binarySearch(list, "456");
int pos2 = Collections.binarySearch(list2, 456);
```



```
//Iterator接口支持泛型
Iterator<String> iter = list.iterator();
while(iter.hasNext()){
    System.out.println(iter.next());
}

Iterator<Double> iter2 = set1.iterator();
while(iter.hasNext()){
    System.out.println(iter.next());
}
```

自定义泛型(1)

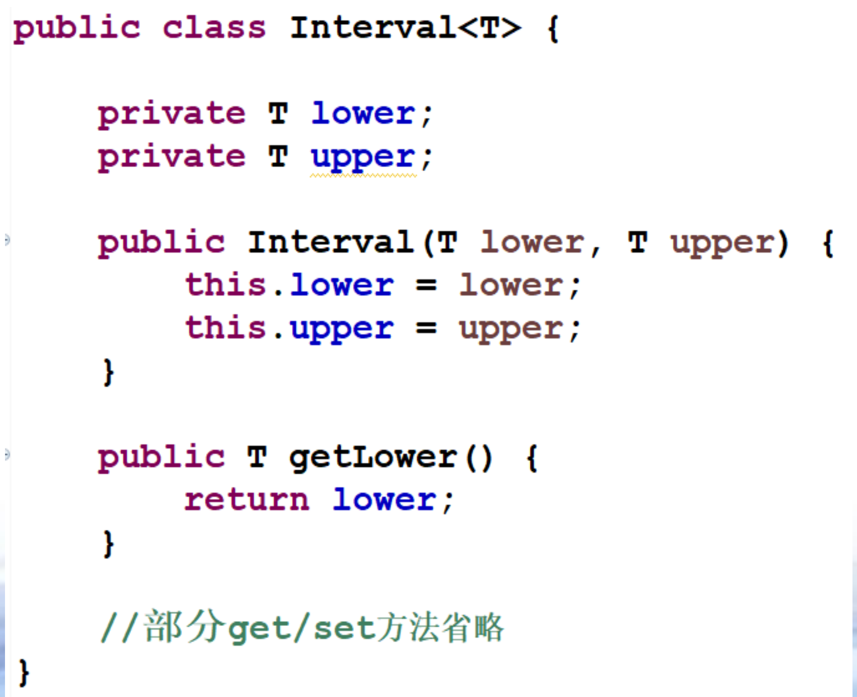
- 泛型类
  - 具有泛型变量的类
  - 在类名后用<T>代表引入类型
    - 多个字母表示多个引入类型如<T, U>等
    - 引入类型可以修饰成员变量/局部变量/参数/返回值
    - 没有专门的template关键字



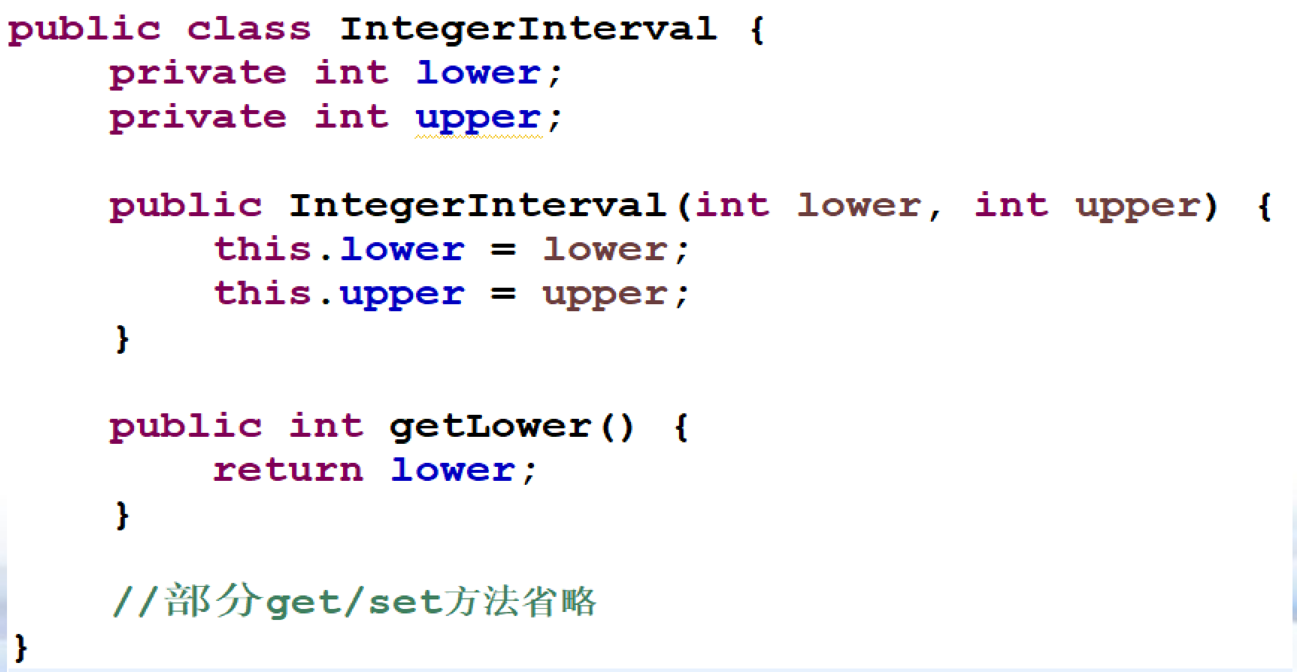
```java
public class Interval<T> {

