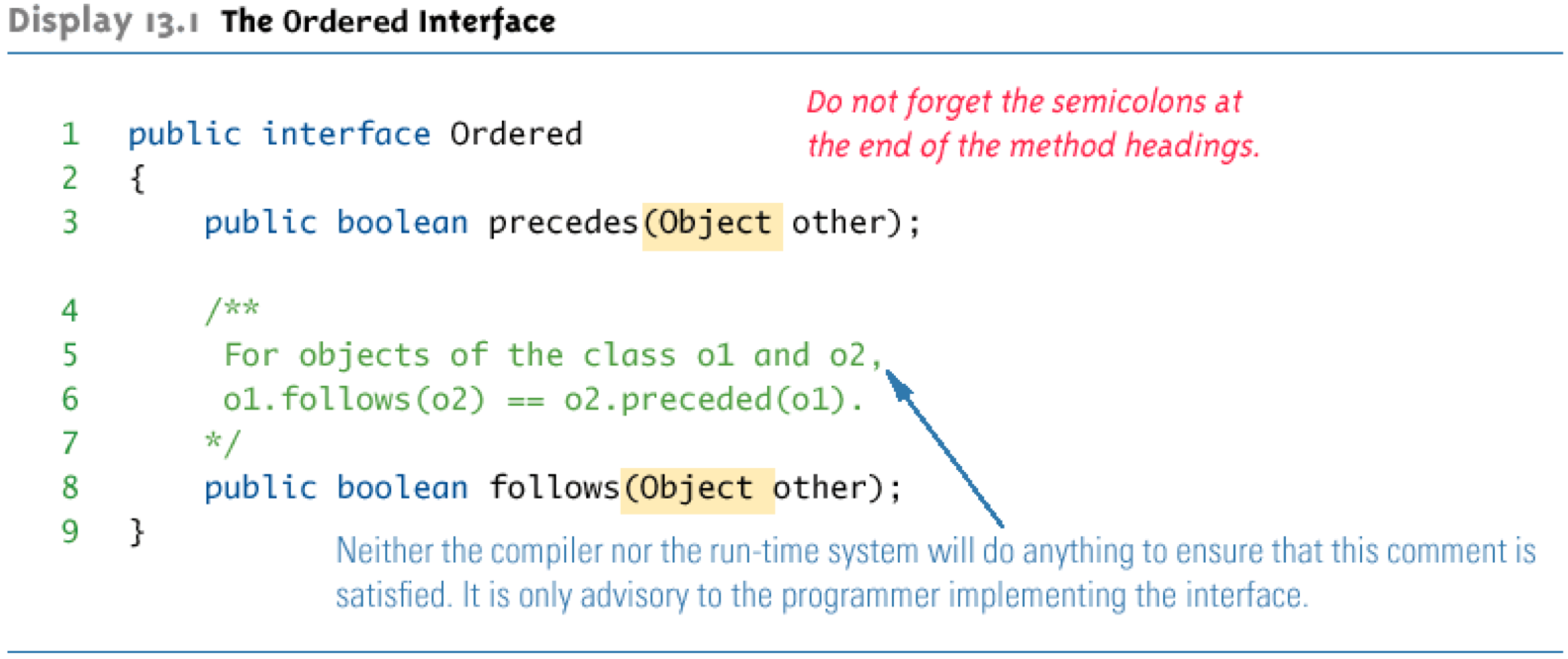
Chapter 13 Interfaces and