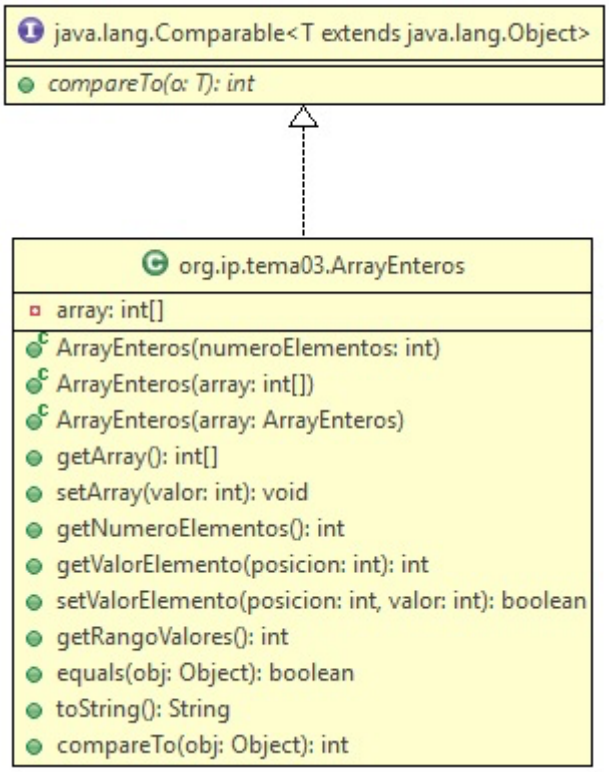
Ejercicios Resueltos Tema 03.

Ejemplo 1: En este ejemplo veremos cómo implementar la clase, **`ArrayEnteros`** (el dato miembro es un *array* de enteros (`int [] array`) y además se implementan funciones básicas para su manejo), cuyo diagrama de clases se proporciona a continuación:



Aclaraciones importantes:

1. Se implementarán 3 constructores diferentes según los parámetros que se le pasen. Destacar que para el constructor `public ArrayEnteros(int numeroElementos)` se creará el `array` y se le asignará a cada elemento el valor de su posición en el `array` (`array[i] = i`).

2. El método `public void setArray(int valor)` le asignará a todos los elementos del `array` el mismo valor, y éste será el que se le pasará como parámetro a través de la variable valor.

3. El método `public int getValorElemento(int posicion)` devolverá el valor que tiene el `array` en la posición (`posicion`), sabiendo que $0 \leq posicion \leq numeroElementos - 1$.

4. El método `public boolean setValorElemento(int posicion, int valor)` devolverá true si ha podido realizar correctamente la asignación del valor en la posición (`posicion`) del `array`, false en caso contrario. Todo en el rango, $0 \leq posicion \leq numeroElementos - 1$.

5. El método `public boolean equals(Object obj)` devuelve `true` si los dos arrays son exactamente iguales en contenido, `false` en caso contrario.

6. La salida del método `public String toString()` debe ser la salida que devuelve cuando aplicamos el método `toString()` de la clase `Arrays`.

7. Se utilizará el valor del rango de valores (`public int getRangoValores()`) de todos los elementos que hay almacenados en el array como criterio para la comparación de `arrays (public int compareTo(Object obj))`. El rango de valores, `getRangoValores()`, se obtiene restando al mayor valor del array el valor del menor elemento del array (mayor – menor).

Solución:

```
package org.ip.tema03;

public class ArrayEnteros implements Comparable<Object> {

   private int [] array;

   public ArrayEnteros(int numeroElementos) {
      array = new int [numeroElementos];
      for (int i = 0; i < array.length; i++) {
         array[i] = i;
      }
   }

   public ArrayEnteros(int[] array) {
      this.array = new int [array.length];
      for (int i = 0; i < array.length; i++) {
         this.array[i] = array[i];
      }
   }

   public ArrayEnteros(ArrayEnteros array) {
      this.array = new int [array.getNumeroElementos()];
      for (int i = 0; i < array.getNumeroElementos(); i++) {
         this.array[i] = array.getValorElemento(i);
      }
   }

   public int [] getArray() {
      return array;
   }

   public void setArray(int valor) {
      for (int i = 0; i < array.length; i++) {
         array[i] = valor;
      }
   }
```

```
   public int getNumeroElementos() {
      return array.length;
   }

   public int getValorElemento(int posicion) {
      if ((posicion < 0) || (posicion > array.length - 1)) throw new RuntimeException("Valor
de posicion fuera de rango");
      return array[posicion];
   }

   public boolean setValorElemento(int posicion, int valor) {
      if ((posicion < 0) || (posicion > array.length - 1))
         return false;
      array[posicion] = valor;
      return true;
   }

   public int getRangoValores() {
      int minValue = Integer.MAX_VALUE;
      int maxValue = Integer.MIN_VALUE;
      for (int i = 0; i < array.length; i++) {
         if (array[i] < minValue)
            minValue = array[i];
         if (array[i] > maxValue)
            maxValue = array[i];
      }
      return maxValue - minValue;
   }

   @Override
   public boolean equals(Object obj) {
      if (this == obj)
         return true;
      if (obj == null)
         return false;
      if (getClass() != obj.getClass())
         return false;
      if (!(obj instanceof ArrayEnteros))
         return false;
      ArrayEnteros other = (ArrayEnteros) obj;

      if (array.length != other.getNumeroElementos())
         return false;
      boolean equal = true;
      for (int i = 0; (i < array.length) && (equal); i++) {
         if (array[i] != other.getValorElemento(i))
            equal = false;
      }
      return equal;
   }
```

```
   @Override
   public String toString() {
      String salida = "Array: [";
      for (int i = 0; i < array.length; i++) {
         if (i != array.length - 1)
            salida += array[i] + "\t";
         else
            salida += array[i];
      }
      salida += "]";
      return salida;
   }

   @Override
   public int compareTo(Object obj) {
      ArrayEnteros otraMatriz = (ArrayEnteros) obj;
      if (this.getRangoValores() > otraMatriz.getRangoValores()){
         return 1;
      }
      else if (this.getRangoValores() < otraMatriz.getRangoValores()){
         return -1;
      }
      else
         return 0;
   }

}
```

