



Activad UT5.1

Programación

Alfonso García Jorge

1ºDAWNA

1.- Implementa las siguientes clases:

- clase *Nota*. Una nota contiene un identificador numérico y una línea de texto. Define constructor, accedentes, mutadores y `toString`.

```
1 public class Nota {  
2     static int cid = 1;  
3     private String nota;  
4  
5     private final int id;  
6  
7     public Nota(String nota) {  
8         this.nota = nota;  
9         this.id = cid;  
10        cid++;  
11    }  
12  
13    public String getNota() { return nota; }  
14  
15    public void setNota(String nota) { this.nota = nota; }  
16  
17    public static int getCid() { return cid; }  
18  
19    public static void setCid(int cid) { Nota.cid = cid; }  
20  
21    public int getId() {  
22        return id;  
23    }  
24  
25    @Override  
26    public String toString() {  
27        return "Nota{" +  
28            "nota='" + nota + '\'' +  
29            ", id=" + id +  
30            '}';  
31    }  
32 }  
33
```

- clase `NotaAlarma`. Una nota que además contiene la hora en la que sonará la alarma. Define constructor, accedentes, mutadores y `toString`.

```
1
2 2 usages
3 public class NotaAlarma extends Nota{
4     4 usages
5     private String hora;
6
7     1 usage
8     public NotaAlarma(String hora, String Nota) {
9         super(Nota);
10        this.hora = hora;
11    }
12
13    no usages
14    public String getHora() { return hora; }
15
16    no usages
17    public String setHora(String hora) {
18        this.hora = hora;
19        return hora;
20    }
21
22    @Override
23    public String toString() {
24        return "NotaAlarma{" +
25            "hora='" + hora + '\'' +
26            ", nota='" + getNota() + '\'' +
27            ", id=" + getId() + '}';
28    }
29 }
```

- clase *BlocNotas* que modela un bloc de notas en el que se pueden introducir notas, listar todas las notas, eliminar una nota mediante su posición en el bloc de notas o saber cuántas notas contiene el bloc de notas. Utiliza un Array.

```
1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 public class BlocNotas {
5     Scanner sc = new Scanner(System.in);
6     ArrayList<NotaAlarma> misNotas;
7
8     public BlocNotas(){
9         misNotas = new ArrayList<>();
10    }
11
12    public void introducirNota(){
13        System.out.println("Escribe una hora: ");
14        String hora = sc.nextLine();
15        System.out.println("Escribe una nota: ");
16        String nota = sc.nextLine();
17        misNotas.add(new NotaAlarma(hora,nota));
18    }
19
20    public void listarNotas(){
21        for (Nota nota : this.misNotas) {
22            System.out.println(nota.toString());
23        }
24    }
25
26    public void eliminarNotas(){
27        System.out.println("Que nota quieres eliminar: ");
28        int opcion = sc.nextInt();sc.nextLine();
29        misNotas.removeIf(nota -> (nota.getId() == opcion));
30        listarNotas();
31    }
32 }
```

- clase Prueba que cree un bloc de Notas de ejemplo y pruebe las operaciones que soporta. .

```

1  import java.util.Scanner;
2  import java.util.ArrayList;
3
4  public class Prueba {
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7
8          BlocNotas misBlocNotas = new BlocNotas();
9
10         boolean condicion = true;
11         while (condicion){
12             misBlocNotas.introducirNota();
13             System.out.println("Quieres continuar? si/no");
14             String respuesta = sc.nextLine();
15             if(respuesta.equals("no")){
16                 condicion = false;
17             }
18         }
19         misBlocNotas.listarNotas();
20         misBlocNotas.eliminarNotas();
21     }
22 }