Inner Classes

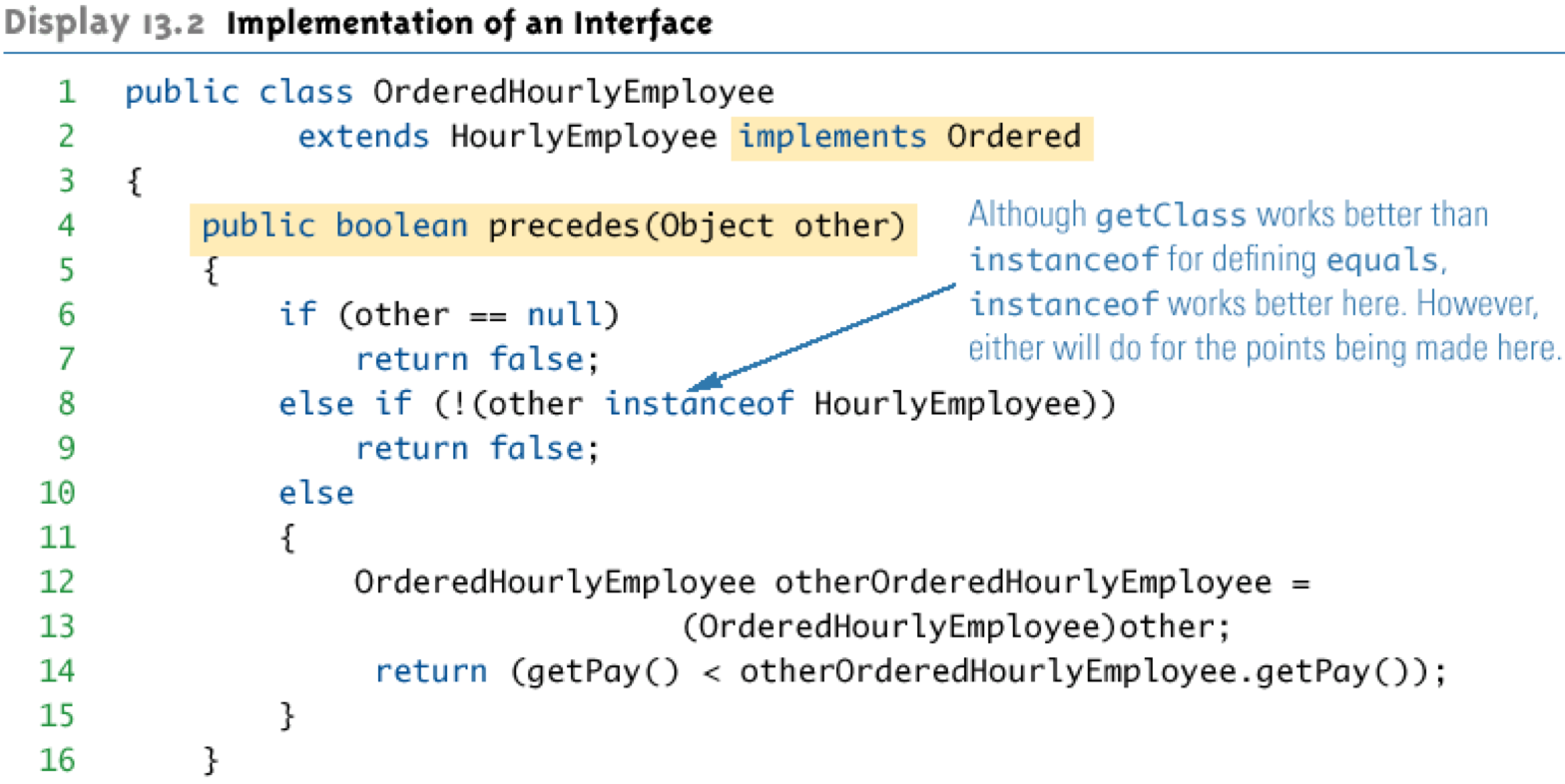
interface:

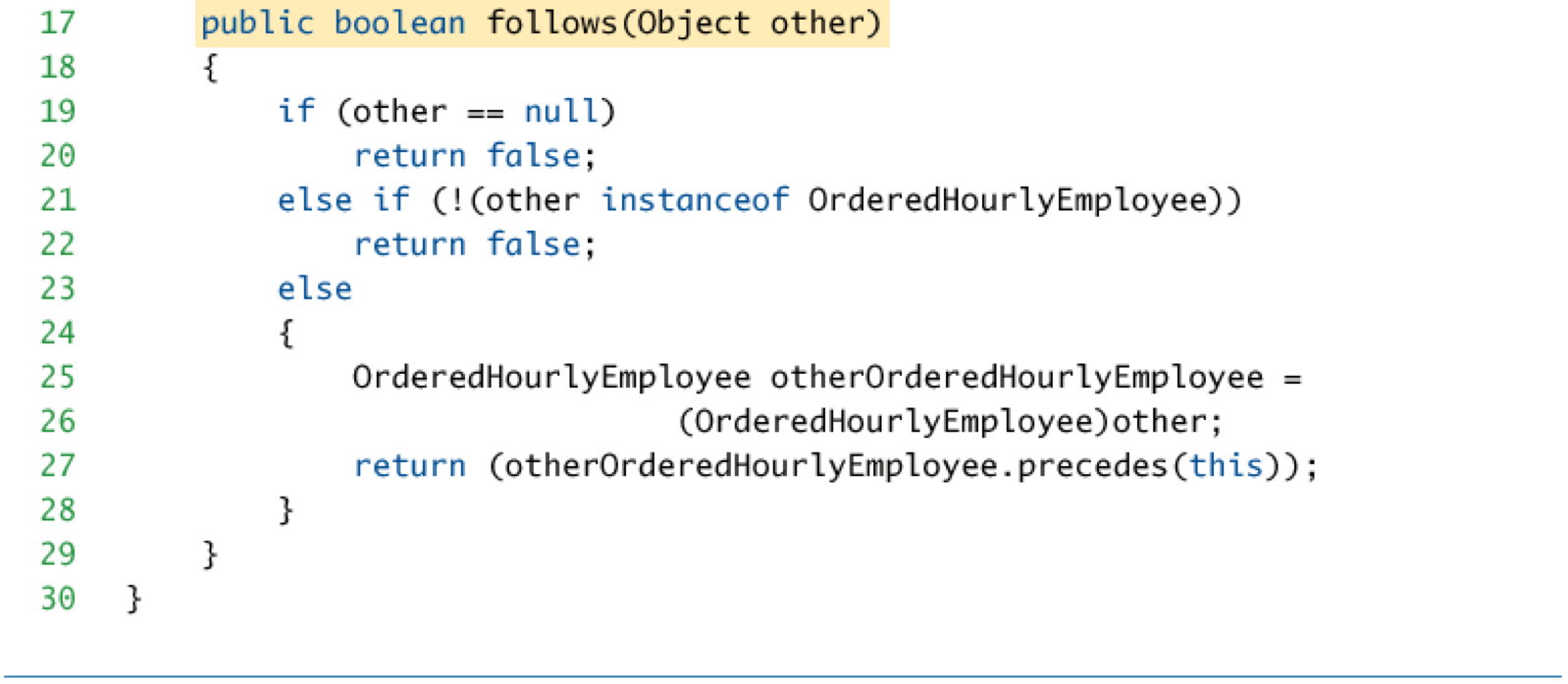
1. an interface contains method headings and constant definitions only, no instance variables, no concrete methods
2. Java should do mutiple inheritance through interfaces
3. all method headings be public
4. to implement an interface…
   1. concrete class must implement all the method headings in the interface
   2. concrete class needs to include “implements Interface\_Name” in the class declaration
5. 多重繼承會出問題，例如B和C繼承A，D繼承B和C，D在執行override method的時候不知道要用哪一個版本
6. implements, extends傻傻分不清楚？
   1. abstract class/class extends abstract class/class
   2. abstract class/class implements interface
   3. interface extends interface
   4. interface implements abstract class/class (不存在，因為interface不實作)
7. for example

interface:

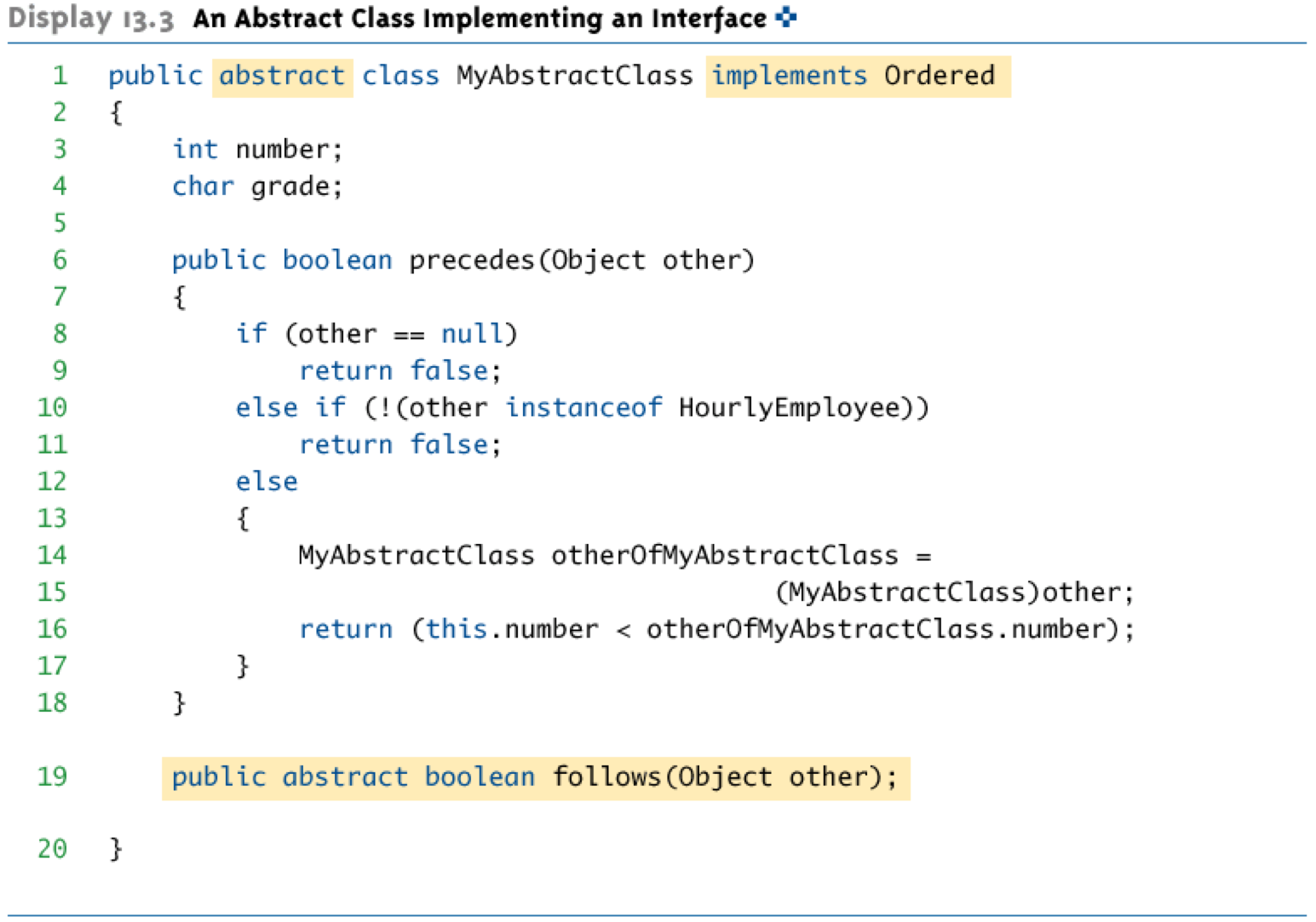


class implements interface:

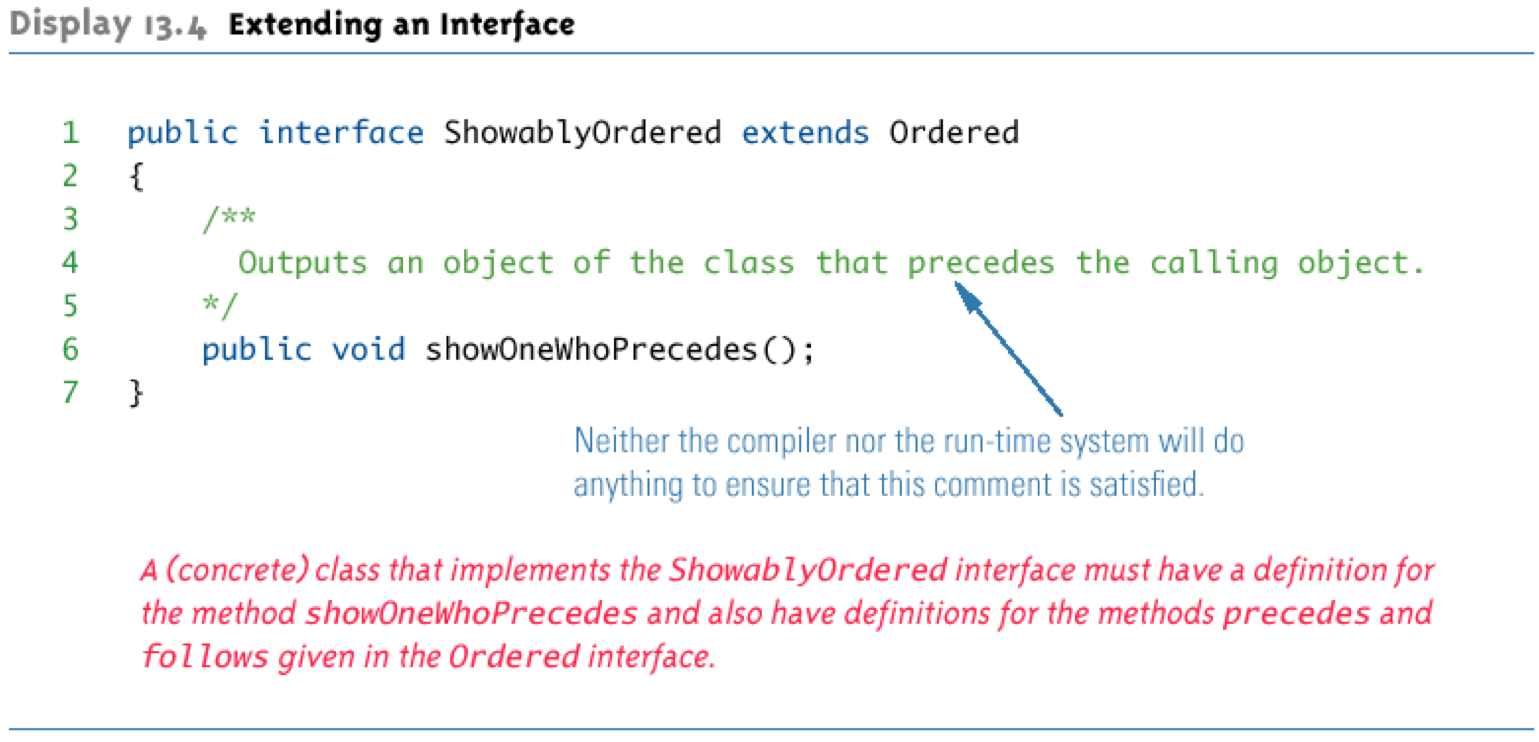




abstract class implements interface:



interface extends interface:



1. Comparable Interface
   1. in java.lang package
   2. all Wrapper class has Comparable interface
   3. Comparable has only one method heading:

/\*\* returns a negative number if the calling object “comes

before” the parameter

returns zero if the calling object “equals” the

parameter

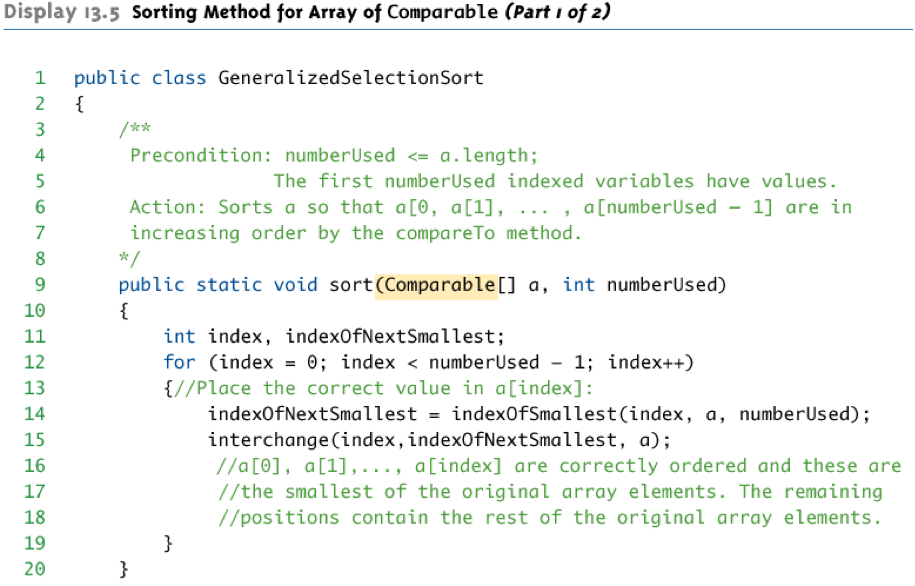
returns a positive number if the calling object “comes