TestArrayEnteros (Ejemplo de ejecución para la clase `ArrayEnteros`)

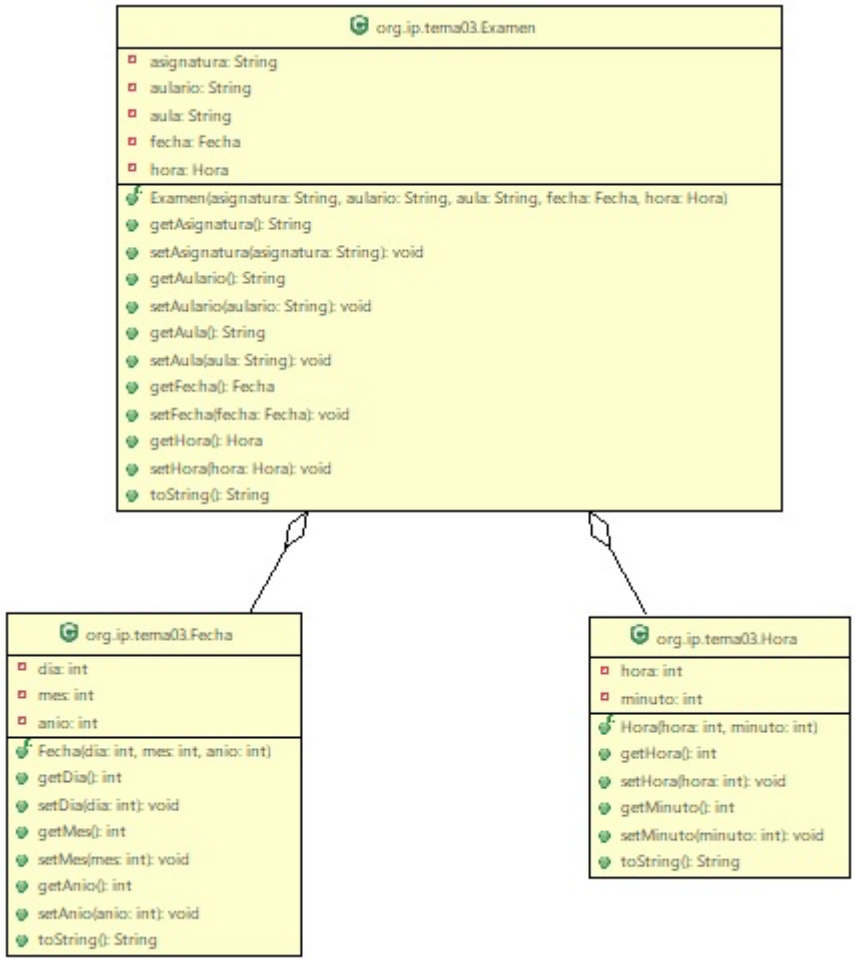
Se recomienda ejecutarlo y deducir su funcionamiento

```
package org.ip.tema03;

public class TestArrayEnteros {

   public static void main(String[] args) {

      int[] arrayExamen = {5, 10, -15, 2, -4, 23, 7, -13, 8, 17, 27};
      int[] otroArrayExamen = {20, 6, -30, 8, 1, 30, -2};

      ArrayEnteros objetoArray1 = new ArrayEnteros(arrayExamen);
      ArrayEnteros objetoArray2 = new ArrayEnteros(otroArrayExamen);
      ArrayEnteros objetoArray3 = new ArrayEnteros(11);

      System.out.println("Array 1: " + objetoArray1.toString());

      System.out.println("Array 2: " + objetoArray2.toString());

      System.out.println("Array 3: " + objetoArray3.toString());
```

```
     System.out.println("Numero de elementos array 1: " +
objetoArray1.getNumeroElementos());
     System.out.println("Valor de la posicion 2 del array 2: " +
objetoArray2.getValorElemento(2));
     System.out.println("Valor de la posicion 7 del array 3: " +
objetoArray3.getValorElemento(7));
     objetoArray3.setValorElemento(7, 0);
     System.out.println("Valor de la posicion 7 del array 3 (modificada): " +
objetoArray3.getValorElemento(7));

     System.out.println("Rango de valores del array 1: " + objetoArray1.getRangoValores());
     System.out.println("Rango de valores del array 2: " + objetoArray2.getRangoValores());
     System.out.println("Rango de valores del array 3: " + objetoArray3.getRangoValores());

     objetoArray3.setValorElemento(3, 0);
     objetoArray3.setValorElemento(9, 1);
     objetoArray3.setValorElemento(5, 1);
     objetoArray3.setValorElemento(10, 2);

     System.out.println("Nuevo Array 3: " + objetoArray3.toString());

     System.out.println("Rango de valores del array 3: " + objetoArray3.getRangoValores());

     if (objetoArray1.equals(objetoArray2))
        System.out.println("Los arrays 1 y 2 son iguales.");
     else
        System.out.println("Los arrays 1 y 2 no son iguales.");

     if (objetoArray2.compareTo(objetoArray1) == 1)
        System.out.println("El array 2 es mayor que el array 1 segun el rango de valores.");
     else if (objetoArray2.compareTo(objetoArray1) == -1)
        System.out.println("El array 1 es mayor que el array 2 segun el rango de valores.");
     else
        System.out.println("Los arrays 1 y 2 son iguales segun el rango de valores.");

     if (objetoArray3.compareTo(objetoArray1) == 1)
        System.out.println("El array 3 es mayor que el array 1 segun el rango de valores.");
     else if (objetoArray3.compareTo(objetoArray1) == -1)
        System.out.println("El array 1 es mayor que el array 3 segun el rango de valores.");
     else
        System.out.println("Los arrays 1 y 3 son iguales segun el rango de valores.");
```

```
      ArrayEnteros objetoArray4 = new ArrayEnteros(objetoArray2);
      System.out.println("Array 4: " + objetoArray4.toString());

      if (objetoArray2.equals(objetoArray4))
         System.out.println("Los arrays 2 y 4 son iguales.");
      else
         System.out.println("Los arrays 2 y 4 no son iguales.");
      if (objetoArray4.compareTo(objetoArray2) == 1)
         System.out.println("El array 4 es mayor que el array 2 segun el rango de valores.");
      else if (objetoArray4.compareTo(objetoArray2) == -1)
         System.out.println("El array 2 es mayor que el array 4 segun el rango de valores.");
      else
         System.out.println("Los arrays 2 y 4 son iguales segun el rango de valores.");

   }
}
```

