Programación con Java

TAREA 09

Alumno: Alejandro Rodríguez Jiménez
Profesor: Marc Esteve García
Curso 2024

Índice

Introducción

Con los programas que vamos a generar en esta tarea aprenderemos el uso de los modificadores en java: public todos los que comparte package lo podrán utilizar, con protected pasarán a sus herencias y con private solo él la utilizará. También veremos la creación de clases (para luego instanciarse como objetos), superclases y todo su árbol genealógico… Junto con las herencias de las superclases a sus hijas, aprenderemos las herencias de una interfaz hacia las clases que la implementen y el uso de “@Override” para sobrescribir métodos en una de estas en una clase hija.

Ejercicio 01:

Nuestra primera clase será llamada “Electrodomesticos”, como la del ejercicio anterior pero, esta también será superclase de otras dos llamadas “Lavadora” y “Television”, en estas dos nuevas clases definiremos unos atributos propios (para “Lavadora” añadiremos “double carga” → “5kg como predeterminado” y para “Televisión”, “double resolucion” → “20” y “boolean sintonicazion” → “false”). Cada uno de los atributos tendrán una correlación el precio final del producto:

CONSUMO			
A	+100€	B	+80€
C	+60€	D	+50€
E	+30€	F	+10€
PESO			
Entre 0kg y 19kg	+10€	Entre 20kg y 49kg	+50€
Entre 50kg y 79kg	+80€	Más de 80kg	+100€
LAVADOR → CARGA			
Mayor de 30kg		+50€	
TELEVISIÓN → RESOLUCIÓN			
Mayor de 40 pulgadas		+30%	
TELEVISIÓN → SINTONIZADOR TDT			
Sí		+50€	

Por último, deberemos hacer un main donde creamos 5 objetos de cada tipo, sin contar “Electrodomesticos”, recorrer el array y dar el precio total de las lavadoras, los televisores y el total de todos los electrodomésticos.

Clase “Electrodomestico”

ATRIBUTOS

```java
import java.util.*;

public class Electrodomestico {
    //  ATRIBUTOS
    protected double precio;
    protected static final double PRECIO_PREDETERMINADO = 100.0;

    protected String color;
    protected static final String COLOR_PREDETERMINADO = "blanco";
    protected static final List<String> COLORES_DISPONIBLES = Arrays.asList(...a:"negro", "rojo", "azul", "gris");

    protected char consumo;
    protected static final char CONSUMO_PREDETERMINADO = 'F';
    protected static final List<Character> CONSUMOS_DISPONIBLES = Arrays.asList(...a:'A', 'B', 'C', 'D', 'E');

    protected double peso;
    protected static final double PESO_PREDETERMINADO = 5;
```

CONSTRUCTORES

```java
//  CONSTRUCTORES
public Electrodomestico() {
    this(PRECIO_PREDETERMINADO, PESO_PREDETERMINADO, COLOR_PREDETERMINADO, CONSUMO_PREDETERMINADO);
}

public Electrodomestico(double precio, double peso) {
    this(precio, peso, COLOR_PREDETERMINADO, CONSUMO_PREDETERMINADO);
}

public Electrodomestico(double precio, double peso, String color, char consumo) {
    this.precio = precio;
    this.peso = peso;
    this.color = comprobarColor(color);
    this.consumo = comprobarLetraConsumo(consumo);
}
```

MÉTODOS

```java
// METODOS
public double getPrecio() {
    return precio;
}

public double getPeso() {
    return peso;
}

public String getColor() {
    return color;
}

public Character getConsumo() {
    return consumo;
}
```

```java
ivate String comprobarColor(String color) {
  color = color.toLowerCase();

  if (COLORES_DISPONIBLES.contains(color)) {
      return color;
  } else {
      return COLOR_PREDETERMINADO;
  }


ivate char comprobarLetraConsumo(char letraConsumo) {
  letraConsumo = Character.toUpperCase(letraConsumo);

  if (CONSUMOS_DISPONIBLES.contains(letraConsumo)) {
      return letraConsumo;
  } else {
      return CONSUMO_PREDETERMINADO;
  }
```

```java
public double precioFinal() {
    double precioFinal = precio;

    if (peso >= 0 && peso < 20) {
        precioFinal += 10;
    } else if (peso < 50) {
        precioFinal += 50;
    } else if (peso < 80) {
        precioFinal += 80;
    } else if  (peso >= 80) {
        precioFinal += 100;
    }

    switch(consumo) {
        case 'A':{
            precioFinal += 100;
            break;
        }

        case 'B':{
            precioFinal += 80;
            break;
        }
        case 'C':{
            precioFinal += 60;
            break;
        }

        case 'D':{
            precioFinal += 50;
            break;
        }

        case 'E':{
            precioFinal += 30;
            break;
        }

        default:{
            precioFinal += 10;
            break;
        }
    }

    return precioFinal;
}
```

```java
@Override
public String toString() {
    return "Electrodoméstico:\n\tPrecio base: " + precio +
     " €\n\tColor: " + color + "\n\tConsumo energético: " +
      consumo + "\n\tPeso: " + peso + " kg\n    Precio final: "+precioFinal();
}
```