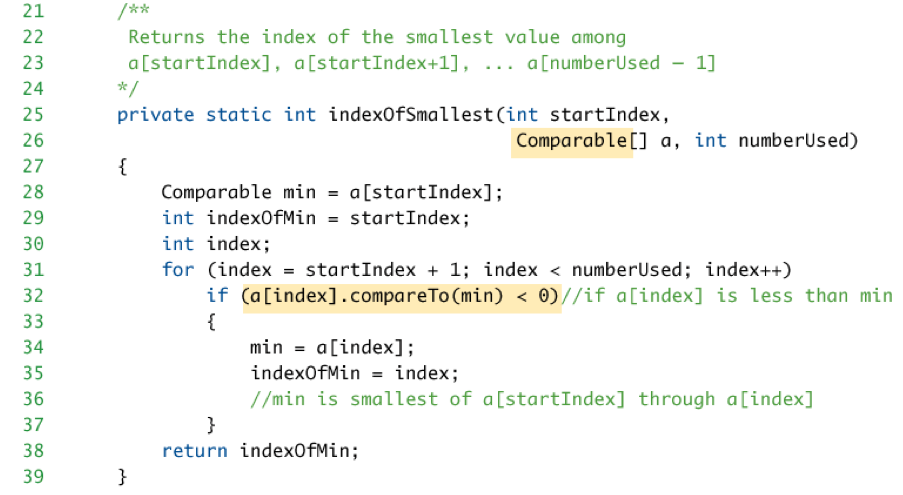
after” the parameter

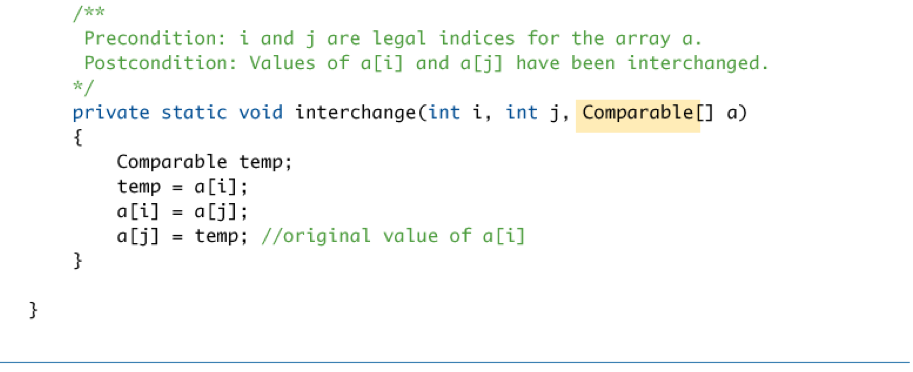
\*/

public int compareTo(Object other)

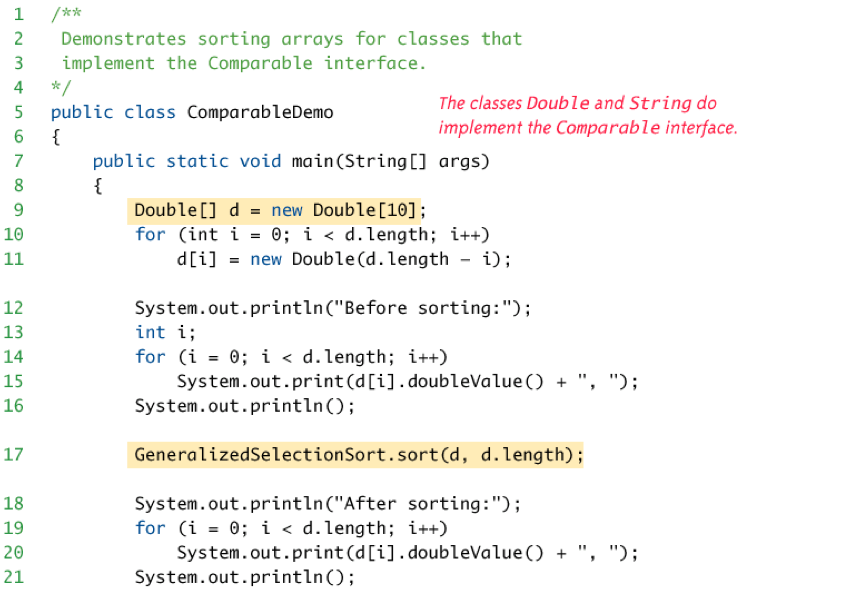
* 1. 宣告interface的地方必須要打上註解，因為在interface內沒有實作出來，所以需要註明如何讓繼承者使用
  2. for example: GeneralizedSelectionSort class，它負責接收Comparable類型的reference

