**Section 20.1 Introduction**

• Generic methods enable you to specify, with one method declaration, a set of related methods.

• Generic classes and interfaces enable you to specify sets of related types.

#### Section 20.2 Motivation for Generic Methods

• Overloaded methods are often used to perform similar operations on different types of data.

• When the compiler encounters a method call, it attempts to locate a method declaration with a name and parameters that are compatible with the argument types in the method call.

#### Section 20.3 Generic Methods: Implementation and Compile-Time Translation

• If the operations performed by several overloaded methods are identical for each argument type, they can be more compactly and conveniently coded using a generic method. A single generic method declaration can be called with arguments of different data types. Based on the types of the arguments passed to a generic method, the compiler handles each method call appropriately.

• All generic method declarations have a type-parameter section (p. [843](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec3_html#page_843)) delimited by angle brackets (< and >) that precedes the method’s return type (p. [843](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec3_html#page_843)).

• A type-parameter section contains one or more type parameters separated by commas.

• A type parameter (p. [843](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec3_html#page_843)) is an identifier that specifies a generic type name. Type parameters can be used as the return type, parameter types and local variable types in a generic method declaration, and they act as placeholders for the types of the arguments passed to the generic method, which are known as actual type arguments (p. [844](http://proquest.safaribooksonline.com/9780133813036/ch20lev2sec2_html#page_844)). Type parameters can represent only reference types.

• Type-parameter names used throughout a method declaration must match those declared in the type-parameter section. A type-parameter name can be declared only once in the type-parameter section but can appear more than once in the method’s parameter list.

• When the compiler encounters a method call, it determines the argument types and attempts to locate a method with the same name and parameters that match the argument types. If there’s no such method, the compiler searches for methods with the same name and compatible parameters and for matching generic methods.

• Objects of a class that implements generic interface Comparable (Comparable) can be compared with method compareTo (p. [846](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec4_html#page_846)), which returns 0 if the objects are equal, a negative integer if the first object is less than the second or a positive integer if the first object is greater than the second.

• All the type-wrapper classes for primitive types implement Comparable.

• Comparable objects can be used with the sorting and searching methods of class Collections.

#### Section 20.4 Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type

• When a generic method is compiled, the compiler performs erasure (p. [848](http://proquest.safaribooksonline.com/9780133813036/ch20lev2sec7_html#page_848)) to remove the type-parameter section and replace the type parameters with actual types. By default each type parameter is replaced with its upper bound (p. [848](http://proquest.safaribooksonline.com/9780133813036/ch20lev2sec7_html#page_848)), which is Objectunless specified otherwise.

• When erasure is performed on a method that returns a type variable (p. [848](http://proquest.safaribooksonline.com/9780133813036/ch20lev2sec7_html#page_848)), explicit casts are inserted in front of each method call to ensure that the returned value has the type expected by the caller.

#### Section 20.5 Overloading Generic Methods

• A generic method may be overloaded with other generic methods or with nongeneric methods.

#### Section 20.6 Generic Classes

• Generic classes provide a means for describing a class in a type-independent manner. We can then instantiate type-specific objects of the generic class.

• A generic class declaration looks like a nongeneric class declaration, except that the class name is followed by a type-parameter section. The type-parameter section of a generic class can have one or more type parameters separated by commas.

• When a generic class is compiled, the compiler performs erasure on the class’s type parameters and replaces them with their upper bounds.

• Type parameters cannot be used in a class’s static declarations.

• When instantiating an object of a generic class, the types specified in angle brackets after the class name are known as type arguments (p. [851](http://proquest.safaribooksonline.com/9780133813036/ch20lev2sec8_html#page_851)). The compiler uses them to replace the type parameters so that it can perform type checking and insert cast operations as necessary.

#### Section 20.7 Raw Types

• It’s possible to instantiate a generic class without specifying a type argument. In this case, the new object of the class is said to have a raw type (p. [857](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec7_html#page_857))—the compiler implicitly uses type Object (or the type parameter’s upper bound) throughout the generic class for each type argument.

