泛型：实现了参数化类型的概念，使代码应用于多种类型。

泛型方法

TestMethod.java

```
public class TestMethod {
  public static <J> void printValue(J inputValue) {//泛型方法：类型参数声明<J>在方法返回类型void之前
    System.out.println(inputValue);
  }

  public static void main(String []args) {
    int a = 10;
    char b = 't';
    double c = 2.22;
    System.out.println("----int----");
    printValue(a);
    System.out.println("----char----");
    printValue(b);
    System.out.println("----double----");
    printValue(c);
  }

}
```

运行结果如下

```
----int----
10
----char----
t
----double----
2.22
```

泛型类

TestClass.java

```
public class TestClass<J> {//以下J的类型为由外部指定，如Integer,String
  J j;

  public void setValue(J j) {
    this.j = j;
  }

  public J getValue() {//返回值类型为J
    return j;
  }

  public static void main(String []args) {
    //TestClass<int> a = new TestClass<int>();//会出错，泛型的类型只能是类类型，不能是简单类型
    TestClass<Integer> a = new TestClass<Integer>();
```

```
        TestClass<String> b = new TestClass<String>();

        a.setValue(1);
        b.setValue("test generic");

        System.out.println("int = "+a.getValue());
        System.out.println("char = "+b.getValue());
    }
}
```

运行结果如下:

```
int = 1
char = test generic
```

通配符

TestGeneric.java

```
import java.util.*;
public class TestGeneric {
    public static void printValue(List<?> inputValue) {
        System.out.println(inputValue.get(0));
    }

    public static void getNumber(List<? extends Number> number) {//上边界限定通界符
        System.out.println(number.get(0));
    }

    public static void getSuper(List<? super String> superValue) {//下边界限定通界符:
        System.out.println(superValue.get(0));
    }

    public static void main(String []args) {
        List<Integer> a = new ArrayList<Integer>();
        List<String> b = new ArrayList<String>();
        a.add(10);
        b.add("test generic");
        System.out.println("-----<?>--------");
        printValue(a);
        printValue(b);
        System.out.println("-----<? extends T>--------");
        getNumber(a);//Integer属于Number类所以来编译通过
        //getNumber(b);//无法编译通过。因为b是String类。
        System.out.println("-----<? super T>--------");
        //getSuper(a);//无法编译通过，因为已经<? super String>限定为String类了。

        getSuper(b);//和上面一样，无法编译过。
    }
}
```

运行结果如下:

```
-----<?>--------
10
test generic
-----<? extends T>--------
10
-----<? super T>--------
test generic
```