Clase “Lavadora”

ATRIBUTOS

```java
public class Lavadora extends Electrodomestico{
    //  ATRIBUTO
    private double carga;
    private final double CARGA_PREDETERMINADA = 5;
```

MÉTODOS

```java
//  METODOS
public double getCarga() {
    return carga;
}

@Override
public double precioFinal() {
    double precioFinal = super.precioFinal();

    if (carga > 30) {
        precioFinal += 50;
    }

    return precioFinal;
}

@Override
public String toString() {
    return "Lavadora:\n\tPrecio base: " + precio + " €\n\tColor: " +
    color + "\n\tConsumo energético: " + consumo + "\n\tPeso: " +
    peso + " kg\n\tCarga: "+carga+"\n    Precio final: "+precioFinal();
}
```

CONSTRUCTORES

```java
//  CONSTRUCTORES
public Lavadora() {
    super();

    this.carga = CARGA_PREDETERMINADA;
}

public Lavadora(double precio, double peso) {
    super(precio, peso);

    this.carga = CARGA_PREDETERMINADA;
}

public Lavadora(double carga, double precio, double peso, String color, char consumo) {
    super(precio, peso, color, consumo);
    this.carga = carga;
}
```

Clase “Television”

ATRIBUTOS

```java
public class Television extends Electrodomestico{
    //  ATRIBUTOS
    private double resolucion;
    private final double RESOLUCION_PREDETERMINADA = 20;
    private boolean sintonizador;
    private final boolean SINTONIZADOR_PREDETERMINADO = false;
```

MÉTODOS

```java
//  METODOS
public double getResolucion() {
    return resolucion;
}

public boolean getSintonizador() {
    return sintonizador;
}

@Override
public double precioFinal() {
    double precioFinal = super.precioFinal();

    if (resolucion > 40) {
        precioFinal += precio * 0.30;
    }

    if (sintonizador == true) {
        precioFinal += 50;
    }

    return precioFinal;
}

@Override
public String toString() {
    return "Television:\n\tPrecio base: " + precio + " €\n\tColor: " +
    color + "\n\tConsumo energético: " + consumo + "\n\tPeso: " +
    peso + " kg\n\tResolución: " + resolucion + "\n\tSintonización TDT: " +
    sintonizador + "\n    Precio final: "+precioFinal();
}
```

CONSTRUCTORES

```java
//  CONSTRUCTORES
public Television() {
    super();

    this.resolucion = RESOLUCION_PREDETERMINADA;
    this.sintonizador = SINTONIZADOR_PREDETERMINADO;
}

public Television(double precio, double peso) {
    super(precio, peso);

    this.resolucion = RESOLUCION_PREDETERMINADA;
    this.sintonizador = SINTONIZADOR_PREDETERMINADO;
}

public Television(double precio, double peso, String color,
 char consumo, double resolucion, boolean sintonizador) {
    super(precio, peso, color, consumo);

    this.resolucion = resolucion;
    this.sintonizador = sintonizador;
}
```