```

```

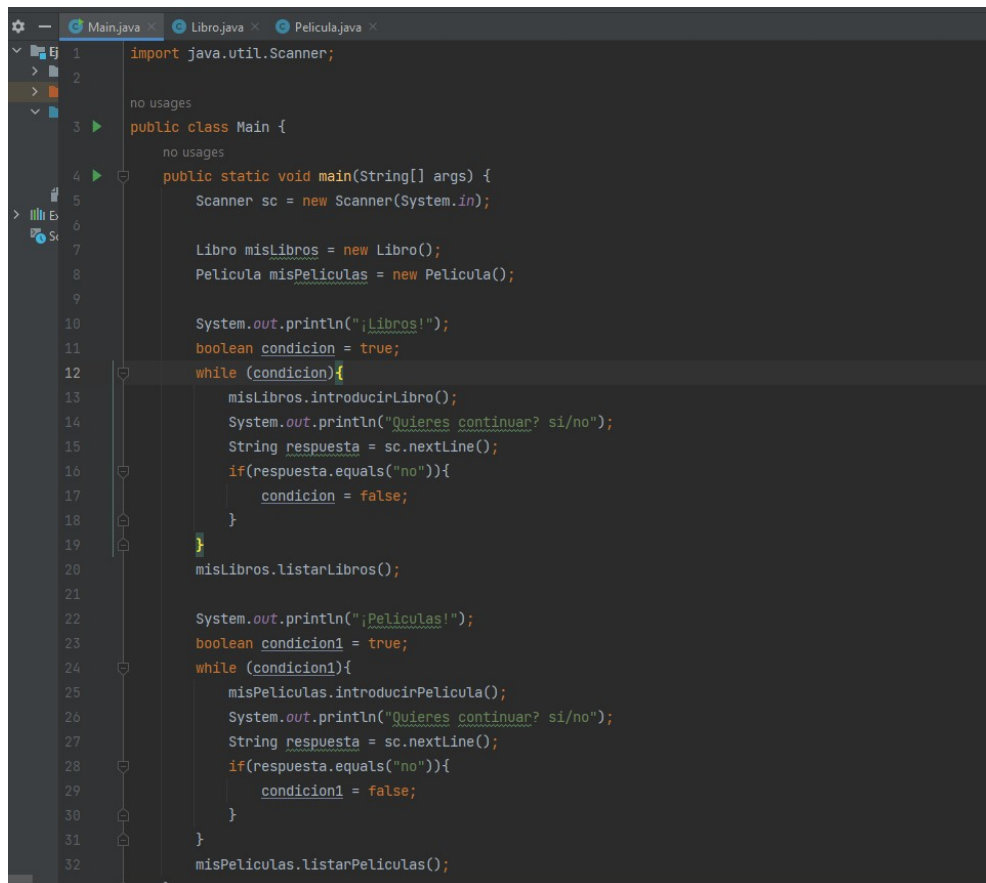
C:\Users\alfon\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea_rt.jar=5050:C:\Program Files\JetBrains\IntelliJ IDEA\bin" -Dfile.encoding=UTF-8
Escribe una hora:
13:00
Escribe una nota:
llamar medico
Quieres continuar? si/no
si
Escribe una hora:
17:00
Escribe una nota:
Gym
Quieres continuar? si/no
si
Escribe una hora:
20:00
Escribe una nota:
cena
Quieres continuar? si/no
no
NotaAlarma{hora='13:00', nota='llamar medico', id=1}
NotaAlarma{hora='17:00', nota='Gym', id=2}
NotaAlarma{hora='20:00', nota='cena', id=3}
Que nota quieres eliminar:
2
NotaAlarma{hora='13:00', nota='llamar medico', id=1}
NotaAlarma{hora='20:00', nota='cena', id=3}

Process finished with exit code 0

```

2.- Una persona desea tener una tienda virtual para vender libros y películas en formato blu-ray. Para esto es necesario, primeramente, representar con un enfoque orientado a objetos, los tipos de productos que se van a vender. Inicialmente, se están considerando las dos clases siguientes:

- Libro, que tiene como atributos el autor y el titulo de tipo String, y el precio de tipo Float.
- Pelicula, cuyos atributos son el título, el protagonista y el director de tipo String, y el precio de tipo Float.
- Las dos clases contienen sus respectivos constructores, métodos getters y setters y el método toString().
- Implementa las clases Libro y Pelicula considerando los atributos, constructores y métodos descritos. Además, escribe una clase principal para crear varios Libros y varias Películas, y saber su título, autor, protagonista, director y precio, según corresponda.



```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         Libro misLibros = new Libro();
8         Pelicula misPeliculas = new Pelicula();
9
10        System.out.println("Libros!");
11        boolean condicion = true;
12        while (condicion){
13            misLibros.introducirLibro();
14            System.out.println("Quieres continuar? si/no");
15            String respuesta = sc.nextLine();
16            if(respuesta.equals("no")){
17                condicion = false;
18            }
19        }
20        misLibros.listarLibros();
21
22        System.out.println("Películas!");
23        boolean condicion1 = true;
24        while (condicion1){
25            misPeliculas.introducirPelicula();
26            System.out.println("Quieres continuar? si/no");
27            String respuesta = sc.nextLine();
28            if(respuesta.equals("no")){
29                condicion1 = false;
30            }
31        }
32        misPeliculas.listarPeliculas();
33    }
34 }
```

```

1  import java.util.ArrayList;
2  import java.util.Scanner;
3
4  5 usages
5  public class Libro {
6      4 usages
7      Scanner sc = new Scanner(System.in);
8      3 usages
9      ArrayList<Libro> libros;
10
11      4 usages
12      private String autor;
13      4 usages
14      private String titulo;
15      4 usages
16      private float precio;
17
18      1 usage
19      public Libro(){
20          libros = new ArrayList<>();
21      }
22
23      1 usage
24      public Libro(String autor, String titulo, float precio) {
25          this.autor = autor;
26          this.titulo = titulo;
27          this.precio = precio;
28      }
29
30      no usages
31      public String getAutor() {
32          return autor;
33      }
34
35      no usages
36      public void setAutor(String autor) {
37          this.autor = autor;
38      }

```

```

39      no usages
40      public String getTitulo() {
41          return titulo;
42      }
43
44      no usages
45      public void setTitulo(String titulo) {
46          this.titulo = titulo;
47      }
48
49      no usages
50      public float getPrecio() {
51          return precio;
52      }
53
54      no usages
55      public void setPrecio(float precio) {
56          this.precio = precio;
57      }
58
59      @Override
60      public String toString() {
61          return "Libro{" +
62              "autor=" + autor + '\n' +
63              ", titulo=" + titulo + '\n' +
64              ", precio=" + precio +
65              '}';
66      }
67
68      1 usage
69      public void introducirLibro(){
70          System.out.println("Escribe un autor: ");
71          String autor = sc.nextLine();
72          System.out.println("Escribe un titulo: ");
73          String titulo = sc.nextLine();
74          System.out.println("Escribe un precio: ");
75          float precio = sc.nextFloat();sc.nextLine();
76          libros.add(new Libro(autor,titulo,precio));
77      }