    private T lower;
    private T upper;

    public Interval(T lower, T upper) {
        this.lower = lower;
        this.upper = upper;
    }

    public T getLower() {
        return lower;
    }

    //部分get/set方法省略
}
```

自定义泛型(2)

- 泛型类调用
  - 传入具体的类
    - Interval<Integer> v1 = new Interval<Integer>(1,2);
    - Interval<Integer> v1 = new Interval<>(1,2);



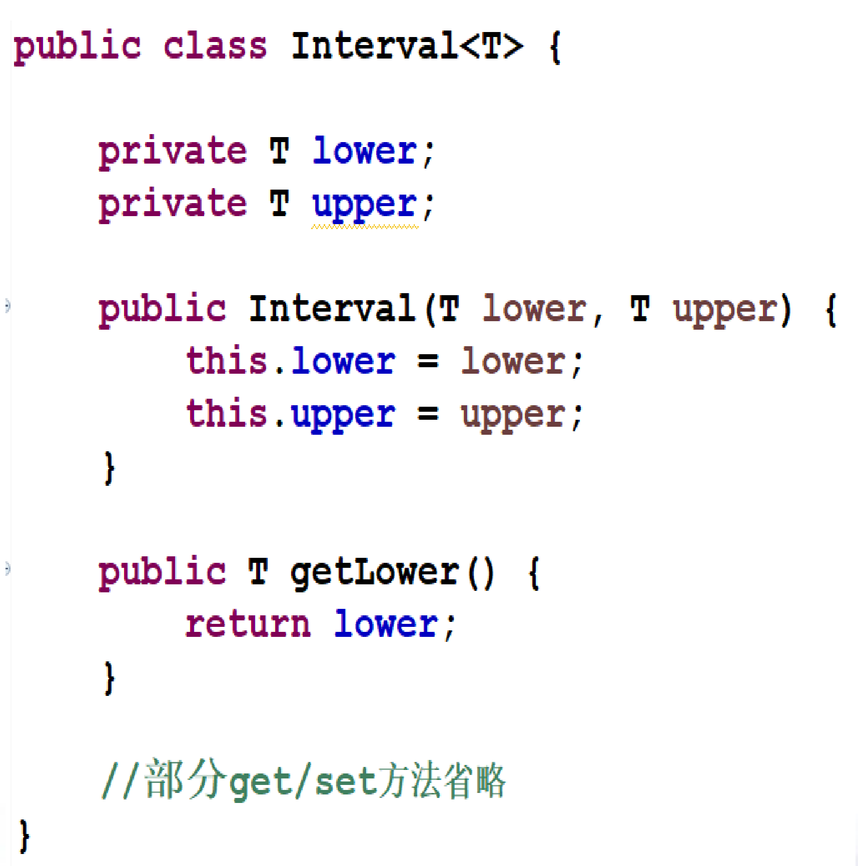
```java
public class Interval<T> {

