泛型的意义

在JDK1.5 之前，所有集合框架中的对象都会被转成最上层的Object类型，取出时再强制转换成指定类型，这样做会导致代码臃肿，而且容易引起ClassCastException异常，所以泛型的意义是**规定集合中可以容纳的元素类型**

好处

1. 将运行时期才会出现的问题(ClassCastException)转移至编译时期(只要在编译时期没有出现警告，那么运行时期就不会出现ClassCastException异常)
2. 避免了强制转换的操作麻烦
3. 提高程序类型的安全性

泛型语法

在jdk7之前，调用构造函数创建对象时也要指定类型，但是JDK8之后省略了这一点

```
ArrayList<String> userList = new ArrayList<>();
```

ArrayList<String>为**参数化类型**

- 把类型当作是参数一样传递
- <数据类型> 只能是引用类型

泛型类/泛型方法

泛型类

泛型类就是把泛型定义在类上，用户使用该类的时候，才把类型明确下来

- 为什么要定义泛型类
  为了完成**类型的扩展**
- 什么时候定义泛型类
  当类中要操作的**引用数据类型**不确定的时候，早期定义Object来完成扩展。现在定义泛型来完成扩展。
- **在类上定义的泛型，在类的方法中也可以使用**

```
//定义泛型类，对整个类都有效
class Utils<T>
```

```
{
    private T q;
    public void setObject(T q)
    {
        this.q = q;
    }
    public T getObject()
    {
        return q;
    }
}

//也可以将泛型定义在方法上，为了避免泛型类中定义的泛型对整个类都有效的特点
class  GenericDemo
{
    public static void main(String[] args)
    {

        Utils<Worker> u = new Utils<Worker>();

        u.setObject(new Worker());
        Worker w = u.getObject();;

    }
}
```

类型通配符

为什么要引入类型通配符

为了处理传入的形参类型不确定的情况

```
public void test(List<Object> c){
    for (int i = 0 ; i < c.size() ; i++){
        System.out.println(c.get(i));
    }
}
```

如果这样写，传入实际参数值时可能会出现问题

```
//创建一个List<String>对象
List<String> strList = new ArrayList<>;
//将strList作为参数来调用前面的test方法
test(strList) ; //这里会发生编译错误
```

需要特别注意：泛型中的类型并不像以前一样有着继承关系，List<String>对象不能被当成List<Object>使用，也就是说，**List<String>类并不是List<Object>类的子类**，并且他们是毫无关系的！！

类型通配符

为了表示各种泛型List的父类，可以使用类型通配符(?)，**将一个问号作为类型实参传给List集合**，写作：List<?>，意思是元素类型未知的List。这个?被称为**通配符**， **它的元素类型可以匹配任何类型**。可以将上面方法改写为如下形式：

```
public void test(List<?> c){
    for (int i = 0 ; i < c.size() ; i++){
        System.out.println(c.get(i));
    }
}
```

- **非常值得注意的是**，当我们使用?号通配符的时候：**就只能调用对象与类型无关的方法，不能调用对象与类型有关的方法**。因为直到外界使用时才知道具体的类型是什么。也就是说，在上面的List集合，是不能使用add()方法的。

限制的泛型通配符

- 指定上限：只有继承了Shape类的子类对象才能被传入drawAll方法
  指定下限同理，使用super关键字

```
public class Canvas
//同时在画布上绘制多个形状，使用被限制的泛型通配符
public void drawAll(List<? extends Shape> shapes)
{
for (Shape s : shapes)
{
s.draw(this);
```

关于类型通配符(?)和类型形参T的区别
聊一聊-JAVA 泛型中的通配符 T，E，K，V，？
T 是一个 **确定的** 类型，通常用于泛型类和泛型方法的定义，？是一个 **不确定** 的类型，通常用于泛型方法的调用代码和形参，**不能用于定义类和泛型方法**。

如何选择通配符？和泛型方法<T>

- 如果**参数之间的类型有依赖关系**，或者**返回值是与参数之间有依赖关系**的。那么就使用泛型方法
- 如果没有依赖关系的，就使用通配符，通配符会灵活一些.

泛型擦除

泛型是提供给**javac编译器**使用的，它用于限定集合的输入类型，让编译器在源代码级别上，即挡住向集合中插入非法数据。但编译器编译完带有泛形的java程序后，**生成的class文件中将不再带有泛形信息**，以此使程序运行效率不受到影响，这个过程称之为“擦除”。