```

```

78
79      1 usage
80      public void listarLibros(){
81          for (Libro libro : this.libros) {
82              System.out.println(libro.toString());
83          }
84      }

```



```

1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 public class Pelicula{
5     Scanner sc = new Scanner(System.in);
6     ArrayList<Pelicula> peliculas;
7
8     private String titulo;
9     private String protagonista;
10    private String director;
11    private float precio;
12
13    public Pelicula() { peliculas = new ArrayList<>(); }
14
15    public Pelicula(String titulo, String protagonista, String director, float precio) {
16        this.titulo = titulo;
17        this.protagonista = protagonista;
18        this.director = director;
19        this.precio = precio;
20    }
21
22    public String getTitulo() { return titulo; }
23
24    public void setTitulo(String titulo) { this.titulo = titulo; }
25
26
27
28
29
30
31

```

```

8 public void setTitulo(String titulo) { this.titulo = titulo; }
9
10 public String getProtagonista() { return protagonista; }
11
12 public void setProtagonista(String protagonista) { this.protagonista = protagonista; }
13
14 public String getDirector() { return director; }
15
16 public void setDirector(String director) { this.director = director; }
17
18 public float getPrecio() { return precio; }
19
20 public void setPrecio(float precio) { this.precio = precio; }
21
22 @Override
23 public String toString() {
24     return "Pelicula{" +
25         "titulo=" + titulo + '\'' +
26         ", protagonista=" + protagonista + '\'' +
27         ", director=" + director + '\'' +
28         ", precio=" + precio +
29         '\'' +
30     }
31 }

```

```

65
66 public void introducirPelicula(){
67     System.out.println("Escribe un titulo: ");
68     String titulo = sc.nextLine();
69     System.out.println("Escribe un protagonista: ");
70     String protagonista = sc.nextLine();
71     System.out.println("Escribe un director: ");
72     String director = sc.nextLine();
73     System.out.println("Escribe un precio: ");
74     float precio = sc.nextFloat();sc.nextLine();
75     peliculas.add(new Pelicula(titulo,protagonista,director,precio));
76 }
77
78 public void listarPeliculas(){
79     for (Pelicula pelicula : this.peliculas) {
80         System.out.println(pelicula.toString());
81     }
82 }
83
84 }

```

```

C:\Users\alfon\jdk\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\Je
;Libros!
Escribe un autor:
Juan ramirez
Escribe un titulo:
Adios
Escribe un precio:
12
Quieres continuar? si/no
si
Escribe un autor:
Pedro
Escribe un titulo:
Renacido
Escribe un precio:
11
Quieres continuar? si/no
no
Libro{autor='Juan ramirez', titulo='Adios', precio=12.0}
Libro{autor='Pedro', titulo='Renacido', precio=11.0}
;Peliculas!
Escribe un titulo:
Ayer
Escribe un protagonista:
jhon
Escribe un director:
mikel
Escribe un precio:
29
Quieres continuar? si/no
no
Pelicula{titulo='Ayer', protagonista='jhon', director='mikel', precio=29.0}

Process finished with exit code 0

```


3.- En el ejercicio anterior, el atributo autor de Libro y los atributos protagonista y director de Película se definieron como de tipo String. Dado que estos atributos contienen el nombre de una persona, esta definición puede no ser homogénea. Por lo tanto, un mejor modelo sería considerar la clase adicional:

- Persona, cuyos atributos son nombre y apellido, de tipo String.

Esta clase, además de contener sus respectivos métodos getters y setters, considera el método:

- esIgual(Persona p), que devuelve true si el nombre y apellido de un objeto Persona son iguales a las de la persona p y false en caso contrario.

De esta forma, las clases Libro y Pelicula se tienen que redefinir como sigue:

- Libro, cuyos atributos son autor de tipo Persona, titulo de tipo String y precio de tipo Float.
- Pelicula, con los atributos protagonista y director de tipo Persona, titulo de tipo String y precio de tipo Float.

Implementa la clase Persona y la redefinición de las clases Libro y Pelicula, considerando los atributos y métodos descritos.

```
16 usages
public class Persona {
    5 usages
    private String nombre;
    5 usages
    private String apellido;

    3 usages
    public Persona(String nombre, String apellido) {
        this.nombre = nombre;
        this.apellido = apellido;
    }

    1 usage
    public String getNombre() {
        return nombre;
    }

    no usages
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    1 usage
    public String getApellido() {
        return apellido;
    }

    no usages
    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
}

26 @Override
27 public String toString() {
28     return "Persona{" +
29         "nombre='" + nombre + '\'' +
30         ", apellido='" + apellido + '\'' +
31         '}';
32 }
33
34 @
35 public boolean igual(Persona persona){
36     return this.nombre.equals(persona.getNombre()) && this.apellido.equals(persona.getApellido());
37 }
```

```
ArrayList<Libro> libros;
```