    private T lower;
    private T upper;

    public Interval(T lower, T upper) {
        this.lower = lower;
        this.upper = upper;
    }

    public T getLower() {
        return lower;
    }

    //部分get/set方法省略
}
```



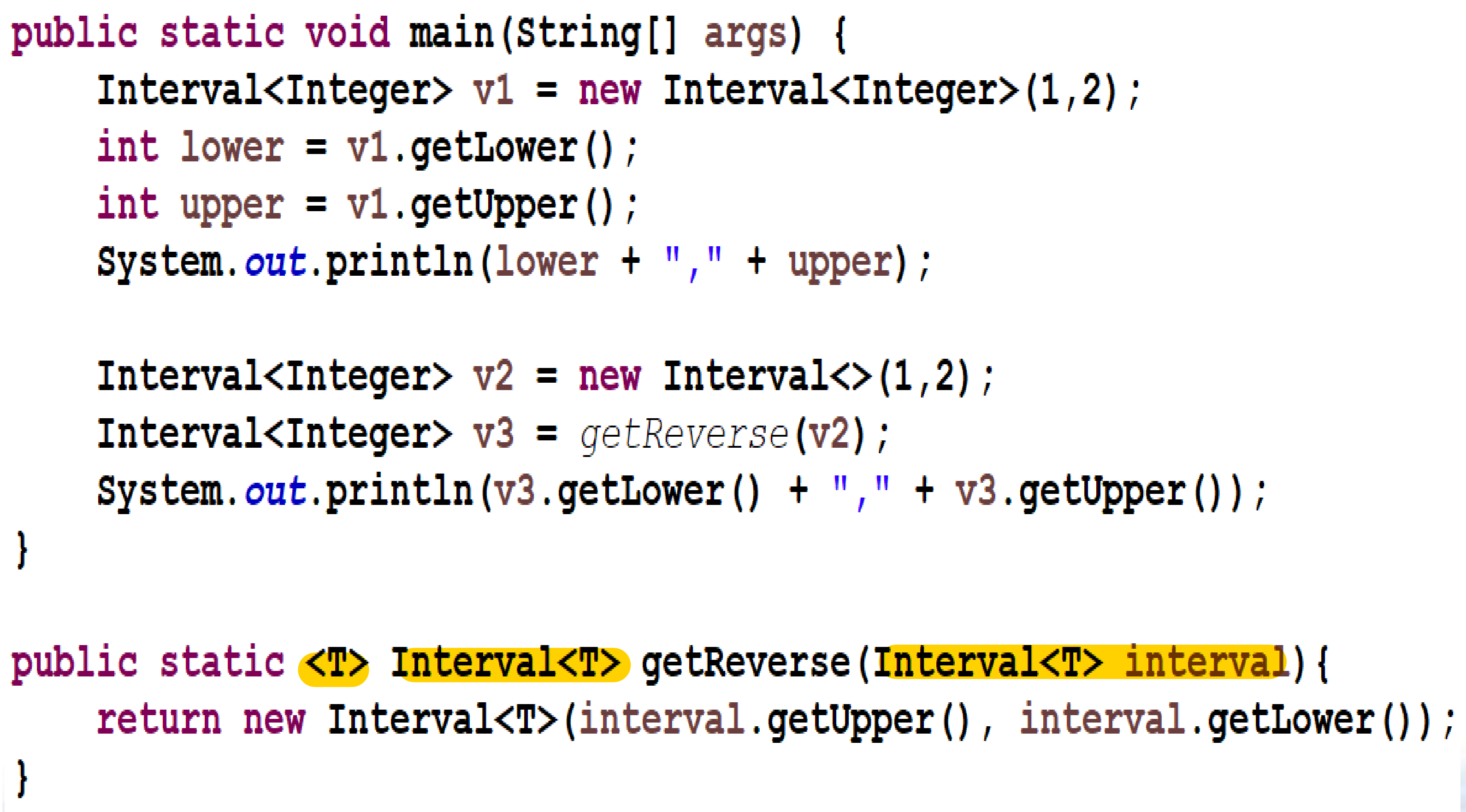
```java
public class IntegerInterval {
    private int lower;
    private int upper;

