# Generic Classes and Methods

Java How to Program, 9/e

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# Objectives

- What are generics methods and generic classes
- Use generic methods and classes

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# 21.1 Introduction

- Generic methods and generic classes (and interfaces) 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.
- Generics also provide compile-time type safety that allows you to catch invalid types at compile time.

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# 21.2 Motivation for Generic Methods

- Overloaded methods are often used to perform similar operations on different types of data.
- Study each `printArray` method.
  - Note that the type array element type appears in each method's header and `for`-statement header.
  - If we were to replace the element types in each method with a generic name—`T` by convention—then all three methods would look like the one in Fig. 21.2.

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# 21.3 Generic Methods: Implementation and Compile-Time Translation

- If the operations performed by several overloaded methods are identical for each argument type, the overloaded methods can be more com-pactly and conveniently coded using a generic- method.
- You can write a single generic method declaration that can be called with arguments of different types.
- Based on the types of the arguments passed to the generic method, the compiler handles each method call appropriately.
- Line 22 begins method `printArray`'s declaration.

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# Code using Generics

- See code of print array using generics

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

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

- Generic method `maximum` determines and return the largest of its three arguments of the same type.
- The relational operator > cannot be used with reference types, but it's possible to compare two objects of the same class if that class implements the generic interface Comparable<T>
- All the type-wrapper classes for primitive types implement this interface.
- Like generic classes, generic interfaces enable you to specify, with a single interface declaration, a set of related types.

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# 21.4 Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type (cont.)

- `Comparable<T>` objects have a compareTo method.
  - The method must return `0` if the objects are equal, a negative integer if `object1` is less than `object2` or a positive integer if `object1` is greater than `object2`.
- A benefit of implementing interface `Comparable<T>` is that `Comparable<T>` objects can be used with the sorting and searching methods of class `Collections` (package `java.util`).

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# See code

- See code of generic method maximum

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# 21.4 Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type (cont.)

- The type-parameter section specifies that `T` extends `Comparable<T>`—only objects of classes that implement interface `Comparable<T>` can be used with this method.
- `Comparable` is known as the upper bound of the type parameter.
- By default, `Object` is the upper bound.
- Type-parameter declarations that bound the parameter always use keyword `extends` regardless of whether the type parameter extends a class or implements an interface.
- The restriction of using `Comparable<T>` objects is important, because not all objects can be compared.

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

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

- interface Comparable<T>
- All the type-wrapper classes for primitive types implement this interface.
- Like generic classes, generic interfaces enable you to specify, with a single interface declaration, a set of related types.

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# 2 Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type (cont.)

- Type-parameter declarations that bound the parameter always use keyword `extends` regardless of whether the type parameter extends a class or implements an interface.
- The restriction of using `Comparable<T>` objects is important, because not all objects can be compared.

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# Basics of Generics

- Class definitions may include parameters for types
  - Called *generics*
- Programmer now can specify any class type for the type parameter
- View class definition, listing 12.11

```
class Sample<T>
```

- Note use of `<T>` for the type parameter

# Basics of Generics

- Legal to use parameter T almost anywhere you can use class type
- Example declaration

```
Sample <String> sample1 =
      new Sample<String>();
```

- Cannot specify a primitive type for the type parameter

## LISTING 12.11 A Class Definition That Uses a Type Parameter

```
public class Sample<T>
{
    private T data;

    public void setData(T newValue)
    {
        data = newValue;
    }

    public T getData()
    {
        return data;
    }
}
```

# Check code

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# Generic Classes

- **LinkedList** is an example of a generic class that has this declaration

```
class LinkedList<E>
```

- Here, **E** specifies the type of objects that the list will hold.

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.

# Person List Example

- Check code

© Copyright 1992–2012 by Pearson Education, Inc. All Rights Reserved.