4 usages

4 usages

```
private Persona autor;
```

```
private String titulo;
```

4 usages

4 usages

```
private String titulo;
```

```
private Persona protagonista;
```

4 usages

4 usages

```
private float precio;
```

```
private Persona director;
```

4 usages

```
private float precio;
```

```
Run: Main x
juan
Escribe el apellido del autor:
pedro
Escribe un titulo:
renacido
Escribe un precio:
13
Quieres continuar? si/no
si
Escribe el nombre del autor:
pedro
Escribe el apellido del autor:
jose
Escribe un titulo:
adios
Escribe un precio:
78
Quieres continuar? si/no
no
Libro{autor='Persona{nombre='juan', apellido='pedro'}', titulo='renacido', precio=13.0}
Libro{autor='Persona{nombre='pedro', apellido='jose'}', titulo='adios', precio=78.0}
¡Peliculas!
Escribe un titulo:
Ayer
Escribe un nombre del protagonista:
jhon
Escribe el apellido del protagonista:
mikel
Escribe el nombre del director:
cameron
Escribe el apellido del director:
dias
Escribe un precio:
34
Quieres continuar? si/no
no
Pelicula{titulo='Ayer', protagonista='Persona{nombre='jhon', apellido='mikel'}', director='Persona{nombre='cameron', apellido='dias'}', precio=34.0}

Process finished with exit code 0
```

4.- Vamos a generalizar las clases Libro y Pelicula del ejercicio anterior.

Como nos podemos dar cuenta, las clases Libro y Pelicula tienen dos atributos en común: titulo y precio. Entonces, podemos considerar una generalización de estas dos clases para crear la superclase:

- Producto, cuyos atributos son el titulo y el precio de un producto, de tipo String y Float, respectivamente. Generalmente los productos están relacionados con un identificador único, por lo tanto esta clase también tiene el atributo id de tipo Integer. Crear producto como una clase abstracta.

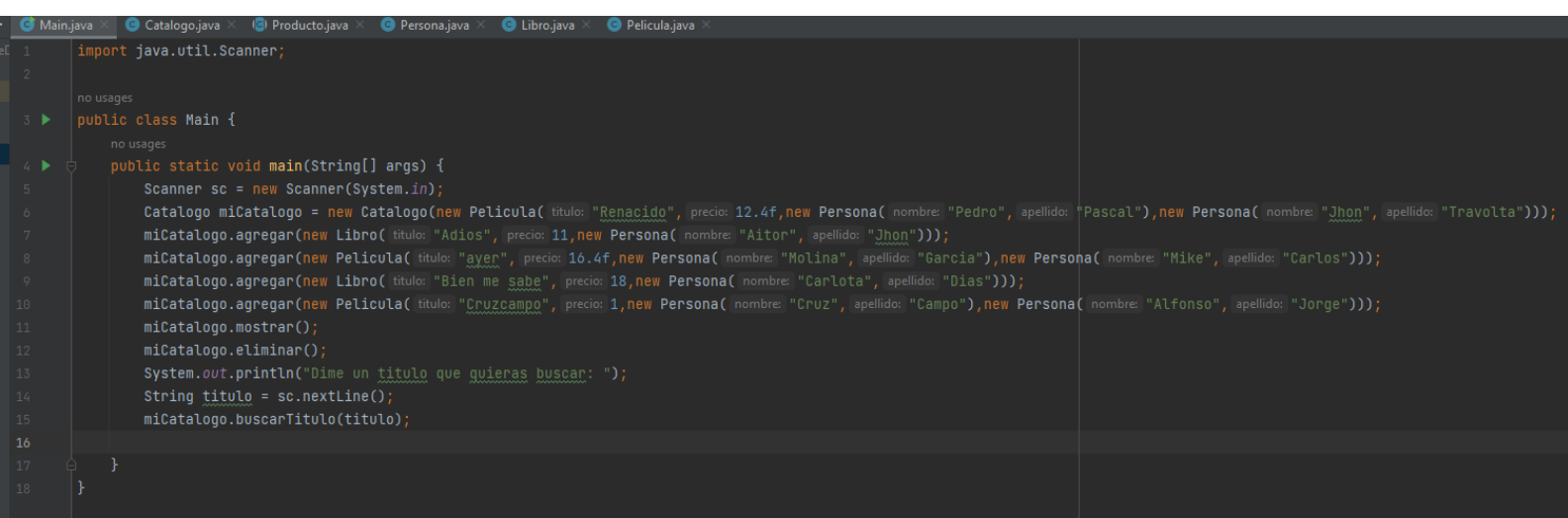
De esta forma, ahora las clases Libro y Pelicula son subclases de Producto y deben heredar de ésta sus atributos y métodos. Además, dado que los productos se van a vender, es necesario contar con un catálogo que los clientes puedan revisar. Por esta razón, se considera la clase:

- Catalogo, que cuenta con un atributo productos, que contiene a los libros y películas en venta (por ejemplo, un arreglo suficientemente grande de Productos), y con otro para conocer la cantidad de productos que están disponibles.