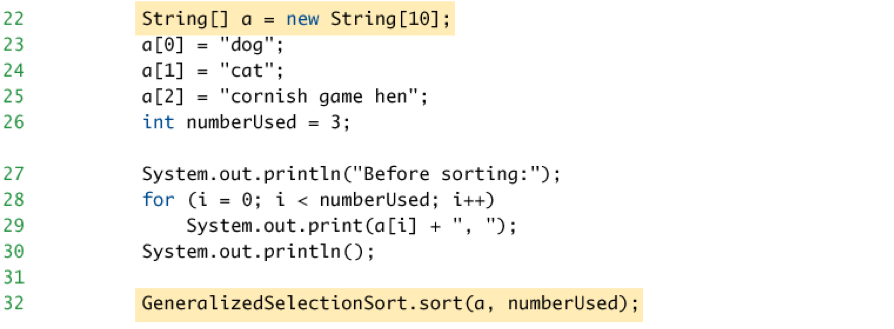


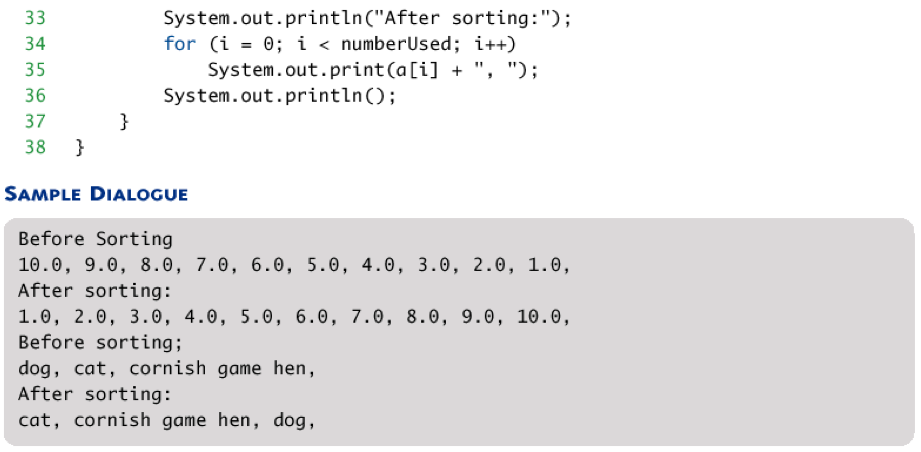


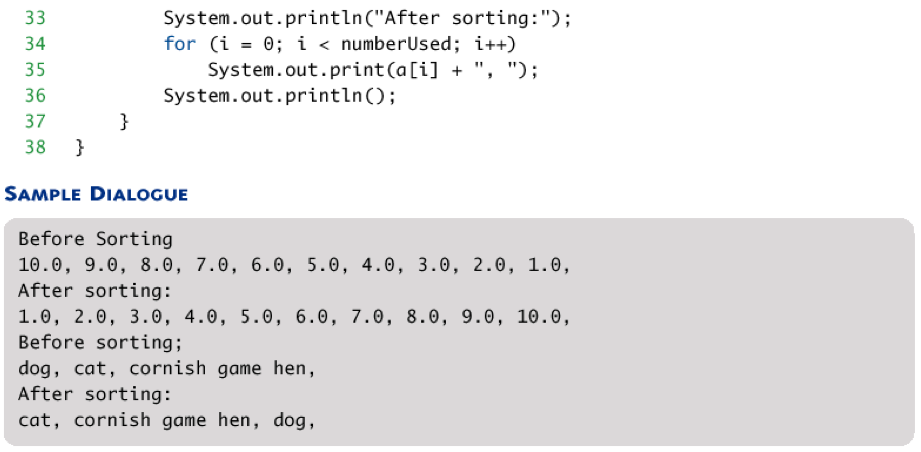


使用GeneralizedSelectionSort的方法：









1. 當一個interface沒有method headings也無constants（completely empty），其用途僅作分類、標記，例如：
   1. Serializable Interface
   2. Cloneable Interface：
      1. 所有繼承Object class都有的clone，但是不一定有被override。若想要標示有override clone method，可以implement Cloneable interface。所以Cloneable interface是用來指示clone method是否有被使用
      2. (以後才會回來看exception)

Inner Classes：

1. inner class is a member of the outer class
2. if inner class is private, then outside the outer class is unaccessable to inner class
3. advantage:
   1. inner class can be hidden
   2. inner class can be used as helping class
4. inner and outer classes have access to each other’s private members, for example:

public class **Outer**{

**private** int x1;

**private** int x3;

**private** void m1(){

**Inner inRef = new Inner();**

**inRef.x2 = 1; //private member access**

}

**private** class **Inner**{

**private** int x2;

**private** int x3;

**private** void m2(){

**x1 = 2; //private member access**

**x3 = 3; //自己的x3**

**Outer.this.x3 = 4; //Outer private access**

}

}

}

1. Java produces a .class file for any class named ClassName.class for outer class and ClassName$InnerClassName.class for inner class
2. static inner class
   1. instance variables of the outer class cannot be referenced
   2. nonstatic methods of the outer class cannot be invoked
3. public inner class
   1. can be used outside of the outer class
   2. 宣告方法：

OuterClass.InnerClass inObject=new OuterClass.InnerClass()

