APRENDERAPROGRAMAR.COM

EJERCICIO Y EJEMPLO RESUELTO: USO DE LA INTERFACE COMPARABLE Y MÉTODO COMPARETO DE JAVA. COMPARAR OBJETOS (CU00913C)

Sección: Cursos
Categoría: Lenguaje de programación Java nivel avanzado I

Fecha revisión: 2039

Resumen: Entrega nº 13 curso "Lenguaje de programación Java Nivel Avanzado I".

Autor: Manuel Sierra y José Luis Cuenca

EJERCICIO Y EJEMPLO RESUELTO

Al igual que hemos introducido anteriormente la interfaz **Cloneable**, ahora vamos a realizar un ejercicio resuelto acerca de cómo una clase debe hacer uso de la interfaz **Comparable**. Esto nos va a permitir que en dicha clase se pueda realizar la comparación de objetos, permitiendo hacer ordenaciones de los mismos, y por ejemplo poderlos utilizar en posteriores clases que más adelante veremos, cómo son los conjuntos ordenados (sortedset).

Vamos a partir también del siguiente código sobre el cual ya trabajamos en la interfaz Cloneable e iremos haciendo las modificaciones necesarias:

```
/* Ejemplo Clase e Interfaz Comparable aprenderaprogramar.com */
public class Persona {
  public int dni, edad;
    public Persona( int d, int e){
       this.dni = d;
       this.edad = e;
  }
}
```

Tenemos por tanto la misma clase Persona con sus atributos dni y edad que son valores enteros. Ahora vamos a implementar la interfaz Comparable y para ello estamos obligados a implementar (sobreescribir) el método **public int compareTo(Persona o).** Este método debe devolver un número negativo, 0, o un número positivo en función de que el objeto que comparemos con el objeto "o" de referencia sea menor, igual o mayor respectivamente que ese objeto de referencia.

Por tanto vamos a incluir el método **public int compareTo(Persona o)** en nuestra clase Persona. Veamos cómo hacerlo. Vamos a cambiar un poco la lógica quizás, pero es un ejemplo. Si tuviéramos 2 Personas y quisiéramos compararlas u ordenarlas lo podríamos hacer por ejemplo por su altura, o por su edad, la pregunta a hacerse sería cuándo una persona es más grande que otra o cuándo es menor, o cuándo son iguales atendiendo a algún tipo de atributo. En nuestro ejemplo nos vamos a inventar esta ordenación y vamos a decir que una persona es más grande que otra si tiene la edad mayor, en cambio será menor si tiene una edad menor y serán iguales si y solo si su edad es igual y el dni también es igual. En este último caso si dos personas tienen la misma edad, se considerará mayor el que tenga el dni más grande. Fíjate que el criterio de ordenación lo hemos decidido nosotros. Una vez decidido, lo

implementaremos en forma de código. Distintos programadores pueden aplicar distintos criterios a la hora de implementar el método compareTo. No obstante, recuerda siempre aplicar el sentido común y pensar en que los resultados que se obtengan deben ser lógicos.

Con los criterios elegidos, nuestro código para la clase Persona quedaría de la siguiente manera:

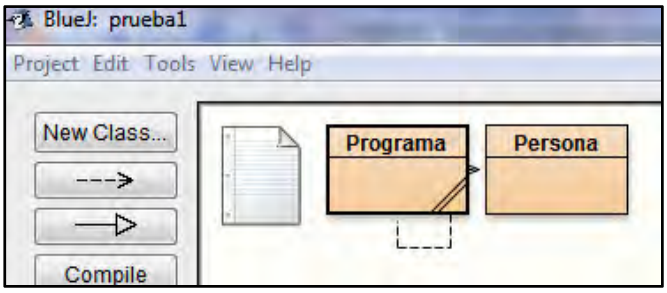
```
/* Ejemplo Clase e Interfaz Comparable aprenderaprogramar.com */
public class Persona implements Comparable<Persona>{
       public int dni, edad;

        public Persona( int d, int e){
        this.dni = d;
        this.edad = e;
  }

   public int compareTo(Persona o) {
      int resultado=0;

      if (this.edad<o.edad) {   resultado = -1;     }
      else if (this.edad>o.edad) {    resultado = 1;     }
      else {

         if (this.dni<o.dni) {  resultado = -1;   }
         else if (this.dni>o.dni) {   resultado = 1;   }
         else {   resultado = 0;   }
      }
      return resultado;
   }
}
```

Como podemos ver resaltado hemos incluido el código **implements Comparable<Persona>**, que nos indica que vamos a implementar en la clase la interfaz Comparable para el parámetro Persona. También hemos incluido el código necesario para implementar el método **public int compareTo(Persona o),** donde además podemos observar que en el caso que las dos personas tengan la misma edad, comparamos sus dni para determinar qué persona es mayor que otra. Es un método que nos hemos creado nosotros y que tal vez no sea aplicable a la vida real, pero nos sirve a modo de ejemplo ya que en general este método será distinto para cada clase dependiendo de cómo queramos comparar los objetos con los que estemos tratando.

