Ejercicios Tema 3: Genericidad

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

Ejercicio de Comparación

- Diseña e implementa la clase **Manzana** sabiendo que:
  - Tiene un atributo numérico entero llamado sabor que indica el nivel de sabor de la manzana en una escala de 0 a 10.
  - Una manzana cualquiera tiene inicialmente un valor medio de sabor, aunque el usuario puede decidir ese valor al crearla.
  - Se desea poder decidir si una manzana es mejor que otra en base a su sabor. Esta funcionalidad puede permitir, por ejemplo, crear un vector de manzanas y ordenarlas de menor a mayor sabor.

Cálculo del Mínimo en Array Genérico

- Diseña la clase Operaciones que incluye un método estático genérico para calcular el elemento mínimo de un array, independientemente de su tipo de datos.
- Diséñese una clase TestOperaciones que use la clase Operaciones para encontrar el mínimo en un array de Integer.

Ejercicio de Vehiculos (1/2)

- Se desea diseñar una aplicación Java que gestione Vehículos (concretamente, Coches y Motos). Estos son los principales requisitos:
  - Se desea saber el modelo y la potencia de cualquier Vehículo. También conocer el número de ruedas.
  - Además, para las Motos se desea saber si llevan carenado. Para los Coches, se quiere conocer si llevan elevalunas eléctrico.
- Por otra parte, se quiere conocer si un Vehículo es más potente que otro para, por ejemplo, poder ordenar un conjunto de Vehículos en base a su potencia, con librerías estándar de ordenación.

Ejercicio de Vehiculos (2/2)

- Se pide:
  - Realizar el diseño e implementación completo de todas las clases de la aplicación para satisfacer los requisitos.
  - Es conveniente emplear los mecanismos estudiados a lo largo del tema 1 para realizar un buen diseño de clases.
- Posteriormente, crear una clase de prueba que:
  - Construya un vector de 10 Motos "Honda CBR 900 RR" con una potencia de 152 caballos y una pequeña desviación entre [0,1] caballo.
  - Ordene el vector asumiendo que está disponible el método ordenacionDirecta de la clase Ordenacion visto en teoría.

Garaje de Vehículos

- Implementa una clase Garaje, que únicamente permita almacenar Vehiculos, utilizando un ArrayList
- Métodos de la clase Garaje:
  - aparca: Recibe un vehículo y lo guarda en el array
  - retira: Elimina el vehículo del array. Lanza la excepción *VehiculoInexistente* si no estaba en el garaje.
- Utiliza para ello los siguientes métodos de la clase ArrayList:
  - add: Añade un objeto al ArrayList
  - remove: Elimina un objeto del ArrayList. Devuelve true si el elemento estaba y false en caso contrario.
- Construye un método de prueba que utilice un Garaje.

Ejercicio: Películas en DVD

- Dada la siguiente clase Java:

```
public class PeliculaEnDvd {
   protected String titulo;
   protected String director;
   protected int anyo;

public PeliculaEnDvd(String titulo, String director, int anyo) {
   this.titulo = titulo;
   this.director = director;
   this.anyo = anyo;
}

public String toString () {
 return "Titulo:" + titulo + " Director:" + director + " Año:" + anyo;
}
```


Ejercicio: Películas en DVD (II)

1. Diseñar la clase Java **PeliculaEnVenta**, que representa cualquier película disponible para la venta que además del título, director y año disponga información sobre:
   - Precio de la película.
   - Número de copias disponibles para la venta.
   - Se desea que las PeliculaEnVenta puedan ser ordenadas de manera creciente según el año de estreno y, para el mismo año, según el orden alfabético del título.
   - Se desea poder consultar el número de copias disponibles para su venta.
   - Se desea poder decrementar el número de copias disponibles para su venta.

Ejercicio: Películas en DVD (III)

2. Diseñar la clase GrupoDePeliculas, que utiliza un vector de *PeliculaEnVenta*.
   - Diseñar *toString()* para que obtenga un listado **ordenado** de todas las películas disponibles .
     - La ordenación se realizará utilizando el siguiente método de la clase Ordenacion:
       public static <T extends Comparable<T>> void *insercionDirecta(T [] a);*
   - Diseñar *void vender(PeliculaEnVenta aVender) throws PeliculaNoEncontrada.*
     - Si la película está disponible para la venta, actualiza el número de copias disponibles para la venta.
     - Si era la última copia disponible, la película se borra mediante el método que se asume existe: *borrarPelicula(PeliculaEnVenta p);*
     - Si la película no está disponible, se lanza la excepción PeliculaNoEncontrada (se asume ya implementada).

Animaladas (I)

- Se desea diseñar una aplicación sobre Animales, Mamiferos, Leones, Cocodrilos, donde se pretende crear una Jaula para albergar Mamíferos.
- Interesa poder saber si un animal es más viejo que otro.
- Se pide corregir los errores en las siguientes clases:

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


Animaladas (II)

```
public class Mamifero extends Animal {
    public Mamifero(){}
}
```

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

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


Animaladas (III)

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

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

Carnet de Conducir

- Dadas las siguiente clases:

```
public interface CarnetPorPuntos { void quitarPuntos ( int penalizacion ); }
```

```
public class CarnetDeConducir implements CarnetPorPuntos  {
  protected String nombre;    protected int puntos;
  public CarnetDeConducir(String nombre ) {this.nombre=nombre; puntos = 12;}
  public String toString( )  {  return nombre + "  (" + puntos + " puntos" + ")"; }
  public final String getNombre( ) { return nombre; }
  public final void quitarPuntos ( int penalizacion ) { puntos -= penalizacion; };
}
```

- Modificar CarnetDeConducir para que sea posible comparar dos carnets en base a sus puntos.