Ejemplo 2: En este ejemplo veremos el uso de la **composición de clases**, en el que la clase `Examen` está compuesta, entre otros atributos, por un objeto de la clase `Fecha` y otro objeto de la clase `Hora`, tal y como se indica en el diagrama de clases que se proporciona a continuación



Solución:

```
package org.ip.tema03;

public class Fecha {

   private int dia;
   private int mes;
   private int anio;

   public Fecha(int dia, int mes, int anio) {
      this.dia = dia;
      this.mes = mes;
      this.anio = anio;
   }
```

```
   public int getDia() {
      return dia;
   }

   public void setDia(int dia) {
      this.dia = dia;
   }

   public int getMes() {
      return mes;
   }

   public void setMes(int mes) {
      this.mes = mes;
   }

   public int getAnio() {
      return anio;
   }

   public void setAnio(int anio) {
      this.anio = anio;
   }

   @Override
   public String toString() {
      return dia + "/" + mes + "/" + anio;
   }

}
```

```
package org.ip.tema03;

public class Hora {

   private int hora;
   private int minuto;

   public Hora(int hora, int minuto) {
      if ((hora < 0) || (hora > 23) || (minuto < 0) || (minuto > 59)) {
         throw new IllegalArgumentException();
      }
      else {
         this.hora = hora;
         this.minuto = minuto;
      }
   }

   public int getHora() {
      return hora;
   }

   public void setHora(int hora) {
      this.hora = hora;
   }
```

```
   public int getMinuto() {
      return minuto;
   }

   public void setMinuto(int minuto) {
      this.minuto = minuto;
   }

   @Override
   public String toString() {
      return (hora < 10 ? "0" : "") + hora + ":" + (minuto < 10 ? "0" : "") + minuto;
   }

}
```

```
package org.ip.tema03;

public class Examen {

   private String asignatura;
   private String aulario;
   private String aula;
   private Fecha fecha;
   private Hora hora;

   public Examen(String asignatura, String aulario, String aula, Fecha fecha, Hora hora) {
      this.asignatura = asignatura;
      this.aulario = aulario;
      this.aula = aula;
      this.fecha = fecha;
      this.hora = hora;
   }

   public String getAsignatura() {
      return asignatura;
   }

   public void setAsignatura(String asignatura) {
      this.asignatura = asignatura;
   }

   public String getAulario() {
      return aulario;
   }

   public void setAulario(String aulario) {
      this.aulario = aulario;
   }

   public String getAula() {
      return aula;
   }
```

