Soluciones Ejercicios Tema 3

Germán Moltó
gmolto@dsic.upv.es
Estructuras de Datos y Algoritmos
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Solución Ejercicio Comparación

```
public class Manzana implements Comparable<Manzana>
{
  private int sabor;
  public final static int SABOR_MINIMO = 0;
  public final static int SABOR_MAXIMO = 10;

  public Manzana(){sabor = (SABOR_MAXIMO-SABOR_MINIMO)/2;}
  public Manzana(int sabor){this.sabor = sabor;}
  public int getSabor(){return sabor;}

  public int compareTo(Manzana o){
     if (getSabor() > o.getSabor()) return 1;
     else if (getSabor() == o.getSabor()) return 0;
     else return -1;
  }}
```


Cálculo del Mínimo en Array Genérico (I)

```
public class Operaciones
{
   public static <T extends Comparable<T>> T minimo(T a[]){
      T min = a[0];  int i = 1;
      while (i < a.length){
        if (a[i].compareTo(min) < 0) min = a[i];
        i++;
      }
     return min;
   }
}//Fin de la Clase Operaciones
```


Cálculo del Mínimo en Array Genérico (II)

```
public class TestOperaciones
{
  public static void main(String args[]){
      Integer v[] = new Integer[10];
     for (int i = 0 ; i < 10; i++) v[i] = 10 - i;
    Operaciones.minimo(v);
  }
}
```

- La clase Operaciones permite buscar el mínimo de cualquier vector de objetos cuya clase implemente la interfaz Comparable<T>.

Solución: Ejercicio de Vehiculos (1)

```
package vehiculos;
public abstract class Vehiculo implements Comparable<Vehiculo>
{
   private String modelo;
   private float potencia;
   public Vehiculo(String modelo, float potencia){
     this.modelo = modelo;
     this.potencia = potencia;
    }
    public String getModelo() {return this.modelo;}
    public float getPotencia(){return this.potencia;}
   ...
```


Solución: Ejercicio de Vehiculos (2)

```
 public int compareTo(Vehiculo o){
        float diff = getPotencia() – o.getPotencia();
        if (diff > 0) return 1;
        else if (diff < 0) return -1;
             else return 0;
    }
    public boolean equals(Object x){
      return this.compareTo( (Vehiculo) x) == 0;
   }
     public abstract int getNumeroRuedas();
}
```


Solución: Ejercicio de Vehículos (3)

```
package vehiculos;
public class Moto extends Vehiculo
{
  private boolean carenado;
  public Moto(String modelo, float potencia, boolean carenado){
     super(modelo, potencia);
     this.carenado = carenado;
  }
  public boolean llevaCarenado(){return carenado;}
  public int getNumeroRuedas(){return 2;}
}
```


Solución: Ejercicio de Vehiculos (4)

```
package vehiculos;
public class Coche extends Vehiculo
{
  boolean elevaLunasElectrico;
  public Coche(String modelo, float potencia, boolean ee){
     super(modelo,potencia);
     this.elevaLunasElectrico = ee;
    }
  public boolean getElevaLunasElectrico(){return
    elevaLunasElectrico;}
  public int getNumeroRuedas(){return 4;}
}
```

Solución: Ejercicio de Vehiculos (5)

```
package vehiculos;
import ordenacionArray.*;
public class TestVehiculo
{
   public static void main(String args[]){
       Random r = new Random();
      Vehiculo v[] = new Vehiculo[10];

       for (int i = 0 ; i < 10 ; i++){
         v[i] = new Moto("Honda CBR 900 RR", 152 + r.nextInt(), true);
       }
       Ordenacion.insercionDirecta(v);
    }
}
```


Solución: Garaje de Vehículos

```
public class Garaje<V extends Vehiculo>{
   private ArrayList<V> elArray;
   public Garaje(){ this.elArray = new ArrayList<V>(); }
   public void aparca(V vehiculo){ this.elArray.add(vehiculo); }
   public void retira(V vehiculo) throws VehiculoInexistente {
    if (!this.elArray.remove(vehiculo))
          throw new VehiculoInexistente();
    }
   public static void main(String args[]){
     Garaje<Vehiculo> g = new Garaje<Vehiculo>();
     g.aparca(new Moto("Honda", 900, true));
   }}
```


Solución Ejercicio: Películas en DVD (I)

```
public class PeliculaEnVenta extends PeliculaEnDvd implements
   Comparable<PeliculaEnVenta> {
 protected double precio;
 protected int copiasDisponibles;
public PeliculaEnVenta(String t, String d, int a, double p, int c) {
    super(t, d, a);
    this.precio = p;  this.copiasDisponibles = c;
}
public int compareTo(PeliculaEnVenta o){
   if (this.anyo < o.anyo) return -1;
   else if (this.anyo > o.anyo) return 1;
   else return this.titulo.compareTo(o.titulo) ;
}
```


Solución Ejercicio: Películas en DVD (II)

```
...
 public boolean equals(Object x){
    return ( this.compareTo((PeliculaEnVenta)x) == 0 ) ;
 }

public int copiasDisponibles(){
   return this.copiasDisponibles;
}
public void decrementarCopiasDisponibles(){
   this.copiasDisponibles--;
 }
} //Fin de la clase PeliculaEnVenta
```

Solución Ejercicio: Películas en DVD (III)

```
public class GrupoDePeliculas{
  PeliculaEnVenta vPeliculas[];

  public GrupoDePeliculas(int talla){
    vPeliculas = new PeliculaEnVenta[talla];
   }
   ...
public String toString(){
  Ordenacion.insercionDirecta(vPeliculas);
  for ( int i = 0 ; i < vPeliculas.length ; i++)
    System.out.println(vPeliculas[i])
 }
...
```


Solución Ejercicio: Películas en DVD (IV)

```
public void vender(PeliculaEnVenta aVender) throws PeliculaNoEncontrada{
   boolean encontrada = false;
   int i = 0;
   while ( (i< vPeliculas.length) && (!encontrada) ) {
          if (vPeliculas[i].equals(aVender)) encontrada= true;
          else i++;
   }
   if(!encontrada)  throw new PeliculaNoEncontrada("No está " + aVender);
   vPeliculas[i].decrementarCopiasDisponibles();
   if (vPeliculas[i].copiasDisponibles == 0) borrarPelicula(vPeliculas[i]);
 }
}
```


Solución Animaladas (I)

```
public class Animal implements Comparable<Animal>{
    int edad;
    public Animal(int edad) {this.edad = edad;}
    public int compareTo(Animal a){ return this.edad - a.edad;  }
}
```


Solución Animaladas (II)

```
public class Mamifero extends Animal {
    public Mamifero(int edad){super(edad);}
}
```

```
public class Leon extends Mamifero {
  public Leon(int edad){super(edad);}
}
```

```
public class Cocodrilo extends Animal{
    public Cocodrilo(int edad){super(edad);}
}
```

Solución Animaladas (III)

```
public class Jaula<E extends Mamifero>{
   public E animal;
   public void encerrar(E animal){
       this.animal = animal;
    }
}
```

```
public static void main(String args[]){
     Jaula<Mamifero> j = new Jaula<Mamifero>();
     j.encerrar(new Leon(25)); }
```


Solución: Carnet de Conducir

```
public class CarnetDeConducir implements CarnetPorPuntos,
   Comparable<CarnetDeConducir>
{
 // Declaración de los atributos y métodos ya existentes
  public int compareTo(CarnetDeConducir c){
      return this.puntos - c.puntos;
  }
}
```