    public IntegerInterval(int lower, int upper) {
        this.lower = lower;
        this.upper = upper;
    }

    public int getLower() {
        return lower;
    }

    //部分get/set方法省略
}
```

自定义泛型(3)

- 泛型类调用



```java
public class Interval<T> {

    private T lower;
    private T upper;

    public Interval(T lower, T upper) {
        this.lower = lower;
        this.upper = upper;
    }

    public T getLower() {
        return lower;
    }

    //部分get/set方法省略
}
```



```java
public static void main(String[] args) {
    Interval<Integer> v1 = new Interval<Integer>(1,2);
    int lower = v1.getLower();
    int upper = v1.getUpper();
    System.out.println(lower + "," + upper);

    Interval<Integer> v2 = new Interval<>(1,2);
    Interval<Integer> v3 = getReverse(v2);
    System.out.println(v3.getLower() + "," + v3.getUpper());
}

public static <T> Interval<T> getReverse(Interval<T> interval){
    return new Interval<T>(interval.getUpper(), interval.getLower());
}
```

自定义泛型(4)

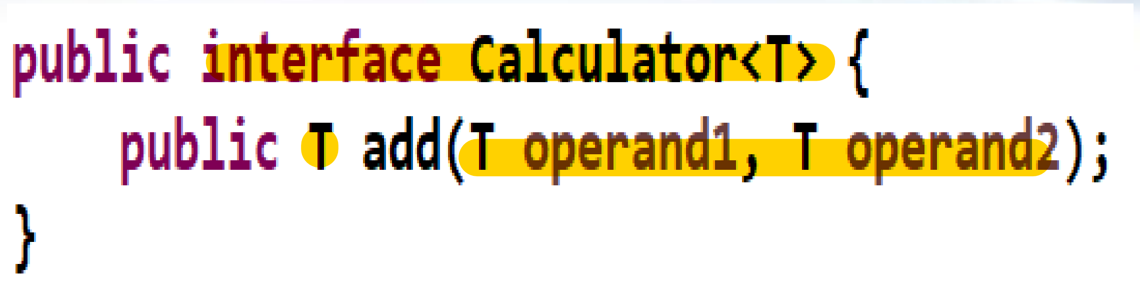
- 泛型方法
  - 具有泛型参数的方法
  - 该方法可在普通类/泛型类中
  - <T>在修饰符后，返回类型前

```java
public class ArrayUtil {
    public static <T> T getMiddle(T... a)
    {
        return a[a.length/2];
    }
}
```

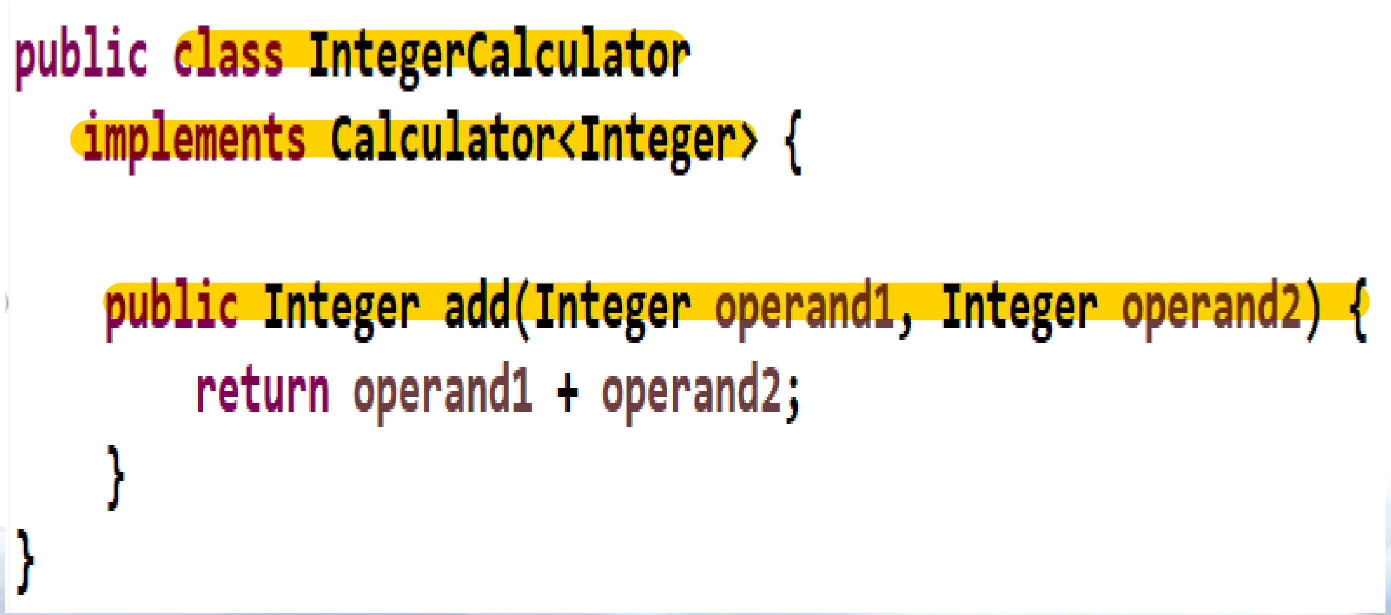
```java
String s1 = ArrayUtil.<String>getMiddle("abc", "def", "ghi");
Integer i1 = ArrayUtil.getMiddle(1,2,3);