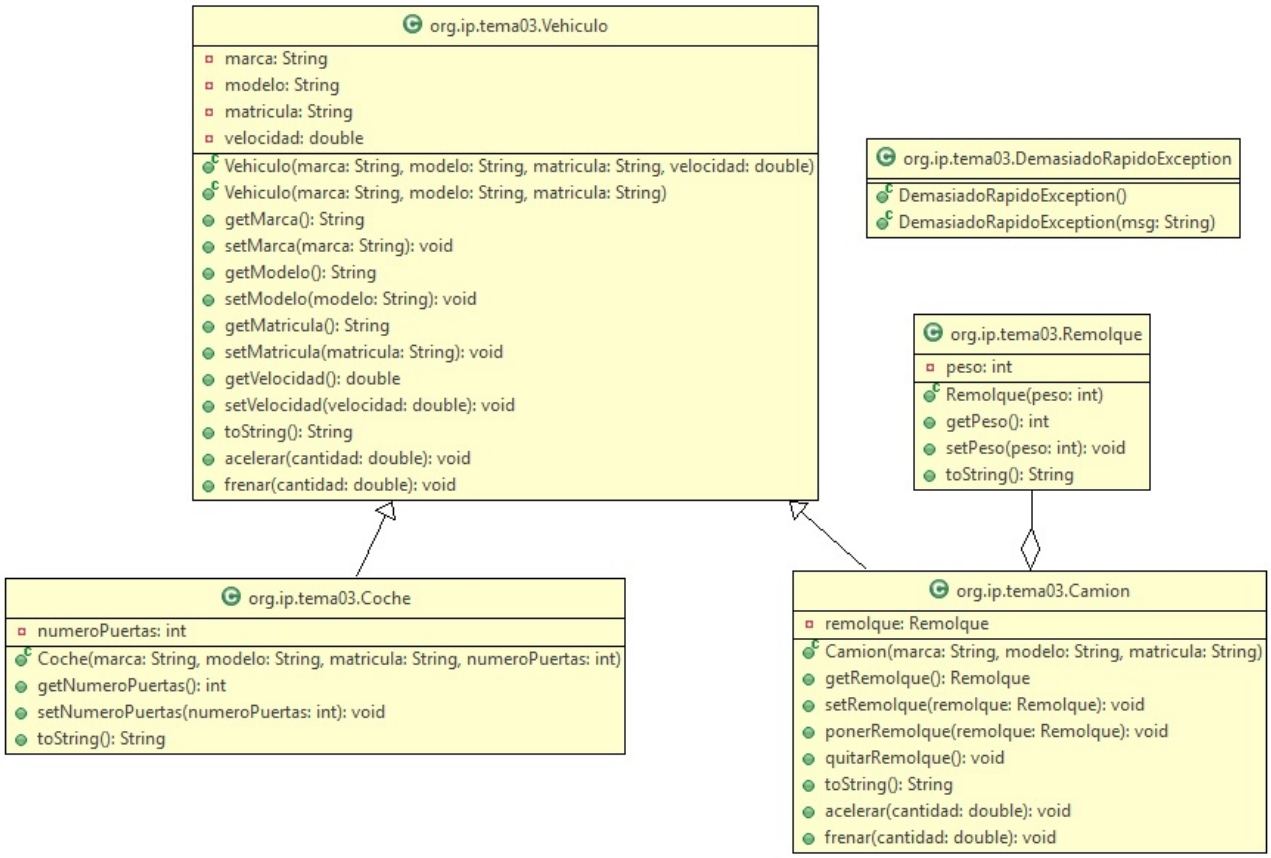
```
   public void setAula(String aula) {
      this.aula = aula;
   }

   public Fecha getFecha() {
      return fecha;
   }

   public void setFecha(Fecha fecha) {
      this.fecha = fecha;
   }

   public Hora getHora() {
      return hora;
   }

   public void setHora(Hora hora) {
      this.hora = hora;
   }

   @Override
   public String toString() {
      return "Examen" + "\n" +
            "Asignatura: " + asignatura + "\n" +
            "Aulario: " + aulario + "\n" +
            "Aula: " + aula + "\n" +
            "Fecha: " + fecha.toString() + "\n" +
            "Hora: " + hora.toString() + "\n" ;
   }

}
```

TestExamen (Ejemplo de ejecución para la clase `Examen`)

Se recomienda ejecutarlo y deducir su funcionamiento

```
package org.ip.tema03;

public class TestExamen {

   public static void main(String[] args) {
      Fecha fecha = new Fecha(4, 2, 2017);
      Hora hora = new Hora(9, 0);
      Examen iP = new Examen("Introducion a la Programacion", "Aulario III", "Aula 7",
                             fecha, hora);
      System.out.println(iP.toString());
      iP.setFecha(new Fecha(7, 2, 2017));
      iP.setHora(new Hora(12, 0));
      System.out.println(iP.toString());
   }
}
```

Ejemplo 3: En este ejemplo veremos el uso de la **herencia de clases**, en el que la clase base es `Vehiculo` y sus derivadas `Coche` y `Camion` (**compuesta** por un objeto de la clase `Remolque`), tal y como se indica en el diagrama de clases que se proporciona a continuación.
También se incluye una clase `DemasiadoRapidoException` que deriva de `Exception` y personaliza la excepción para cuando se excede el límite de velocidad de un determinado vehículo.



Destacar que cuando se hereda una clase base, en la clase derivada (hija) se pueden acceder a las variables y funciones miembros que permiten los especificadores de visibilidad (`public` y `protected`). Para heredar se utiliza la palabra reservada `extends`

