**为了定义一个类为泛型类型，需要将泛型类型放在类名之后，例如，GernericStack<E>。**

**为了定义一个方法为泛型类型，需要将泛型类型放在方法返回类型之前，例如，<E>void max(E o1,E o2)**

**可以使用非受限通配、受限通配或者下限通配来对一个泛型类型指定范围。**

**不能使用泛型类型参数来创建实例。如：E object=new E();错误**

**不能使用泛型类型参数创建数组。如：E[] elements = new E[capacity];错误**

**可以E[] elements=(E[])new Object[capacity];**

**不能使用泛型类创建泛型数组，如：**

**ArrayList<String>[] list =new ArrayList<String>[10];错误**

**可以ArrayList<String>[] list=(ArrayList<String>[])new ArrayList[10]**

**泛型类是被它的所有实例所共享的。**

**在静态上下文中不允许类的参数是泛型类型，在泛型类中，其静态方法或静态数据域会被所有实例共享，实例的类型会不一样，例如，String、Integer，不符合泛型类的规定。例如，**

**public class Test<E>{**

**public static void m(E o1){ // Illegal**

**}**

**public static E o1; // Illegal**

**static {**

**E o2; // Illegal**

**}**

**}**

**异常类不能是泛型**

**小结**

**1.泛型具有参数化类型的能力。可以定义使用泛型类型的类或方法，编译器会用具体类型来替换泛型类型。**

**2.泛型的主要优势是能够在编译时而不是在运行时检测错误。**

**3.泛型类或方法允许指定这个类或方法可以带有的对象类型。如果试图使用带有不兼容对象的类或方法，编译器会检测出这个错误。**

**4.定义在类、接口或者静态方法中的泛型称为形式泛型类型，随后可以用一个实际具体类型来替换它。替换泛型类型的过程称为泛型实例化。**

**5.不使用类型参数的泛型类称为原始类型，例如ArrayList。使用原始类型是为了向后兼容java较早的版本。**

**6.通配泛型类型有三种形式：？、？ extends T 和？ super T，这里T代表一个泛型类型。第一种形式？称为非受限通配，他和？ extends Object是一样的。第二种形式？ extends T称为受限通配，代表T或者T的一个子类型。第三种类型？ super T称为下限通配，表示T或者T的一个父类型。**

**7.使用称为类型消除（类型擦除）的方法来实现泛型。编译器使用泛型类型信息来编译代码，但是随后消除它。因此，泛型信息在运行时是不可用的。这个方法能够使泛型代码向后兼容使用原始类型的遗留代码。**

**8.不能使用泛型类型参数来创建实例。**

**9.不能使用泛型类型参数来创建数组。**

**10.不能在静态环境中使用泛型类型参数。**

**11.在异常类中不能使用泛型类型参数。**