#### Section 20.8 Wildcards in Methods That Accept Type Parameters

• Class Number is the superclass of both Integer and Double.

• Number method doubleValue (p. [862](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec8_html#page_862)) obtains the Number’s underlying primitive value as a double value.

• Wildcard type arguments enable you to specify method parameters, return values, variables, and so on, that act as supertypes of parameterized types. A wildcard-type argument is denoted by ? (p. [862](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec8_html#page_862)), which represents an “unknown type.” A wildcard can also have an upper bound.

• Because a wildcard (?) is not a type-parameter name, you cannot use it as a type name throughout a method’s body.

• If a wildcard is specified without an upper bound, then only the methods of type Object can be invoked on values of the wildcard type.

• Methods that use wildcards as type arguments (p. [862](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec8_html#page_862)) cannot be used to add elements to a collection referenced by the parameter.

### Self-Review Exercises

[**20.1**](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec12_html#ch20ans1) 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.

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

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

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.

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

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

[**20.2**](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec12_html#ch20ans2) 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.

b) A type-parameter section is delimited by \_\_\_\_\_\_\_\_\_\_.

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.

d) The statement "Stack objectStack = new Stack();" indicates that objectStack stores \_\_\_\_\_\_\_\_\_\_.

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

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

### Answers to Self-Review Exercises

[**20.1**](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec11_html#ch20que1)

a) 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) False. All generic method declarations have a type-parameter section that immediately precedes the method’s return type.

c) True.

d) True.

e) False. Type-parameter names among different generic methods need not be unique.

f) True.

[**20.2**](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec11_html#ch20que2)

a) Generic methods, generic classes.

b) angle brackets (< and >).

c) type parameters.

d) a raw type.

e) type-parameter section.

f) ? extends T.

### Exercises

**20.3 (Explain Notation)** Explain the use of the following notation in a Java program:

public class Array<T> { }

**20.4 (Generic Method** ***selectionSort*)** Write a generic method selectionSort based on the sort program of [Fig. 19.4](http://proquest.safaribooksonline.com/9780133813036/ch19lev2sec12_html#ch19fig04). Write a test program that inputs, sorts and outputs an Integer array and a Float array. [Hint: Use <T extends Comparable<T>> in the type-parameter section for method selectionSort, so that you can use method compareTo to compare the objects of the type that Trepresents.]

**20.5 (Overloaded Generic Method** ***printArray*)** Overload generic method printArray of [Fig. 20.3](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec3_html#ch20fig03) so that it takes two additional integer arguments, lowSubscript and highSubscript. A call to this method prints only the designated portion of the array. ValidatelowSubscript and highSubscript. If either is out of range, the overloaded printArray method should throw anInvalidSubscriptException; otherwise, printArray should return the number of elements printed. Then modify main to exercise both versions of printArray on arrays integerArray, doubleArray and characterArray. Test all capabilities of both versions of printArray.

**20.6 (Overloading a Generic Method with a Nongeneric Method))** Overload generic method printArray of [Fig. 20.3](http://proquest.safaribooksonline.com/9780133813036/ch20lev1sec3_html#ch20fig03) with a nongeneric version that specifically prints an array of Strings in neat, tabular format, as shown in the sample output that follows:

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0868pro01a)

Array stringArray contains:  
one      two      three      four  
five     six      seven      eight

**20.7 (Generic** ***isEqualTo*** **Method)** Write a simple generic version of method isEqualTo that compares its two arguments with theequals method and returns true if they’re 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?

**20.8 (Generic Class** ***Pair*)** Write a generic class Pair which has two type parameters—F and S—each representing the type of the first and second element of the pair, respectively. Add get and set methods for the first and second elements of the pair. [Hint: The class header should be public class Pair<F, S>.]

**20.9 (Overloading Generic Methods)** How can generic methods be overloaded?

**20.10 (Overload Resolution)** 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?

**20.11****(What Does this Statement Do?)** Explain why a Java program might use the statement

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0868pro02a)

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