Solución:

```java
package org.ip.tema03;

public class Vehiculo {

   private String marca;
   private String modelo;
   private String matricula;
   private double velocidad;

   public Vehiculo(String marca, String modelo, String matricula, double velocidad) {
      this.marca = marca;
      this.modelo = modelo;
      this.matricula = matricula;
      this.velocidad = velocidad;
   }

   public Vehiculo(String marca, String modelo, String matricula) {
      this.marca = marca;
      this.modelo = modelo;
      this.matricula = matricula;
      this.velocidad = 0.0;
   }

   public String getMarca() {
      return marca;
   }

   public void setMarca(String marca) {
      this.marca = marca;
   }

   public String getModelo() {
      return modelo;
   }

   public void setModelo(String modelo) {
      this.modelo = modelo;
   }

   public String getMatricula() {
      return matricula;
   }

   public void setMatricula(String matricula) {
      this.matricula = matricula;
   }

   public double getVelocidad() {
      return velocidad;
   }
```

```
   public void setVelocidad(double velocidad) {
      this.velocidad = velocidad;
   }

   @Override
   public String toString() {
      return "El vehiculo con marca:" + marca + ", modelo:" + modelo + " y matricula:"
           + matricula + " va a una velocidad de:" + velocidad + " km/h";
   }

   public void acelerar(double cantidad) throws DemasiadoRapidoException {
      if ((velocidad + cantidad) > 120)
         throw new DemasiadoRapidoException("Limite de velocidad sobrepasado por el
vehiculo");
      velocidad += cantidad;
   }

   public void frenar(double cantidad) throws DemasiadoRapidoException {
      if ((velocidad - cantidad) < 0)
         throw new DemasiadoRapidoException("El vehiculo se para repentinamente");
      velocidad -= cantidad;
   }
}


package org.ip.tema03;

public class Coche extends Vehiculo {

   private int numeroPuertas;

   public Coche(String marca, String modelo, String matricula, int numeroPuertas) {
      super(marca, modelo, matricula);
      this.numeroPuertas = numeroPuertas;
   }

   public int getNumeroPuertas() {
      return numeroPuertas;
   }

   public void setNumeroPuertas(int numeroPuertas) {
      this.numeroPuertas = numeroPuertas;
   }

   @Override
   public String toString() {
      return "Coche => " + super.toString() + ". Tiene " + numeroPuertas + " puertas";
   }
}
```

```java
package org.ip.tema03;

public class Remolque {

   private int peso;

   public Remolque(int peso) {
      this.peso = peso;
   }

   public int getPeso() {
      return peso;
   }

   public void setPeso(int peso) {
      this.peso = peso;
   }

   @Override
   public String toString() {
      return "Remolque con un peso de " + peso + " Kgr";
   }
}
```

```java
package org.ip.tema03;

public class Camion extends Vehiculo {

   private Remolque remolque;

   public Camion(String marca, String modelo, String matricula) {
      super(marca, modelo, matricula);
      this.remolque = null;
   }

   public Remolque getRemolque() {
      return remolque;
   }

   public void setRemolque(Remolque remolque) {
      this.remolque = remolque;
   }

   public void ponerRemolque(Remolque remolque) {
      setRemolque(remolque);
   }

   public void quitarRemolque() {
      this.remolque = null;
   }
```

```
   @Override
   public String toString() {
      if (remolque != null)
         return "Camion => " + super.toString() + ". Lleva un " + remolque.toString();
      else
         return "Camion => " + super.toString();
   }

   @Override
   public void acelerar(double cantidad) throws DemasiadoRapidoException {
      if ((remolque != null) && ((cantidad + getVelocidad()) > 100))
         throw new DemasiadoRapidoException("Camion demasiado rapido");
      super.acelerar(cantidad);
   }

   @Override
   public void frenar(double cantidad) throws DemasiadoRapidoException {
      if ((remolque != null) && ((cantidad - getVelocidad()) < 0))
         throw new DemasiadoRapidoException("Camion se para repentinamente");
      super.frenar(cantidad);
   }
}
```

```
package org.ip.tema03;

public class DemasiadoRapidoException extends Exception {

   public DemasiadoRapidoException() {

   }

   public DemasiadoRapidoException(String msg) {
      super(msg);
   }
}
```

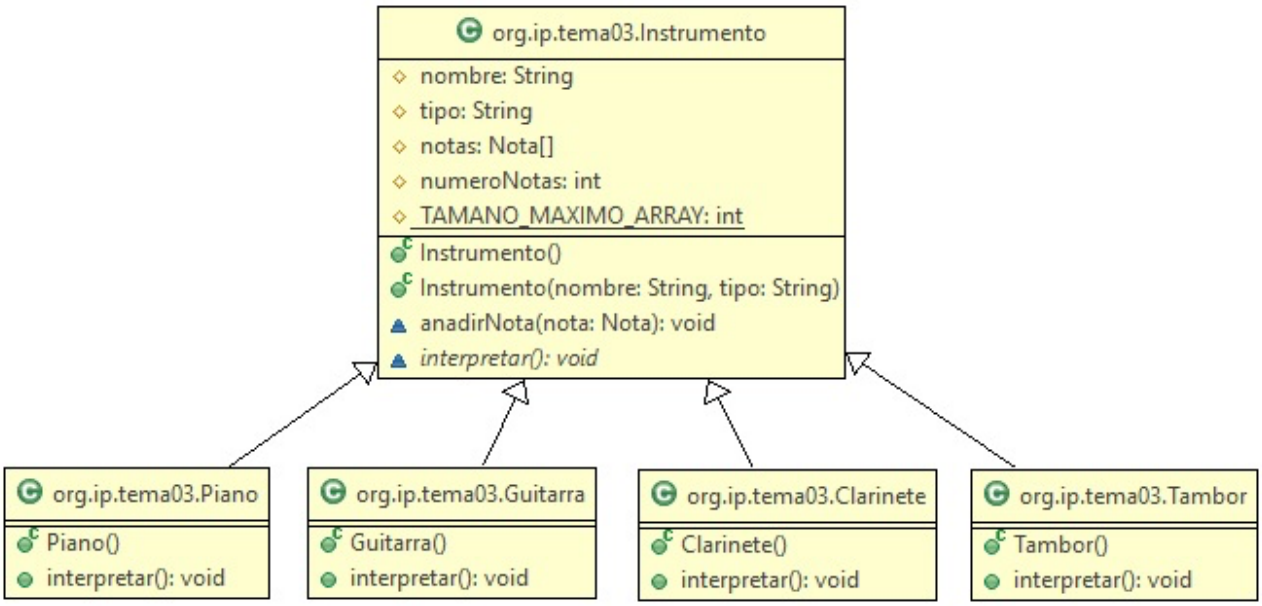
TestVehiculos (Ejemplo de ejecución para la clase `Vehiculo` y sus clases derivadas)