Todas las clases anteriores consideran sus respectivos constructores, métodos getters y setters y el método toString(). Adicionalmente, la clase Catalogo considera los siguientes métodos:

- agregar(Producto p), agrega el Producto p al catálogo.
- eliminar(Integer id), elimina el producto cuyo identificador único es id.
- buscar(String titulo), devuelve, contenidos en un Catalogo, a todos los Productos cuyo título es titulo.
- buscar(Persona p), devuelve, contenidos en un Catalogo, a todos los Productos cuyo autor, director o protagonista, según sea el caso, es p.

Implementa las clases Persona, Libro y Catalogo, considerando los atributos, constructores y métodos descritos.



```
1 import java.util.Scanner;
2
3 no usages
4 public class Main {
5     no usages
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         Catalogo miCatalogo = new Catalogo(new Pelicula( titulo: "Renacido", precio: 12.4f, new Persona( nombre: "Pedro", apellido: "Pascal"), new Persona( nombre: "Jhon", apellido: "Travolta")));
9         miCatalogo.agregar(new Libro( titulo: "Adios", precio: 11, new Persona( nombre: "Aitor", apellido: "Jhon")));
10        miCatalogo.agregar(new Pelicula( titulo: "Ayer", precio: 16.4f, new Persona( nombre: "Molina", apellido: "Garcia"), new Persona( nombre: "Mike", apellido: "Carlos")));
11        miCatalogo.agregar(new Libro( titulo: "Bien me sabe", precio: 18, new Persona( nombre: "Carlota", apellido: "Dias")));
12        miCatalogo.agregar(new Pelicula( titulo: "Cruzcampo", precio: 1, new Persona( nombre: "Cruz", apellido: "Campo"), new Persona( nombre: "Alfonso", apellido: "Jorge")));
13        miCatalogo.mostrar();
14        miCatalogo.eliminar();
15        System.out.println("Dime un título que quieras buscar: ");
16        String titulo = sc.nextLine();
17        miCatalogo.buscarTitulo(titulo);
18    }
19 }
```

```

1  import java.util.ArrayList;
2  import java.util.Scanner;
3
4  2 usages
   public class Catalogo {
5       2 usages
       Scanner sc = new Scanner(System.in);
6
7       7 usages
       private ArrayList<Producto> productos;
8       1 usage
       public Catalogo(Producto producto){
9           productos = new ArrayList<>();
10          productos.add(producto);
11      }
12
13      1 usage
       public ArrayList<Producto> getProductos() { return productos; }
14
15      4 usages
       public void agregar(Producto producto){
16          productos.add(producto);
17      }
18
19      2 usages
       public void mostrar(){
20          for (int i = 0; i < productos.size(); i++) {
21              System.out.println(productos.get(i).toString());
22          }
23      }
24
25      1 usage
       public void eliminar(){
26          System.out.println("Que Id quiere eliminar?");
27          int opcion = sc.nextInt();sc.nextLine();
28          productos.removeIf(producto -> producto.getId() == opcion);
29          mostrar();
30      }
31

```

```

1 usage
   public void buscarTitulo(String titulo){
       for (Producto producto : getProductos()) {
           if(producto instanceof Libro){
               Libro libro = (Libro) producto;
               if (libro.getTitulo().equals(titulo)){
                   System.out.println("Titulo: " + libro.getTitulo() + ", autor: " + libro.getAutor() + ", ID: " + libro.getId());
               }
           }else{
               Pelicula pelicula = (Pelicula) producto;
               if (pelicula.getTitulo().equals(titulo)){
                   System.out.println("Titulo: " + pelicula.getTitulo() + ", protagonista: " + pelicula.getProtagonista() + ", ID: " + pelicula.getId());
               }
           }
       }
   }
}

```

```
8 usages 2 inheritors
1 public abstract class Producto {
2     4 usages
3     private String titulo;
4     4 usages
5     private float precio;
6     4 usages
7     static int cid = 1;
8     3 usages
9     private final int id;
10
11     2 usages
12     public Producto(String titulo, float precio) {
13         this.titulo = titulo;
14         this.precio = precio;
15         this.id = cid;
16         cid++;
17     }
18
19     6 usages
20     public String getTitulo() { return titulo; }
21
22     no usages
23     public void setTitulo(String titulo) { this.titulo = titulo; }
24
25     2 usages
26     public float getPrecio() { return precio; }
27
28     no usages
29     public void setPrecio(float precio) { this.precio = precio; }
30
31     no usages
32     public static int getCid() { return cid; }
33
34     no usages
35     public static void setCid(int cid) { Producto.cid = cid; }
36
37     5 usages
38     public int getId() {
39         return id;
40     }
```

```
21 usages
1 public class Persona {
2     5 usages
3     private String nombre;
4     5 usages
5     private String apellido;
6
7     8 usages
8     public Persona(String nombre, String apellido) {
9         this.nombre = nombre;
10        this.apellido = apellido;
11    }
12
13    1 usage
14    public String getNombre() {
15        return nombre;
16    }
17
18    no usages
19    public void setNombre(String nombre) {
20        this.nombre = nombre;
21    }
22
23    1 usage
24    public String getApellido() {
25        return apellido;
26    }
27
28    no usages
29    public void setApellido(String apellido) {
30        this.apellido = apellido;
31    }
32
33    @Override
34    public String toString() {
35        return "Persona{" +
36            "nombre='" + nombre + '\'' +
37            "apellido='" + apellido + '\'' +
38            '}';
39    }
```

```

1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 public class Libro extends Producto{
5     Scanner sc = new Scanner(System.in);
6     ArrayList<Libro> libros;
7
8     private Persona autor;
9
10
11     public Libro(String titulo, float precio, Persona autor) {
12         super(titulo, precio);
13         this.autor = autor;
14     }
15
16     public Persona getAutor() { return autor; }
17
18     public void setAutor(Persona autor) { this.autor = autor; }
19
20
21     @Override
22     public String toString() {
23         return "Libro{" +
24             "autor=" + autor.toString() + '\n' +
25             ", titulo=" + getTitulo() + '\n' +
26             ", precio=" + getPrecio() +
27             ", id=" + getId() +
28             '}'
29     }
30 }