Main

```java
import java.text.*;
import java.util.*;

public class electrodosMain {
    Run | Debug
    public static void main(String[] args) {
        DecimalFormat formateo = new DecimalFormat(pattern:"0.00");
        Television television1 = new Television(precio:120.20, peso:12.5,
        color:"negro", consumo:'D', resolucion:22, sintonizador:true);
        Television television2 = new Television(precio:250.0, peso:15.0,
        color:"blanco", consumo:'A', resolucion:32, sintonizador:false);
        Television television3 = new Television(precio:400.50, peso:18.7,
        color:"rojo", consumo:'C', resolucion:42, sintonizador:true);
        Television television4 = new Television(precio:699.99, peso:20.2,
        color:"azul", consumo:'B', resolucion:50, sintonizador:false);
        Television television5 = new Television(precio:899.95, peso:25.0,
        color:"gris", consumo:'E', resolucion:55, sintonizador:true);

        Lavadora lavadora1 = new Lavadora(carga:8, precio:299.99,
        peso:45.0, color:"azul", consumo:'A');
        Lavadora lavadora2 = new Lavadora(carga:7, precio:199.0,
        peso:40.0, color:"rojo", consumo:'B');
        Lavadora lavadora3 = new Lavadora(carga:9, precio:399.50,
        peso:50.5, color:"gris", consumo:'C');
        Lavadora lavadora4 = new Lavadora(carga:6, precio:499.99,
        peso:55.2, color:"negro", consumo:'D');
        Lavadora lavadora5 = new Lavadora(carga:10, precio:599.95,
        peso:60.0, color:"blanco", consumo:'E');

        ArrayList<Object> electrodomesticos = new ArrayList<>();
        electrodomesticos.addAll(List.of(television1, television2,
        television3, television4, television5, lavadora1, lavadora2,
        lavadora3, lavadora4, lavadora5));

        double precioElectrodomesticos = 0.0;
        double precioLavadoras = 0.0;
        double precioTelevisiones = 0.0;
        for (Object electrodo : electrodomesticos) {
            if (electrodo instanceof Television) {
                Television tele = (Television) electrodo;
                precioTelevisiones += tele.precioFinal();
                precioElectrodomesticos += tele.precioFinal();
            } else {
                Lavadora lava = (Lavadora) electrodo;
                precioLavadoras += lava.precioFinal();
                precioElectrodomesticos += lava.precioFinal();
            }
        }

        System.out.println("Precio de las lavadoras:\t" + formateo.format(precioLavadoras) +
        "€\nPrecio de los televisores:\t" + formateo.format(precioTelevisiones) +
        "€\nPrecio total electrodomésticos:\t" + formateo.format(precioElectrodomesticos) + "€");
    }
}
```

Resultado consola

```
Precio de las lavadoras:        2658,43?
Precio de los televisores:      3570,77?
Precio total electrodomésticos: 6229,20?
```

Ejercicio 02:

Para el ejercicio dos, utilizaremos el archivo del tema anterior llamada “Serie” y, crearemos una clase hermana llamada “Videojuego” para enlazarlas con una interfaz llamada “Entregable” donde crearemos los métodos: entregar (establece true a “entregado”), devolver (establece false a “entregado”) y estaEntregado (Devuelve el estado de “entregado”). Para cada una de las clases tendremos que definir unos atributos, sus getters & setters y sobreescribimos el método toString.

Clase “Serie”

ATRIBUTOS

```java
import java.util.*;

public class Series implements Entregable {
    // ATRIBUTOS
    private String titulo;
    private final String TITULO_PRED = "";
    private int temporadas;
    private final int TEMP_PRED = 3;
    private boolean entregado;
    private final boolean ENTREGA_PRED = false;
    private String genero;
    private final String GENERO_PRED = "";
    private String autor;
    private final String AUTOR_PRED = "";
```

CONSTRUCTORES

```java
// CONSTRUCTORES
public Series() {
    this.titulo = TITULO_PRED;
    this.temporadas = TEMP_PRED;
    this.entregado = ENTREGA_PRED;
    this.genero = GENERO_PRED;
    this.autor = AUTOR_PRED;
}

public Series(String titulo, String autor) {
    this.titulo = titulo;
    this.temporadas = TEMP_PRED;
    this.entregado = ENTREGA_PRED;
    this.genero = GENERO_PRED;
    this.autor = autor;
}

public Series(String titulo, int temporadas, String genero, String autor) {
    this.titulo = titulo;
    this.temporadas = temporadas;
    this.entregado = ENTREGA_PRED;
    this.genero = genero;
    this.autor = autor;
}
```