//null is ok
String s2 = ArrayUtil.<String>getMiddle("abc", "def", null);

//error 寻找共同超类，再转型
Integer i2 = ArrayUtil.getMiddle(1,2.5f,3L);
```

自定义泛型(5)

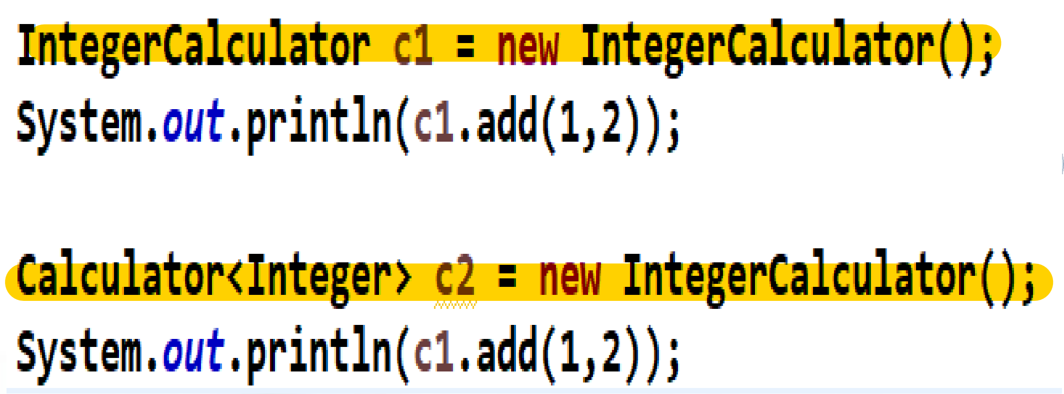
- 泛型接口
  - 和泛型类相似，在类名后加<T>
  - T用来指定方法返回值和参数
  - 实现接口时，指定类型



```java
public interface Calculator<T> {
    public T add(T operand1, T operand2);
}
```



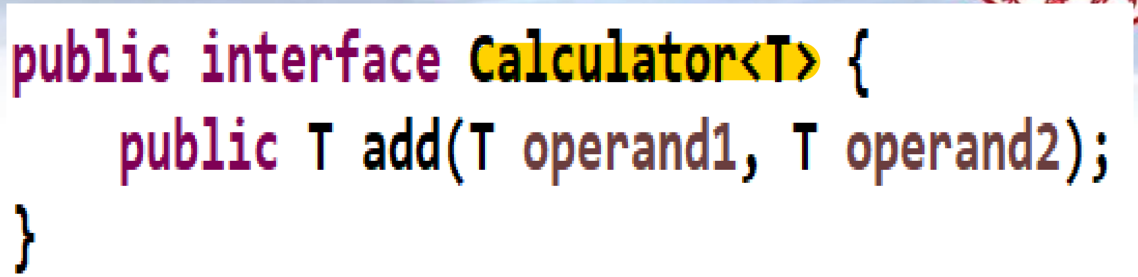
```java
public class IntegerCalculator
  implements Calculator<Integer> {