Se recomienda ejecutarlo y deducir su funcionamiento

```
package org.ip.tema03;

public class TestVehiculos {

   public static void main(String[] args) {

      Vehiculo [] arrayVehiculos = new Vehiculo[5];

      arrayVehiculos[0] = new Coche("Ford", "Focus", "1346FKC", 5);
      arrayVehiculos[1] = new Camion("Volvo", "FL337", "7371GHX");
      arrayVehiculos[2] = new Coche("Mazda", "CX7", "9075HVT", 5);
      arrayVehiculos[3] = new Camion("Iveco", "190E31", "1378FXZ");
      arrayVehiculos[4] = new Coche("Renault", "Clio", "4443FBC", 3);

      for (int i = 0; i < arrayVehiculos.length; i++) {
         if (arrayVehiculos[i] instanceof Camion) {
            ((Camion)arrayVehiculos[i]).ponerRemolque(new Remolque(20000));
         }
      }

      for (int i = 0; i < arrayVehiculos.length; i++) {
         try {
            arrayVehiculos[i].acelerar(110);
         }
         catch (DemasiadoRapidoException e) {
            System.out.println("Algun vehiculo ha excedido su limite de velocidad");
         }
      }

      for (int i = 0; i < arrayVehiculos.length; i++) {
         try {
            arrayVehiculos[i].frenar(30);
         }
         catch (DemasiadoRapidoException e) {
            System.out.println("Algun vehiculo se va a parar repentinamente");
         }
      }

      for (int i = 0; i < arrayVehiculos.length; i++) {
         System.out.println(arrayVehiculos[i].toString());
      }
   }
}
```

Ejemplo 4: Repasando las **clases abstractas** sabemos que, una clase abstracta no puede ser instanciada (no se pueden crear objetos directamente - `new`), solo puede ser heredada. Al menos un método de la clase es `abstract`, esto obliga a que la clase completa sea definida como `abstract`, sin embargo la clase puede tener el resto de métodos no abstractos. Los métodos `abstract` no llevan cuerpo (no llevan los caracteres `{}`). La primera subclase concreta que herede de una clase `abstract` debe implementar todos los métodos de la superclase.

Para ver un ejemplo, tenemos la clase `abstract Instrumento` que almacena una serie de atributos como el nombre, tipo y máximo de (100) notas musicales. Mientras haya espacio se podrá añadir nuevas notas musicales, al final de un *array* denominado `notas`, con el método `anadirNota()`. La clase también dispone de un método abstracto `interpretar()` que hay que implementar en cada clase que herede de la clase `Instrumento`, mostrará por pantalla las notas musicales según las interprete. Además habrá que añadir cuatro clases que deriven de la clase `Instrumento`, como son `Clarinete`, `Guitarra`, `Piano` y `Tambor`.



Solución:

```
package org.ip.tema03;

public abstract class Instrumento {

   public enum Nota {DO, RE, MI, FA, SOL, LA, SI}

   protected String nombre;
   protected String tipo;
   protected Nota notas[];
   protected int numeroNotas;
   static protected int TAMANO_MAXIMO_ARRAY = 100;

   public Instrumento() {
      this.nombre = "";
      this.tipo = "";
      this.notas = new Nota[TAMANO_MAXIMO_ARRAY];
      this.numeroNotas = 0;
   }

   public Instrumento(String nombre, String tipo) {
      this.nombre = nombre;
      this.tipo = tipo;
      this.notas = new Nota[TAMANO_MAXIMO_ARRAY];
      this.numeroNotas = 0;
   }

   void anadirNota(Nota nota) {
      if (numeroNotas < notas.length) {
         notas[numeroNotas] = nota;
         numeroNotas++;
      }
   }

   // metodo abstracto
   abstract void interpretar();
}
```

```
package org.ip.tema03;

public class Clarinete extends Instrumento {

   public Clarinete() {
      super("Clarinete", "Viento");
   }
```

```
   @Override
   public void interpretar() {
      System.out.println(nombre + " - " + tipo);
      for (int i = 0; i < numeroNotas; i++) {
         switch (notas[i]) {
         case DO:
            System.out.print("DoooooC ");
            break;
         case RE:
            System.out.print("ReeeeeC ");
            break;
         case MI:
            System.out.print("MiiiiiC ");
            break;
         case FA:
            System.out.print("FaaaaaC ");
            break;
         case SOL:
            System.out.print("SooooolC ");
            break;
         case LA:
            System.out.print("LaaaaaC ");
            break;
         case SI:
            System.out.print("SiiiiiC ");
            break;
         }
      }
      System.out.println("");
   }
}
```

```
package org.ip.tema03;

public class Guitarra extends Instrumento {

   public Guitarra() {
      super("Guitarra", "Cuerda");
   }

   @Override
   public void interpretar() {
      System.out.println(nombre + " - " + tipo);
      for (int i = 0; i < numeroNotas; i++) {
         switch (notas[i]) {
         case DO:
            System.out.print("DoooG ");
            break;
         case RE:
            System.out.print("ReeeG ");
            break;
```

```
            case MI:
               System.out.print("MiiiG ");
               break;
            case FA:
               System.out.print("FaaaG ");
               break;
            case SOL:
               System.out.print("SooolG ");
               break;
            case LA:
               System.out.print("LaaaG ");
               break;
            case SI:
               System.out.print("SiiiG ");
               break;
            }
         }
         System.out.println("");
      }
   }
```

```
package org.ip.tema03;

public class Piano extends Instrumento {

   public Piano() {
      super("Piano", "Teclado y cuerdas");
   }

   @Override
   public void interpretar() {
      System.out.println(nombre + " - " + tipo);
      for (int i = 0; i < numeroNotas; i++) {
         switch (notas[i]) {
         case DO:
            System.out.print("Do ");
            break;
         case RE:
            System.out.print("Re ");
            break;
         case MI:
            System.out.print("Mi ");
            break;
         case FA:
            System.out.print("Fa ");
            break;
         case SOL:
            System.out.print("Sol ");
            break;
```

```
            case LA:
               System.out.print("La ");
               break;
            case SI:
               System.out.print("Si ");
               break;
            }
         }
         System.out.println("");
      }
}
```

