```
13      RationalMatrix rationalMatrix = new RationalMatrix();
14
15      System.out.println("\nm1 + m2 is ");
16      GenericMatrix.printResult(
17        m1, m2, rationalMatrix.addMatrix(m1, m2), '+');
18
19      System.out.println("\nm1 * m2 is ");
20      GenericMatrix.printResult(
21        m1, m2, rationalMatrix.multiplyMatrix(m1, m2), '*');
22    }
23  }
```

```
m1 + m2 is
 1/5 1/6 1/7     1/6 1/7 1/8     11/30 13/42 15/56
 2/5 1/3 2/7  +  1/3 2/7 1/4  =  11/15 13/21 15/28
 3/5 1/2 3/7     1/2 3/7 3/8     11/10 13/14 45/56

m1 * m2 is
 1/5 1/6 1/7     1/6 1/7 1/8     101/630 101/735 101/840
 2/5 1/3 2/7  *  1/3 2/7 1/4  =  101/315 202/735 101/420
 3/5 1/2 3/7     1/2 3/7 3/8     101/210 101/245 101/280
```

复习题

19.23　为什么 GenericMatrix 类中的 add、multiple 以及 zero 方法定义为抽象的？

19.24　IntegerMatrix 类中 add、multiple 以及 zero 方法是如何实现的？

19.25　RationalMatrix 类中 add、multiple 以及 zero 方法是如何实现的？

19.26　如果 printResult 方法如下定义，将会报什么错？

```
public static void printResult(
  E[][] m1, E[][] m2, E[][] m3, char op)
```

关键术语

actual concrete type（实际具体类型）
bounded generic type（受限泛型类型）
bounded wildcard(<? extends E>)（受限通配）
formal generic type（形式泛型类型）
generic instantiation（泛型实例化）
lower-bound wildcard(<? super E>)（下限通配）
raw type（原始类型）
unbounded wildcard(<?>)（非受限通配）

本章小结

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，这里的 T 代表一个泛型类型。第一种形式 ? 称为非受限通配，它和 ? 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类的泛型类型参数。
11. 在异常类中不能使用泛型类型参数。

测试题

回答位于网址 www.cs.armstrong.edu/liang/intro10e/quiz.html 的本章测试题。

编程练习题

19.1 （修改程序清单 19-1）修改程序清单 19-1 中的 GenericStack 类，使用数组而不是 ArrayList 来实现它。你应该在给栈添加新元素之前检查数组的大小。如果数组满了，就创建一个新数组，该数组是当前数组大小的两倍，然后将当前数组的元素复制到新数组中。

19.2 （使用继承实现 GenericStack）程序清单 19-1 中，GenericStack 是使用组合实现的。定义一个新的继承自 ArrayList 的栈类。画出 UML 类图，然后实现 GenericStack。编写一个测试程序，提示用户输入 5 个字符串，然后以逆序显示它们。

19.3 （ArrayList 中的不同元素）编写以下方法，返回一个新的 ArrayList。新的列表中包含来自原列表中的不重复元素。

```
public static <E> ArrayList<E> removeDuplicates(ArrayList<E> list)
```

19.4 （泛型线性搜索）为线性搜索实现以下泛型方法。

```
public static <E extends Comparable<E>>
  int linearSearch(E[] list, E key)
```

19.5 （数组中的最大元素）实现下面的方法，返回数组中的最大元素。

```
public static <E extends Comparable<E>> E max(E[] list)
```

19.6 （二维数组中的最大元素）编写一个泛型方法，返回二维数组中的最大元素。

```
public static <E extends Comparable<E>> E max(E[][] list)
```

19.7 （泛型二分查找法）使用二分查找法实现下面的方法。

```
public static <E extends Comparable<E>>
  int binarySearch(E[] list, E key)
```

19.8 （打乱 ArrayList）编写以下方法，打乱 ArrayList。

```
public static <E> void shuffle(ArrayList<E> list)
```

19.9 （对 ArrayList 排序）编写以下方法，对 ArrayList 排序。

```
public static <E extends Comparable<E>>
  void sort(ArrayList<E> list)
```

19.10 （ArrayList 中的最大元素）编写以下方法，返回 ArrayList 中的最大元素。

```
public static <E extends Comparable<E>> E max(ArrayList<E> list)
```

19.11 （ComplexMatrix）使用编程练习题 13.17 中所介绍的 Complex 类来开发 ComplexMatrix 类，用于执行涉及复数的矩阵运算。ComplexMatrix 类继承自 GenericMatrix 类并实现 add、multiple 以及 zero 方法。你需要修改 GenericMatrix 并将每个出现的 Number 替换为 Object，因为 Complex 不是 Number 的子类。编写一个测试程序，创建两个矩阵并且调用 printResult 方法显示它们相加和相乘的结果。