1. derived inner class is allowed, for example

public class Outer{

private int x1;

private int x3;

private void m1(){

Inner inRef = new Inner();

inRef.x2 = 1; //private member access

}

**private** class Innerextends Outer{

private int x2;

private int x3;

private void m2(){

x1 = 2; //private member access

x3 = 3; //自己的x3

Outer.this.x3 = 4; //Outer private access

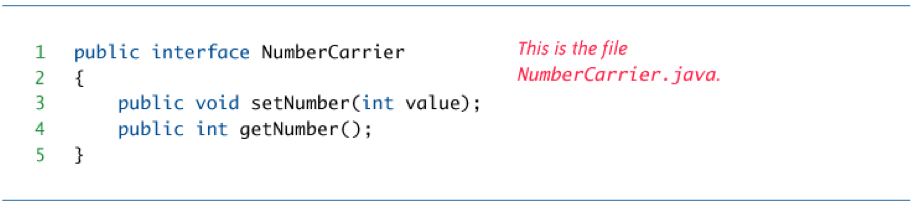
}

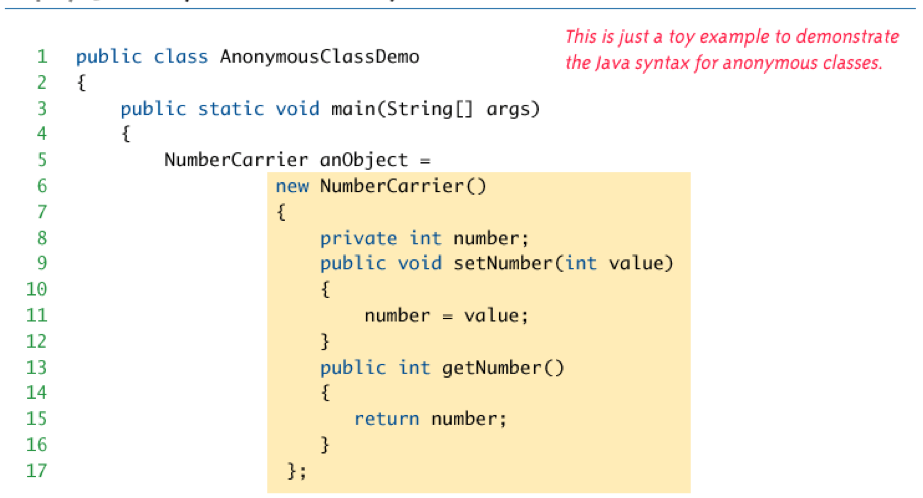
}

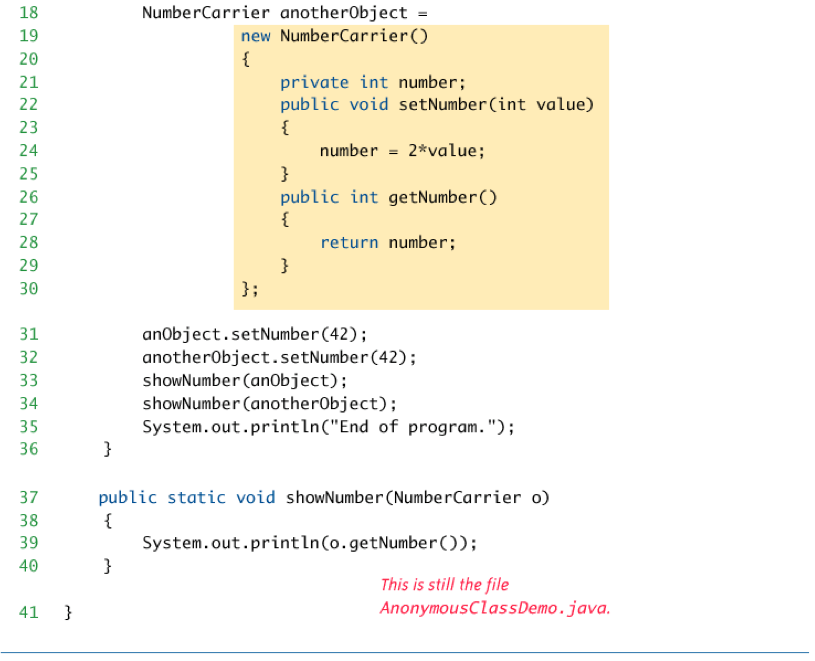
}

//Inner class即包含了Outer class所有的data member和member function

1. Anonymous Class
   1. temporary use of class
   2. class definition is embedded inside the expression with the new operator
   3. for example:







1. inner class規則
   1. interface內不可有inner class (因為interface不實作東西)
   2. interface內可有inner interface
   3. class內可有inner abstract class
   4. class內可有inner interface
   5. abstract內可有inner class
   6. abstract內可有inner interface