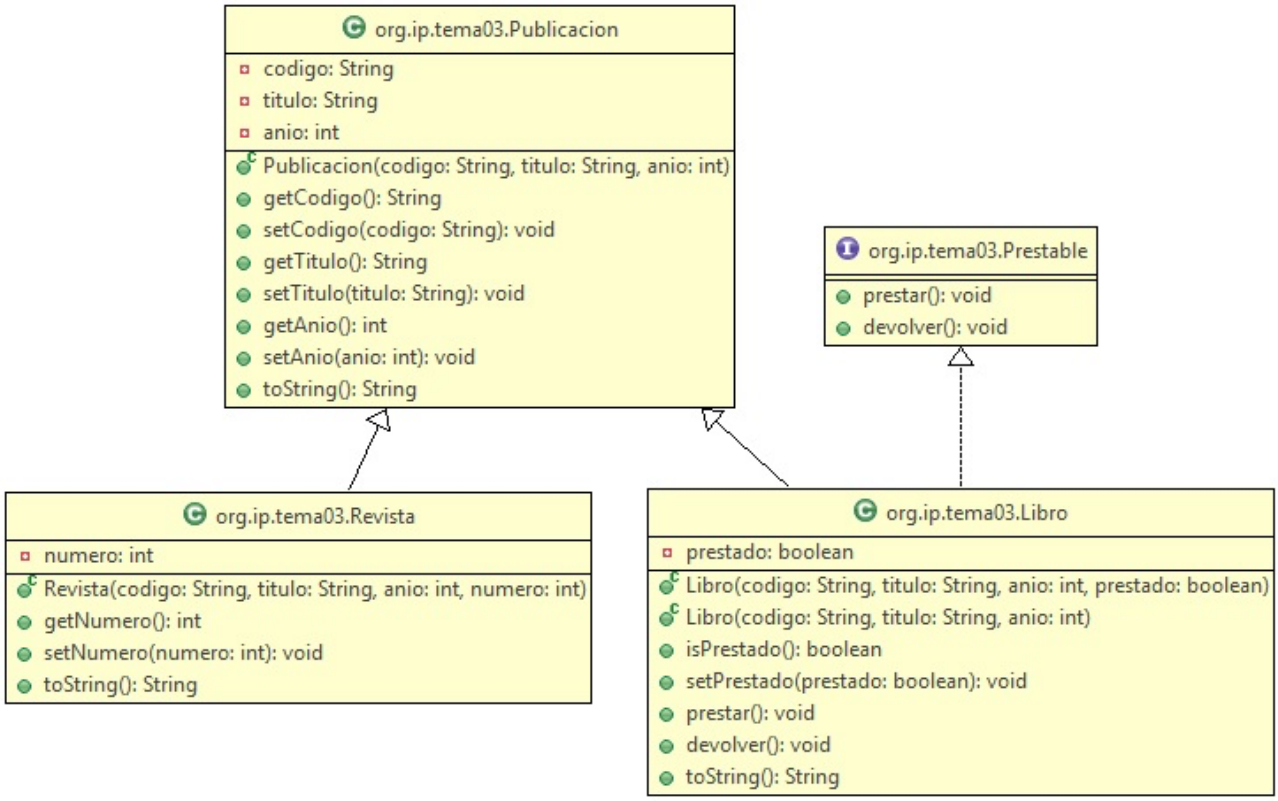
```
package org.ip.tema03;

public class Tambor extends Instrumento {

   public Tambor() {
      super("Tambor", "Percusion");
   }

   @Override
   public void interpretar() {
      System.out.println(nombre + " - " + tipo);
      for (int i = 0; i < numeroNotas; i++) {
         switch (notas[i]) {
         case DO:
            System.out.print("DoT ");
            break;
         case RE:
            System.out.print("ReT ");
            break;
         case MI:
            System.out.print("MiT ");
            break;
         case FA:
            System.out.print("FaT ");
            break;
         case SOL:
            System.out.print("SolT ");
            break;
         case LA:
            System.out.print("LaT ");
            break;
         case SI:
            System.out.print("SiT ");
            break;
         }
      }
      System.out.println("");
   }
}
```

TestInstrumentos (Ejemplo de ejecución para la clase `Instrumento` y sus clases derivadas)

Se recomienda ejecutarlo y deducir su funcionamiento

```
package org.ip.tema03;

public class TestInstrumentos {

   public static void main(String[] args) {
      Guitarra guitarra = new Guitarra();
      guitarra.anadirNota(Instrumento.Nota.DO);
      guitarra.anadirNota(Instrumento.Nota.SI);
      guitarra.anadirNota(Instrumento.Nota.SOL);
      guitarra.anadirNota(Instrumento.Nota.RE);
      guitarra.interpretar();
      System.out.println();

      Piano piano = new Piano();
      piano.anadirNota(Instrumento.Nota.DO);
      piano.anadirNota(Instrumento.Nota.SI);
      piano.anadirNota(Instrumento.Nota.MI);
      piano.anadirNota(Instrumento.Nota.FA);
      piano.anadirNota(Instrumento.Nota.DO);
      piano.interpretar();
      System.out.println();

      Clarinete clarinete = new Clarinete();
      clarinete.anadirNota(Instrumento.Nota.FA);
      clarinete.anadirNota(Instrumento.Nota.LA);
      clarinete.anadirNota(Instrumento.Nota.DO);
      clarinete.anadirNota(Instrumento.Nota.FA);
      clarinete.anadirNota(Instrumento.Nota.DO);
      clarinete.anadirNota(Instrumento.Nota.LA);
      clarinete.anadirNota(Instrumento.Nota.FA);
      clarinete.anadirNota(Instrumento.Nota.LA);
      clarinete.anadirNota(Instrumento.Nota.DO);
      clarinete.anadirNota(Instrumento.Nota.FA);
      clarinete.interpretar();
      System.out.println();

      Tambor tambor = new Tambor();
      tambor.anadirNota(Instrumento.Nota.SI);
      tambor.anadirNota(Instrumento.Nota.LA);
      tambor.anadirNota(Instrumento.Nota.SI);
      tambor.anadirNota(Instrumento.Nota.LA);
      tambor.anadirNota(Instrumento.Nota.MI);
      tambor.interpretar();
      System.out.println();
   }
}
```

