Abstract Classes

• Abstract classes (Continued)
  – A class that contains at least one abstract method must be declared as an abstract class
  – A subclass of an abstract class must be declared abstract if it does not provide implementations for all abstract methods in the superclass

• Example
  – CD player and DVD player
    • Both involve an optical disk
    • Operations
      – Insert, remove, play, record, and stop such discs

Figure 9-8
CD and DVD have an abstract base class GDP

Abstract Classes

• Abstract classes
  – An abstract class is used only as the basis for subclasses
    • It defines a minimum set of methods and data fields for its subclasses
  – An abstract class has no instances
  – An abstract class should, in general, omit implementations except for the methods that
    • Provide access to private data fields
    • Express functionality common to all of the subclasses

Java Interfaces Revisited

• A Java interface
  – Specifies the common behavior of a set of classes
  – Common uses
    • Facilitate moving from one implementation of a class to another
      – A client can reference a class’s interface instead of the class itself
    • Specify behaviors that are common to a group of classes
Java Interfaces Revisited

- Inheritance can be used to define a subinterface
- The Java API provides many interfaces and subinterfaces
  - Example: java.util.Iterable
    - An iterator is a class that provides access to another class that contains many objects

The ADTs List and Sorted List Revisited

- BasicADTInterface
  - Can be used to organize the commonalities between the ADT list and the ADT sorted list
- ListInterface
  - A new interface based on BasicADTInterface

Implementation of the ADT Sorted List That Used the ADT List

- Operations
  - createSortedList()
  - isEmpty():boolean (query)
  - size():integer [query]
  - sortedAdd(in newItemType:ListItemType) throw ListException
  - sortedRemove(in anItem:ListItemType) throw ListException
  - removeAll()
  - get(in index:integer) throw ListIndexOutOfBoundsException
  - locateIndex(in anItem:ListItemType):integer (query)

Java Generics: Generic Classes

- ADT developed in this text relied upon the use of Object class
- Problems with this approach
  - Items of any type could be added to same ADT instance
  - ADT instance returns objects
    - Cast operations are needed
    - May lead to class-cast exceptions
- Avoid this issues by using Java generics
  - To specify a class in terms of a data-type parameter

Generic Wildcards

- Generic classes are not necessary related
- Generic ? wildcard
  - Stands for unknown data type
- Example
  ```java
  public void process(NewClass<? extends String> temp)
  {
    System.out.println("getData() => " + temp.getData());
  } // end process
  ```
Generic Classes and Inheritance

- You can use inheritance with a generic class or interface
- Method overriding rules
  - Declare a method with the same parameters in the subclass
  - Return type is a subtype of all the methods it overrides
- It is sometimes useful to constrain the data-type parameter to a class or one of its subclasses or an implementation of a particular interface
  - To do so, use the keyword `extends`

Generic Methods

- Method declarations can also be generic
  - Methods can use data-type parameters
- Generic methods are invoked like regular non-generic methods
- Example
  
  ```java
  public static <T extends Comparable<? super T>> void sort(ArrayList<T> list) {
    // implementation of sort appears here
  }
  ```

Abstract Classes

- `Object`
- `Person`
- `Student`

![Sample class hierarchy](image)

Iterators

- Iterator
  - Object that can access a collection of objects one object at a time
  - Traverses the collection of objects
- JCF defines generic interface `java.util.Iterator`
  - And a subinterface `ListIterator`

Summary

- A subclass inherits all members of its previously defined superclass, but can access only the public and protected members
- Subclasses and superclasses
  - A subclass is type-compatible with its superclass
  - The relationship between superclasses and subclasses is an is-a relationship
- A method in a subclass overrides a method in the superclass if they have the same parameter declarations

Summary

- An abstract method in a class is a method that you can override in a subclass
- A subclass inherits
  - The interface of each method that is in its superclass
  - The implementation of each nonabstract method that is in its superclass
- An abstract class
  - Specifies only the essential members necessary for its subclasses
  - Can serve as the superclass for a family of classes
Summary

- Early (static) binding: compiler determines at compilation time the correct method to invoke
- Late (dynamic) binding: system determines at execution time the correct method to invoke
- When a method that is not declared `final` is invoked, the type of object is the determining factor under late binding
- Generic classes enable you to parameterize the type of a class’s data
- Iterators provide an alternative way to cycle through a collection of items