```

```

1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 public class Pelicula extends Producto{
5     Scanner sc = new Scanner(System.in);
6     ArrayList<Pelicula> peliculas;
7
8     private Persona protagonista;
9     private Persona director;
10
11     public Pelicula(String titulo, float precio, Persona protagonista, Persona director) {
12         super(titulo, precio);
13         this.protagonista = protagonista;
14         this.director = director;
15     }
16
17     public Persona getProtagonista() { return protagonista; }
18
19     public void setProtagonista(Persona protagonista) { this.protagonista = protagonista; }
20
21     public Persona getDirector() { return director; }
22
23     public void setDirector(Persona director) { this.director = director; }
24
25     @Override
26     public String toString() {
27         return "Pelicula{" +
28             "protagonista=" + protagonista.toString() + '\n' +
29             "director=" + director.toString() + '\n' +
30             "titulo=" + getTitulo() + '\n' +
31             "precio=" + getPrecio() + '\n' +
32             "id=" + getId() + '\n' +
33             '}'
34     }
35 }

```



```
C:\Users\alfon\.jdk\openjdk-19.0.2\bin\java.exe --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\lib\idea_rt.jar=57528:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\bin\java.exe -jar C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\bin\java.exe
Película{titulo='Renacido', protagonista='Persona{nombre='Pedro', apellido='Pascal'}', director='Persona{nombre='Jhon', apellido='Travolta'}', precio=12.4, id=1}
Libro{autor='Persona{nombre='Aitor', apellido='Jhon'}', titulo='Adios', precio=11.0, id=2}
Película{titulo='ayer', protagonista='Persona{nombre='Molina', apellido='Garcia'}', director='Persona{nombre='Mike', apellido='Carlos'}', precio=16.4, id=3}
Libro{autor='Persona{nombre='Carlota', apellido='Dias'}', titulo='Bien me sabe', precio=18.0, id=4}
Película{titulo='Cruzcampo', protagonista='Persona{nombre='Cruz', apellido='Campo'}', director='Persona{nombre='Alfonso', apellido='Jorge'}', precio=1.0, id=5}
Que Id quiere eliminar?
3
Película{titulo='Renacido', protagonista='Persona{nombre='Pedro', apellido='Pascal'}', director='Persona{nombre='Jhon', apellido='Travolta'}', precio=12.4, id=1}
Película{titulo='ayer', protagonista='Persona{nombre='Molina', apellido='Garcia'}', director='Persona{nombre='Mike', apellido='Carlos'}', precio=16.4, id=3}
Libro{autor='Persona{nombre='Carlota', apellido='Dias'}', titulo='Bien me sabe', precio=18.0, id=4}
Película{titulo='Cruzcampo', protagonista='Persona{nombre='Cruz', apellido='Campo'}', director='Persona{nombre='Alfonso', apellido='Jorge'}', precio=1.0, id=5}
Dime un titulo que quieras buscar:
Renacido
Titulo: Renacido, protagonista: Persona{nombre='Pedro', apellido='Pascal'}, ID: 1

Process finished with exit code 0
```


5.- Elaborar una clase RACIONAL que modele los números racionales implementando al menos las operaciones de suma, resta, opuesto e inverso de un número racional a imitación de la suma o resta de los números reales o enteros.

6.- En un puerto se alquilan amarres para barcos de distinto tipo. Para cada ALQUILER se guarda el nombre y DNI del cliente, las fechas inicial y final de alquiler, la posición del amarre y el barco que lo ocupará.

Un BARCO se caracteriza por su matrícula, su eslora en metros y año de fabricación.

Un alquiler se calcula multiplicando el número de días de ocupación (incluyendo los días inicial y final) por un módulo función de cada barco (obtenido simplemente multiplicando por 10 los metros de eslora) y por un valor fijo (300 euros en la actualidad).

Sin embargo ahora se pretende diferenciar la información de algunos tipos de barcos:

- número de mástiles para veleros
- potencia en CV para embarcaciones deportivas a motor
- potencia en CV y número de camarotes para yates de lujo.

