BITS Pilani

Object-Oriented Programming (CS F213)

Module III: Inheritance and Polymorphism in Java

CS F213 RL 10.3: Comparable and Comparator Interfaces

Dr. Pankaj Vyas
Department of Computer Science, BITS-Pilani, Pilani Campus

CS F213 RL 10.3 : Topics

- Comparable and Comparator Interfaces in Java

Comparable Interface

- Provides an interface for comparing any two objects of same class.
- General Form :

Requires Type Casting

1. Un-parameterized Form

```
public          interface          Comparable
{
        public          int          compareTo(Object o);
}
```

2. Parameterized Form

```
public          interface          Comparable<T>
{
        public          int          compareTo(<T> o);
}
```

- By implementing this interface , programmers can implement the logic for comparing two objects of same class for less than, greater than or equal to. Helps in Sorting.

How to Implement Comparable Interface (Un-Parameterized)

```
class BOX Implements Comparable
{
………………………………………….
………………………………………….
………………………………………….
public int compareTo(Object other)
{
BOX box = (BOX) other;
......Logic for comparison ….
} // End of Method
…………………………………….
} // End of class Box
```

```
class Student Implements Comparable
{
………………………………………….
………………………………………….
………………………………………….
public int compareTo(Object other)
{
Student std = (Student) other;
......Logic for comparison ….
}
…………………………………….
}// End of class Student
```

How to Implement Comparable Interface (Parameterized)

```
class BOX Implements Comparable<BOX>
{
……………………………………………
……………………………………………
……………………………………………
public int compareTo(Box other)
{
        ......Logic for comparison ….
} // End of Method
………………………………………
} // End of class Box
```

```
class Student Implements Comparable<Student>
{
……………………………………………
……………………………………………
……………………………………………
public int compareTo(Student other)
{
        ......Logic for comparison ….
}
………………………………………
}// End of class Student
```

Comparable Interface : Example 1

```
// File Name : ComparableTest.java
class Box
{
        // Instance Fields
        private double length;
        private double width;
        private double height;
        // Constructor
        Box(double l, double b, double h)
        {
                length=l;   width=b;   height=h;
        }
        // Accessor Methods
        public double getLength() { return length;}
        public double getWidth()  { return width;}
        public double getHeight() { return height;}
        // Area Method
        public double area()
        {
                return 2*(length*width + width*height+height*length);
        }
// Volume Method
public double volume()
{
return length*width*height;
}
public      String      toString()
{
  String s1 = “Length = “+ length;
  String s2 = “Width = “+ width;
  String s3 = “Height = “+ height;
  String  s4 = “Area =“+ area();
  String  s5 = “Volume=“+volume();
  return s1 + s2 + s3 + s4 + s5;
} // End of Method
}// End of BOX class
```

Comparable Interface : Example 1 …

```
class Test
{
        public    static    void    main(String args[])
        {
                  int[]     data      = {10, -5, 56, 78, 11, 89, 23};
                  String[]  names     = {“Cornell”, “Horstmann”, “Herbert”, “David”, “Elina”};

                  Box[]     boxes     =         new Box[5];
                  boxes[0]  =  new  Box(10,6,7);
                  boxes[1]  =  new  Box(10,20,5);
                  boxes[2]  =  new  Box(5,20,25);
                  boxes[3]  =  new  Box(40,30,45);
                  boxes[4]  =  new  Box(100,16,8);

                  Arrays.sort(data);    for (int  i : data)          System.out.println(i);
                  Arrays.sort(names); for (String  i : names)     System.out.println(i);
                  Arrays.sort(boxes);  for(Box  i : boxes)          System.out.println(i);
        }// End of Method
}// End of class Test
```

Comparable Interface : Example 1 …

OUTPUT

```
-5
10
11
23
56
78
89
Cornell
David
Elina
Herbert
Horstmann
Exception in thread "main" java.lang.ClassCastException: Box cannot be cast to
java.lang.Comparable
	at java.util.ComparableTimSort.countRunAndMakeAscending(Unknown Source)
	at java.util.ComparableTimSort.sort(Unknown Source)
	at java.util.Arrays.sort(Unknown Source)
	at Test.main(CompTest.java:54)
```

Comparable Interface : Example 2

- To use sort() method, the class must implement Comparable Interface. Make Any of the following changes in Example 1.

```
// File Name : ComparableTest.java
class      Box      implements        Comparable
{
           public              int        compareTo(Object o)
           {
                    Box b = (Box) o;
                    return (int) (this.area() - b.area());
           } // End of Method
} // End of class Box
```

```
// File Name : ComparableTest.java
class      Box      implements        Comparable<Box>
{
           public              int        compareTo(Box o)
           {
                    return (int) (this.area() - b.area());
           } // End of Method
} // End of class Box
```

Problems with Comparable Interface

- Method int compareTo(Object obj) needs to be included in the base class itself.
- Only one ordering logic can be active at a time.
- Different comparison order requires changes in the base class itself.
- Each time we need different order we need to change the code itself.

Comparator Interface

- Also provides an interface for comparing any two objects of same class.
- But, the two objects that are to compared have to be passed explicitly
- General Form :

1. Un-parameterized Form (Requires Type Casting of Object Type Parameters)

```
public          interface          Comparator
{
        public          int          compare(Object first, Object second);
}
```

2. Parameterized Form

```
public          interface          Comparator<T>
{
        public          int          compare(T first, T second);
}
```

Comparator Interface Example

```
// File Name: comp.java
class Box
{
		// Assume the Implementation From the Previous Slides
}// End of class Box
// Write Your Own Comparator Classes
class	BoxComparisonByLength	implements		Comparator<Box>
{
		public	int		compareTo(Box first, Box Second)
		{
				return (int) (first.getLength() – second.getLength());
		} // End of Method
}// End of class BoxComparisonByLength
class	BoxComparisonByArea		implements		Comparator<Box>
{
		public	int		compareTo(Box first, Box Second)
		{
				return (int) (first.area() – second.area());
		} // End of Method
}// End of class BoxComparisonByArea
```

Comparator Interface Example …

```
class       BoxComparisonByAreaLength implements       Comparator<Box>
{
            public    int       compareTo(Box first, Box Second)
            {
                      double    a1        =         first.area();
                      double    a2        =         second.area();

                      if        (a1 == a2)
                                return    (int) (a1.getLength() – a2.getLength());
                      else
                                return    (int) (a1.area() – a2.area());
            } // End of Method
}// End of class BoxComparisonByAreaLength
```

Comparator Interface Example …

```
// Driver Class
class Test
{
    public      static   void    main(String      args[])
    {
        Box[]    boxes    =      new     Box[5];
        // Filling Elements
        boxes[0] =  new  Box(10,6,7);
        boxes[1]  =  new  Box(10,20,5);
        boxes[2] =  new  Box(5,20,25);
        boxes[3]  =  new  Box(40,30,45);
        boxes[4] =  new  Box(100,16,8);

        // Creating Comparator Instances

        Comparator<Box> bC      =      new     BoxComparisonByLength();
        Arrays.sort(boxes, bC);

        bC      =      new     BoxComparisonByArea();

        Arrays.sort(boxes, bC);
    }// End of Method
}// End of  class Test
```

Sorts By Length of Box

Sorts By Area of Box

innovate achieve lead

Thank You