1. Java中的泛型

和C++中类似，分为泛型方法和泛型类

2. 泛型方法

就是在函数定义的前面加<T> 或 <T, E>，然后在函数中就可以使用了。

调用方法时，可以调用就可以了。

class Test {  
 private static <T> void funcTest(T t1, T t2){  
 if (t1.equals(t2)) {  
 System.*out*.println("equal");  
 }  
 }  
 public static void main(String[] args) {  
 *funcTest*(1, 1); // OK  
 }  
}

注意的是：

(1) T只能代表对象的引用，不可是基本类型。但这个基本没用，传入基本类型也可以隐式转换为对应的对象。

public class Main {  
 private static <T> void funcTest(T t1, T t2){  
 if (t1.equals(t2)) {  
 System.*out*.println("equal");  
 }  
 }  
 public static void main(String[] args) {  
 *funcTest*(1, 1); // OK  
 }  
}

(2) 可以对T进行一些限制。比如要求T是A的子类、要求T实现了B的接口等

public class Main {  
 private static <T extends Test> void funcTest(T t1, T t2){  
 if (t1.equals(t2)) {  
 System.*out*.println("equal");  
 }  
 }  
}

3. 泛型类

在定义类名的后面加 <T>或<T, E>，然后在类中就可以使用了。

使用时，定义泛型类时需要制定具体类型。

class Test <T>{  
 private T t;  
 public void func(T t){  
 System.*out*.println(t);  
 }  
 public static void f(T t){// ERROR  
  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 //Test<int> test = new Test<int>(); Error  
 Test<Integer> integerTest = new Test<Integer>();  
 integerTest.func(1);  
 }  
}

注意：

(1) 同样T只能代表对象的引用，不可是基本类型。

(2) 同样可以对T进行一些限制。比如要求T是A的子类、要求T实现了B的接口等

(3) 泛型类中的static方法中不可使用泛型。

4. 通配符?

对函数的形参，可以使用通配符来说明，函数可以接受任意类型。

public class Main {  
 private static void funcTest(List<?> list) {  
 //list.add(1); ERROR   
 System.*out*.println(list);  
 }  
 private static void func(? list) { //ERROR  
 System.*out*.println(list);  
 }  
 public static void main(String[] args) {  
 List<Integer> ints = new ArrayList<>();  
 ints.add(1);  
 *funcTest*(ints);  
 }  
}

注意：

(1) 同样?只能代表对象的引用，不可是基本类型。

(2) 同样可以对?进行一些限制。比如要求T是A的子类、要求T实现了B的接口等

(3) ？不能单独使用。

(4) 通配符中的参数是只读的。

5. 泛型方法和通配符的比较

泛型方法中，参数是可改变的。通配符中的参数是只读的。