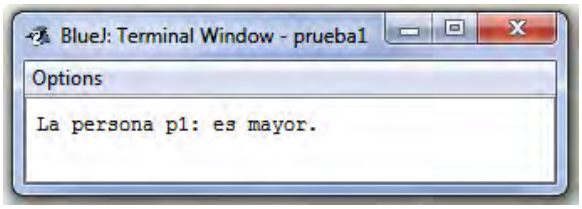
Ahora al igual que en el uso de la interfaz Cloneable vamos a codificar una clase “Programa”, que básicamente lo que hará será generar 2 Personas y compararlas mostrando cuál es la mayor.

```
/* Ejemplo Clase e Interfaz Comparable aprenderaprogramar.com */
public class Programa {

   public static void main(String arg[]) {

      Persona p1 = new Persona(74999999,35);
      Persona p2 = new Persona(72759474,30);

    if (p1.compareTo(p2) < 0 ) { System.out.println("La persona p1: es menor."); }

   else if(p1.compareTo(p2) > 0 ) {System.out.println("La persona p1: es mayor."); }

    else { System.out.println ("La persona p1 es igual a la persona p2"); }
     }
}
```

Así al igual que en el caso anterior para Cloneable, ahora tendremos para Comparable las siguientes clases definidas en BlueJ:



La salida del programa donde hemos hecho uso del método **compareTo** es la siguiente. Ahí podemos ver que la persona p1 es mayor que la persona p2 definida ya que su edad era 35 años mientras que para la persona p2 era 30. Por lo que el método **compareTo** que implementa la clase Persona devuelve un 1 indicando que la persona p1 es mayor.



Por tanto cada vez que deseemos que una clase implemente esta interfaz habrá que hacer algo similar a lo realizado en la implementación del método **compareTo()**, es decir, deberemos de devolver un número negativo cuando el objeto que invoca el método **compareTo(Object o)** sea menor que el objeto que se le pasa para la comparación. En este caso hemos llamado al objeto de la comparación "o", que es de la clase Object, pero cuando estemos implementando el método este objeto será de la clase en particular que sea. Así en nuestro ejemplo anterior este objeto a comparar era de la clase Persona

evidentemente. Ten en cuenta que un objeto Persona es también un objeto Object por el mecanismo de herencia de Java.

Esta implementación de la interfaz Comparable es muy habitual ya que en general muchas veces vamos a desear que nuestras clases sean comparables no solo por el hecho de poder comparar objetos de dicha clase, sino por el uso de poder ordenar objetos de esa clase. Como es sabido la ordenación es una de las funciones más importantes y necesarias en el ámbito de la informática ya que tener la información ordenada hace mucho más rápidas las consultas o modificaciones de datos.

Finalmente planteamos una reflexión: ¿Por qué crees que se usa la implementación de interfaces en Java? Vamos a tratar de dar una respuesta. Hay varios motivos, pero uno principal es que mediante la implementación de interfaces todos los programadores usan el mismo nombre de método y estructura formal para comparar objetos (o clonar, u otras operaciones). Imagínate que estás trabajando en un equipo de programadores y tienes necesidad de utilizar una clase que ha codificado otro programador. Gracias a que se implementan interfaces Java, no necesitarás estudiar el código para ver qué método hay que invocar para comparar objetos de esa clase. Dado que todos los programadores usan la implementación de interfaces, sabemos que la comparación de objetos (o clonación, etc.) debe de hacerse invocando determinados métodos de determinada forma. Esto facilita el desarrollo de programas y la comprensión de los mismos, especialmente cuando hablamos de programas o desarrollos donde pueden intervenir cientos o miles de clases diferentes.

EJERCICIO

Utilizando Comparable y compareTo resuelve el siguiente problema donde debemos partir de una clase Persona con atributos nombre, edad y altura. Queremos ordenar por edad y por altura a las siguientes personas:

Nombre	Altura	Edad
Mario	187	22
Pepe	173	52
Manuel	158	27
David	164	25
Alberto	184	80

Debemos comparar las personas y ordenarlas por altura primero (de mayor a menor) y por edad (de menor a mayor) después. Por pantalla debe mostrarse la lista de personas sin ordenar, ordenada por altura y ordenada por edad.

Ejemplo de ejecución:

Personas sin ordenar

1. Mario - Altura: 187 - Edad: 22
2. Pepe - Altura: 173 - Edad: 52
3. Manuel - Altura: 158 - Edad: 27
4. David - Altura: 164 - Edad: 25
5. Alberto - Altura: 184 - Edad: 80

Personas ordenadas por altura

1. Mario - Altura: 187 - Edad: 22
2. Alberto - Altura: 184 - Edad: 80
3. Pepe - Altura: 173 - Edad: 52
4. David - Altura: 164 - Edad: 25
5. Manuel - Altura: 158 - Edad: 27

Personas ordenadas por edad

1. Mario - Altura: 187 - Edad: 22
2. David - Altura: 164 - Edad: 25
3. Manuel - Altura: 158 - Edad: 27
4. Pepe - Altura: 173 - Edad: 52
5. Alberto - Altura: 184 - Edad: 80

Para comprobar si es correcta tu solución puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00914C

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=58&Itemid=180