    public Integer add(Integer operand1, Integer operand2) {
        return operand1 + operand2;
    }
}
```



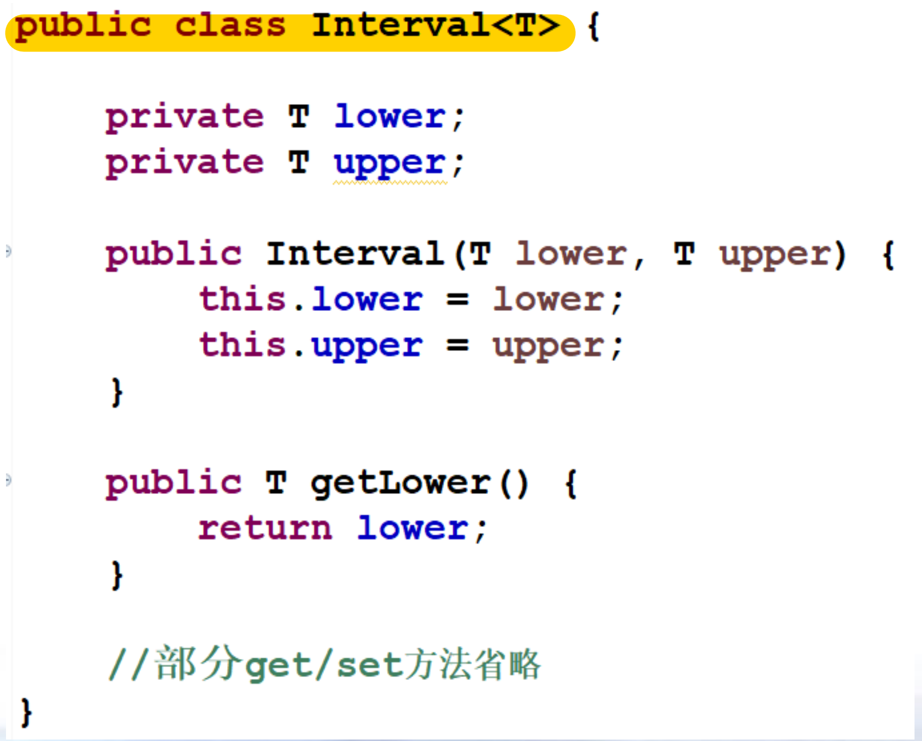
```java
IntegerCalculator c1 = new IntegerCalculator();
System.out.println(c1.add(1,2));