El módulo de los barcos de un tipo especial se obtiene como el módulo normal más:

- el número de mástiles para veleros
- la potencia en CV para embarcaciones deportivas a motor
- la potencia en CV más el número de camarotes para yates de lujo.

```
1 public class Alquiler {
2     private String nombre;
3     private String dni;
4     private int diaIni;
5     private int diaFin;
6     private String posicion;
7     private Barco barco;
8
9     public Alquiler(String nombre, String dni, int diaIni, int diaFin, String posicion, Barco barco) {
10         this.nombre = nombre;
11         this.dni = dni;
12         this.diaIni = diaIni;
13         this.diaFin = diaFin;
14         this.posicion = posicion;
15         this.barco = barco;
16     }
17
18     public String getNombre() {
19         return nombre;
20     }
21
22     public void setNombre(String nombre) {
23         this.nombre = nombre;
24     }
25
26     public String getDni() {
27         return dni;
28     }
29
30     public void setDni(String dni) {
31         this.dni = dni;
32     }
33
34     public int getDiaIni() {
35         return diaIni;
36     }
37
38     public void setDiaIni(int diaIni) {
39         this.diaIni = diaIni;
40     }
41
42     public int getDiaFin() {
43         return diaFin;
44     }
45
46     public void setDiaFin(int diaFin) {
47         this.diaFin = diaFin;
48     }
49
50     public String getPosicion() {
51         return posicion;
52     }
53
54     public void setPosicion(String posicion) {
55         this.posicion = posicion;
56     }
57
58     public Barco getBarco() {
59         return barco;
60     }
61
62     public void setBarco(Barco barco) {
63         this.barco = barco;
64     }
65
66     @Override
67     public String toString() {
68         return "Alquiler{" +
69             "nombre='" + nombre + '\'' +
70             ", dni='" + dni + '\'' +
71             ", dia=" + diaIni +
72             ", mes=" + diaFin +
73             ", posicion='" + posicion + '\'' +
74             ", barco=" + barco +
75             '}';
76     }
77 }
```

```
Alquiler.java Barco.java Main.java
7 usages
1 public class Barco {
2     private String matricula;
3     private int eslora;
4     private int yearFab;
5     private String tipo;
6     private int numMastiles;
7     private int CvDeport;
8     private int CvLujo;
9
10    3 usages
11    public Barco(String matricula, int eslora, int yearFab, String tipo) {
12        this.matricula = matricula;
13        this.eslora = eslora;
14        this.yearFab = yearFab;
15        this.tipo = tipo;
16    }
17
18    no usages
19    public String getMatricula() {
20        return matricula;
21    }
```

```
39 }
40
41 no usages
42 public String getTipo() {
43     return tipo;
44 }
45
46 no usages
47 public void setTipo(String tipo) {
48     this.tipo = tipo;
49 }
50
51 @Override
52 public String toString() {
53     return "Barco{" +
54         "matricula='" + matricula + '\'' +
55         ", eslora=" + eslora +
56         ", yearFab=" + yearFab +
57         ", tipo='" + tipo + '\'' +
58         '}';
59 }
```

```
1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 no usages
5 public class Main {
6     no usages
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9         ArrayList<Alquiler> alquileres = new ArrayList<>();
10         alquileres.add(new Alquiler( nombre: "Pedro", dni: "45354566", dialni: 12, diaFin: 2, posicion: "Parking 3",
11             new Barco( matricula: "444G", eslora: 54, yearFab: 2022, tipo: "Velero")));
12         alquileres.add(new Alquiler( nombre: "Alfonso", dni: "255444H", dialni: 7, diaFin: 6, posicion: "Parking 7",
13             new Barco( matricula: "888H", eslora: 78, yearFab: 2021, tipo: "Deportivo")));
14         alquileres.add(new Alquiler( nombre: "Ana", dni: "23453456", dialni: 23, diaFin: 5, posicion: "Parking 10",
15             new Barco( matricula: "4467I", eslora: 67, yearFab: 2020, tipo: "Lujo")));
16         for (Alquiler alquiler : alquileres) {
17             System.out.println(alquiler.toString());
18         }
19     }
20 }
```

Run: Main ×

▶

⬆

⬇

■

📷

🔍

🗑

C:\Users\alfon\jdk\openjdk-19.0.2\bin\java.exe

"-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\lib\idea_rt.jar=60329:C:\P

Alquiler{nombre='Pedro', dni='45354566', dia=12, mes=2, posicion='Parking 3', barco=Barco{matricula='444G', eslora=54, yearFab=2022, tipo='Velero'}}

Alquiler{nombre='Alfonso', dni='255444H', dia=7, mes=6, posicion='Parking 7', barco=Barco{matricula='888H', eslora=78, yearFab=2021, tipo='Deportivo'}}

Alquiler{nombre='Ana', dni='23453456', dia=23, mes=5, posicion='Parking 10', barco=Barco{matricula='467I', eslora=67, yearFab=2020, tipo='Lujo'}}

