21 Generic Classes and Methods

Every man of genius sees the world at a different angle from his fellows.
—Havelock Ellis

…our special individuality, as distinguished from our generic humanity.
—Oliver Wendell Holmes, Sr.

Born under one law, to another bound.
—Lord Brooke

Objectives

In this chapter you'll learn:

- To create generic methods that perform identical tasks on arguments of different types.
- To create a generic `Stack` class that can be used to store objects of any class or interface type.
- To understand how to overload generic methods with nongeneric methods or with other generic methods.
- To understand raw types and how they help achieve backward compatibility.
- To use wildcards when precise type information about a parameter is not required in the method body.

Self-Review Exercises

21.1 State whether each of the following is *true* or *false*. If *false*, explain why.

a) A generic method cannot have the same method name as a nongeneric method.

ANS: False. Generic and nongeneric methods can have the same method name. A generic method can overload another generic method with the same method name but different method parameters. A generic method also can be overloaded by providing nongeneric methods with the same method name and number of arguments.

b) All generic method declarations have a type-parameter section that immediately precedes the method name.

ANS: False. All generic method declarations have a type-parameter section that immediately precedes the method's return type.

c) A generic method can be overloaded by another generic method with the same method name but different method parameters.

ANS: True.

d) A type parameter can be declared only once in the type-parameter section but can appear more than once in the method's parameter list.

ANS: True.

e) Type-parameter names among different generic methods must be unique.

ANS: False. Type-parameter names among different generic methods need not be unique.

f) The scope of a generic class's type parameter is the entire class except its `static` members.

ANS: True.

21.2 Fill in the blanks in each of the following:

a) ________ and ________ enable you to specify, with a single method declaration, a set of related methods, or with a single class declaration, a set of related types, respectively.

ANS: Generic methods, generic classes.

b) A type-parameter section is delimited by ________.

ANS: angle brackets (< and >).

c) A generic method's ________ can be used to specify the method's argument types, to specify the method's return type and to declare variables within the method.

ANS: type parameters.

d) The statement "`Stack objectStack = new Stack();`" indicates that `objectStack` stores ________.

ANS: a raw type.

e) In a generic class declaration, the class name is followed by a(n) ________.

ANS: type-parameter section.

f) The syntax ________ specifies that the upper bound of a wildcard is type `T`.

ANS: `? extends T`.

Exercises

***NOTE: Solutions to the programming exercises are located in the* `ch21solutions` *folder. Each exercise has its own folder named* `ex21_##` *where ## is a two-digit number representing the exercise number. For example, exercise 21.8's solution is located in the folder* `ex21_08`.**

21.3 Explain the use of the following notation in a Java program:

```
public class Array< T > { }
```

ANS: This notation declares a generic class named `Array` with a single type parameter `T`.

21.4 Write a simple generic version of method `isEqualTo` that compares its two arguments with the `equals` method and returns `true` if they are equal and `false` otherwise. Use this generic method in a program that calls `isEqualTo` with a variety of built-in types, such as `Object` or `Integer`. What result do you get when you attempt to run this program?

ANS: For classes that override the `equals` method, the program will compare the objects based on their contents. For example, the `Integer` class's `equals` method would compare the contents of two `Integer` objects. For classes that do not override the `equals` method, the program will compare the objects based on their references, not their contents.

21.9 How can generic methods be overloaded?

ANS: A generic method may be overloaded in several ways. Generic methods can be overloaded by other generic methods that specify the same method name but different method parameters. Generic methods can also be overloaded by non-generic methods that have the same method name and number of parameters.

21.10 The compiler performs a matching process to determine which method to call when a method is invoked. Under what circumstances does an attempt to make a match result in a compile-time error?

ANS: If the compiler cannot match the method call made to either a non-generic method or a generic method, or if the matching process results in multiple matches with no matches more specific than the others at compile time, the compiler generates an error.

21.11 Explain why a Java program might use the statement

```
ArrayList< Employee > workerList = new ArrayList< Employee >();
```

ANS: When creating objects from a generic class, it is necessary to provide a type argument (or possibly several type arguments) to instantiate the objects with actual types. The above statement would be used to create an `ArrayList` object that stores `Employee` objects. The compiler can then perform type checking to ensure that the code uses the `ArrayList` of `Employees` properly.