Calculator<Integer> c2 = new IntegerCalculator();
System.out.println(c1.add(1,2));
```

自定义泛型(6)

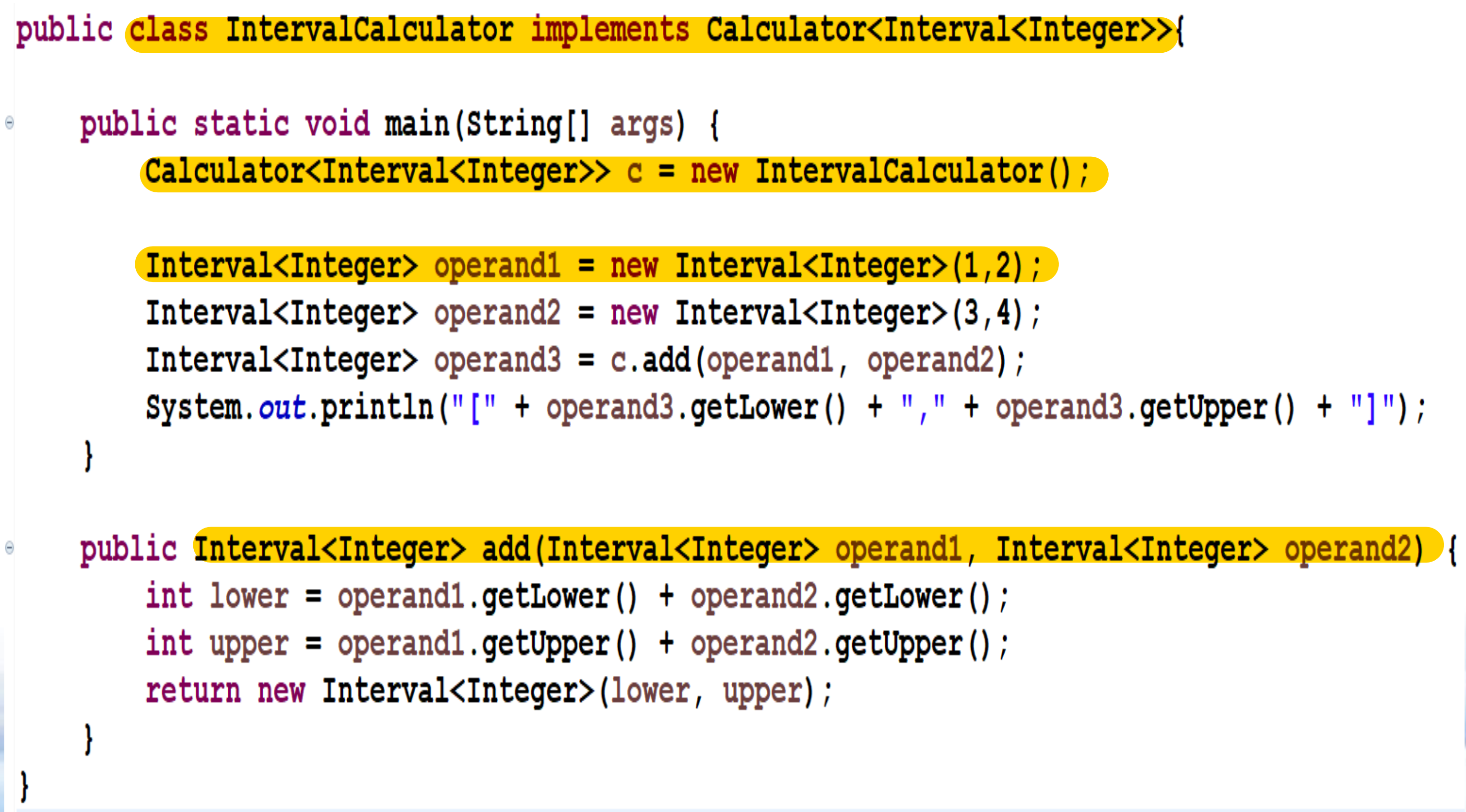
- 泛型接口
  - T也可以再是一个泛型类



```java
public interface Calculator<T> {
    public T add(T operand1, T operand2);
}
```



```java
public class Interval<T> {

    private T lower;
    private T upper;

    public Interval(T lower, T upper) {
        this.lower = lower;
        this.upper = upper;
    }

    public T getLower() {
        return lower;
    }

    //部分get/set方法省略
}
```



```java
public class IntervalCalculator implements Calculator<Interval<Integer>>{

    public static void main(String[] args) {
        Calculator<Interval<Integer>> c = new IntervalCalculator();

        Interval<Integer> operand1 = new Interval<Integer>(1,2);
        Interval<Integer> operand2 = new Interval<Integer>(3,4);
        Interval<Integer> operand3 = c.add(operand1, operand2);
        System.out.println("[" + operand3.getLower() + "," + operand3.getUpper() + "]");
    }

    public Interval<Integer> add(Interval<Integer> operand1, Interval<Integer> operand2) {
        int lower = operand1.getLower() + operand2.getLower();
        int upper = operand1.getUpper() + operand2.getUpper();
        return new Interval<Integer>(lower, upper);
    }
}
```

总结

- 自定义泛型设计
  - 泛型类：整个类都被泛化，包括变量和方法
  - 泛型方法：方法被泛化，包括返回值和参数
  - 泛型接口：泛化子类方法

谢 谢！