Ejemplo 5: En este ejemplo veremos el uso de la **herencia de clases** y de las **interfaces** en Java. Tenemos una jerarquía de clases donde las clases `Revista` y `Libro` extienden de la clase `Publicacion`, además la clase `Libro` implementa la interface `Prestable`. Todo ello tal y como se indica en el diagrama de clases se proporciona a continuación.



Solución:

```java
package org.ip.tema03;

public class Publicacion {

   private String codigo;
   private String titulo;
   private int anio;

   public Publicacion(String codigo, String titulo, int anio) {
      this.codigo = codigo;
      this.titulo = titulo;
      this.anio = anio;
   }

   public String getCodigo() {
      return codigo;
   }

   public void setCodigo(String codigo) {
      this.codigo = codigo;
   }

   public String getTitulo() {
      return titulo;
   }

   public void setTitulo(String titulo) {
      this.titulo = titulo;
   }

   public int getAnio() {
      return anio;
   }

   public void setAnio(int anio) {
      this.anio = anio;
   }

   @Override
   public String toString() {
      return "Codigo: " + codigo + "\n"
           + "Titulo: " + titulo + "\n"
           + "Anio de publicacion: " + anio + "\n";
   }
}
```

```java
package org.ip.tema03;

public class Revista extends Publicacion {

   private int numero;

   public Revista(String codigo, String titulo, int anio, int numero) {
      super(codigo, titulo, anio);
      this.numero = numero;
   }

   public int getNumero() {
      return numero;
   }

   public void setNumero(int numero) {
      this.numero = numero;
   }

   @Override
   public String toString() {
      return super.toString() + "Numero: " + numero + "\n";
   }
}
```

```java
package org.ip.tema03;

public interface Prestable {

   public void prestar();
   public void devolver();

}
```

```java
package org.ip.tema03;

public class Libro extends Publicacion implements Prestable {

   private boolean prestado;

   public Libro(String codigo, String titulo, int anio, boolean prestado) {
      super(codigo, titulo, anio);
      this.prestado = prestado;
   }

   public Libro(String codigo, String titulo, int anio) {
      super(codigo, titulo, anio);
      this.prestado = false;
   }
```

```
   public boolean isPrestado() {
      return prestado;
   }

   public void setPrestado(boolean prestado) {
      this.prestado = prestado;
   }

   public void prestar() {
      prestado = true;
   }

   public void devolver() {
      prestado = false;
   }

   @Override
   public String toString() {
      return super.toString() + (prestado ? "prestado" : "no prestado") + "\n";
   }
}
```

TestPublicaciones (Ejemplo de ejecución para la clase `Publicacion` y sus clases derivadas). En este ejemplo se puede comprobar el uso del polimorfismo, pues en tiempo de ejecución se puede detectar el tipo de objeto de la jerarquía de clases que hay en el *array* de `Publicacion` y en base a eso proceder de una forma o de otra.

Se recomienda ejecutarlo y deducir su funcionamiento

```
package org.ip.tema03;

public class TestPublicaciones {

   public static int numeroPrestados(Object [] array) {
      int contador = 0;
      for (int i = 0; i < array.length; i++) {
         if ((array[i] instanceof Libro) && ((Libro)array[i]).isPrestado())
            contador++;
      }
      return contador;
   }

   public static int publicacionesAnterioresA(Publicacion [] arrayPublicaciones, int anio) {
      int contador = 0;
      for (int i = 0; i < arrayPublicaciones.length; i++) {
         if ((arrayPublicaciones[i] instanceof Publicacion) &&
((Publicacion)arrayPublicaciones[i]).getAnio() < anio)
            contador++;
      }
      return contador;
   }
```

```
   public static void main(String[] args) {

      Publicacion [] biblioteca = {new Libro("000001L", "Introduction to Java Programming",
2015),
            new Revista("000001R", "GeoInformatica", 2017, 75),
            new Libro("000002L", "Machine Learning with Java", 2017),
            new Libro("000003L", "Data Structures and Algorithms in Java", 2014),
            new Revista("000002R", "Information Systems", 2017, 87),
            new Revista("000003R", "Parallel Databases", 2017, 51),
            };

      ((Libro)biblioteca[0]).prestar();
      ((Libro)biblioteca[3]).prestar();
      for (int i = 0; i < biblioteca.length; i++) {
         System.out.println(biblioteca[i]);
      }

      int anio = 2016;
      System.out.println(publicacionesAnterioresA(biblioteca, anio) + " publicaciones
anteriores a " + anio);

      System.out.println(numeroPrestados(biblioteca) + " libros prestados");
   }
}
```