Process finished with exit code 0

7.- Codifica los siguientes diagramas de clase y ejecuta en la clase MAIN instanciando los objetos necesarios.

```
1 import java.util.Date;
2
3 no usages
4 public class Main implements ServMedico{
5     no usages
6     public static void main(String[] args) {
7         Alumno alumno = new Alumno( matricula: "547474H", carrera: "Derecho");
8         Docente docente = new Docente( rfc: "44F", new Date(), new CVitae( formacion: "Gestoria"));
9     }
10
11 no usages
12 @Override
13 public void regProveedor() {
14 }
15
16 no usages
17 @Override
18 public void regVigenciaServicio() {
19 }
20
21 no usages
22 @Override
23 public void regNSegS() {
24 }
25 }
```

```
1 1 usage 1 implementation
2 public interface ServMedico {
3     no usages 1 implementation
4     public void regProveedor();
5     no usages 1 implementation
6     public void regVigenciaServicio();
7     no usages 1 implementation
8     public void regNSegS();
9 }
10
11 }
```

```
Alumno.java x ServMedico.java x Main.java x Docente.java x CVitae.java x
2 usages
1 public class Alumno {
2     4 usages
3     private String matricula;
4     4 usages
5     private String carrera;
6
7     1 usage
8     public Alumno(String matricula, String carrera) {
9         this.matricula = matricula;
10        this.carrera = carrera;
11    }
12
13    no usages
14    public String getMatricula() {
15        return matricula;
16    }
17
18    no usages
19    public void setMatricula(String matricula) {
20        this.matricula = matricula;
21    }
22
23    no usages
24    public String getCarrera() {
25        return carrera;
26    }
27
28    no usages
29    public void setCarrera(String carrera) {
30        this.carrera = carrera;
31    }
32
33    @Override
34    public String toString() {
35        return "Alumno{" +
36            "matricula='" + matricula + '\'' +
37            ", carrera='" + carrera + '\'' +
38            '}';
39    }
40 }

import java.util.Date;
2 usages
1 public class Docente {
2     3 usages
3     private String rfc;
4     3 usages
5     private Date date;
6     3 usages
7     private CVitae experiencia;
8
9     1 usage
10    public Docente(String rfc, Date date, CVitae experiencia) {
11        this.rfc = rfc;
12        this.date = date;
13        this.experiencia = experiencia;
14    }
15
16    no usages
17    public String getRfc() {
18        return rfc;
19    }
20
21    no usages
22    public void setRfc(String rfc) {
23        this.rfc = rfc;
24    }
25
26    no usages
27    public Date getDate() {
28        return date;
29    }
30
31    no usages
32    public void setDate(Date date) {
33        this.date = date;
34    }
35
36    no usages
37    public CVitae getExperiencia() {
38        return experiencia;
39    }
40 }
```

```
Alumno.java x ServMedico.java x Main.java x Docente.java x CVitae.java x
5 usages
1 public class CVitae {
2     3 usages
3     private String formacion;
4
5     1 usage
6     public CVitae(String formacion) {
7         this.formacion = formacion;
8     }
9
10    no usages
11    public String getFormacion() {
12        return formacion;
13    }
14
15    no usages
16    public void setFormacion(String formacion) {
17        this.formacion = formacion;
18    }
19 }
```

```
Propietario.java Main.java Vehiculo.java regVehicular.java Automovil.java CveVehicular.java Modelo.java Marca.java
1 ▶ no usages
  public class Main implements regVehicular{
2 ▶   no usages
  public static void main(String[] args) {
3     Propietario propietario = new Propietario( rfc: "343454g", nombre: "Pedro", direccion: "Calle Torres", telefono: "6435663556");
4     Automovil automovil = new Automovil();
5     Vehiculo vehiculo = new Vehiculo();
6   }
7
8   no usages
9   @Override
10  public double pagoTramite() {
11      return 0;
12  }
13
14  no usages
15  @Override
16  public String obtenerCve() {
17      return null;
18  }
19 }
```

```
Propietario.java Main.java Vehiculo.java regVehicular.java Automovil.java CveVehicular.java
1 5 usages
  public class Propietario{
2    3 usages
    private String rfc;
3    3 usages
    private String nombre;
4    3 usages
    private String direccion;
5    3 usages
    private String telefono;
6
7    1 usage
    public Propietario(String rfc, String nombre, String direccion, String telefono) {
8        this.rfc = rfc;
9        this.nombre = nombre;
10       this.direccion = direccion;
11       this.telefono = telefono;
12   }
13
14  no usages
15  public String getRfc() {
16      return rfc;
17  }
18
19  no usages
20  public void setRfc(String rfc) {
21      this.rfc = rfc;
22  }
23
24  no usages
25  public String getNombre() {
26      return nombre;
27  }
28
29  no usages
30  public void setNombre(String nombre) {
31      this.nombre = nombre;
32  }
33
34  no usages
35  public String getDireccion() {
36  }
```

```
Vehiculo.java
1 2 usages
  public class Vehiculo {
2    2 usages
    private Marca marca;
3    2 usages
    private Modelo modelo;
4
5    2 usages
    private Propietario dueño;
6
7    2 usages
    private CveVehicular cveVehicular;
8
9    no usages
10   public Marca getMarca() {
11       return marca;
12   }
13
14   no usages
15   public void setMarca(Marca marca) {
16       this.marca = marca;
17   }
18
19   no usages
20   public Modelo getModelo() {
21       return modelo;
22   }
23
24   no usages
25   public void setModelo(Modelo modelo) {
26       this.modelo = modelo;
27   }
28
29   no usages
30   public Propietario getDueño() {
31       return dueño;
32   }
33
34   no usages
35   public void setDueño(Propietario dueño) {
36       this.dueño = dueño;
37   }
38 }
```

```
Propietario.java Main.java Vehiculo.java regVehicular.java Automovil.java
1 1 usage 1 implementation
  public interface regVehicular {
2    no usages 1 implementation
    double pagoTramite();
3    no usages 1 implementation
    String obtenerCve();
4  }
5 }
```