MÉTODOS

```java
// METODOS
public String getTitulo(){
    return titulo;
}

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

public int getTemporadas() {
    return temporadas;
}

public void setTemporada(int temporadas) {
    this.temporadas = temporadas;
}
```

```java
public String getGenero() {
    return genero;
}

public void setGenero(String genero) {
    this.genero = genero;
}

public String getAutor() {
    return autor;
}

public void setAutor(String autor) {
    this.autor = autor;
}
```

```java
@Override
public String toString() {
    return "  -    Serie\n\t{" +
            "titulo='" + titulo + '\'' +
            ", temporadas=" + temporadas + '\'' +
            ", entregado=" + entregado + '\'' +
            ", genero=" + genero + '\'' +
            ", autor=" + autor + '\'' +
            '}';
}

public String masTemporadasToString() {
    return " La serie con mayor cantidad de " +
    "temporadas es:\n\t'" + titulo + "' con '" +
    temporadas + "' temporadas.";
}

@Override
public void entregado() {
    this.entregado = true;
}
```

```java
@Override
public void devolver() {
    this.entregado = false;
}

@Override
public boolean estaEntregado() {
    return entregado;
}

@Override
public Series comparativa(ArrayList<Object> Juegos_Series) {
    Series serie = new Series();
    int temporadas = 0;
    for (Object o : Juegos_Series) {
        if (o instanceof Series) {
            Series serialKiller = (Series) o;
            if (serialKiller.getTemporadas() > temporadas) {
                temporadas = serialKiller.getTemporadas();
                serie = serialKiller;
            }
        }
    }
    return serie;
}
```

Clase “Videojuego”

ATRIBUTOS

```java
import java.util.*;

public class Videojuego implements Entregable {
    //  ATRIBUTOS
    private String titulo;
    private final String TITULO_PRED = "";
    private int horas_estimadas;
    private final int HORAS_PRED = 10;
    private boolean entregado;
    private final boolean ENTREGA_PRED = false;
    private String genero;
    private final String GENERO_PRED = "";
    private String compania;
    private final String COMPANIA_PRED = "";
```

CONSTRUCTORES

```java
//  CONSTRUCTORES
public Videojuego() {
    this.titulo = TITULO_PRED;
    this.horas_estimadas = HORAS_PRED;
    this.entregado = ENTREGA_PRED;
    this.genero = GENERO_PRED;
    this.compania = COMPANIA_PRED;
}

public Videojuego(String titulo,  int horas_estimadas) {
    this.titulo = titulo;
    this.horas_estimadas = horas_estimadas;
    this.entregado = ENTREGA_PRED;
    this.genero = GENERO_PRED;
    this.compania = COMPANIA_PRED;
}

public Videojuego(String titulo,  int horas_estimadas, String genero, String compania) {
    this.titulo = titulo;
    this.horas_estimadas = horas_estimadas;
    this.entregado = ENTREGA_PRED;
    this.genero = genero;
    this.compania = compania;
}
```

MÉTODOS

```java
//  METODOS
public String getTitulo() {
    return titulo;
}

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

public int getHoras() {
    return horas_estimadas;
}

public void setHoras(int horas) {
    this.horas_estimadas = horas;
}
```

```java
public String getGenero() {
    return genero;
}

public void setGenero(String genero) {
    this.genero = genero;
}

public String getCompania() {
    return compania;
}

public void setCompania(String compania) {
    this.compania = compania;
}
```

```java
@Override
public String toString() {
    return "  -   Juego\n\t{" +
            "titulo= '" + titulo + '\'' +
            ", horas estimadas= '" +
            horas_estimadas + " horas"+'\'' +
            ", entregado= '" + entregado + '\'' +
            ", genero= '" + genero + '\'' +
            ", compañia= '" + compania + '\'' +
            '}';
}

public String masHorasToString() {
    return " El juego con mayor cantidad de " +
    "horas es:\n\t'" + titulo + "' con un promedio " +
    " de horas de --> '" + horas_estimadas + "'";
}

@Override
public void entregado() {
    this.entregado = true;
}
```

```java
@Override
public void devolver() {
    this.entregado = false;
}

@Override
public boolean estaEntregado() {
    return entregado;
}

@Override
public Videojuego comparativa(ArrayList<Object> Juegos_Series) {
    Videojuego juego = new Videojuego();
    int horasMaximas = 0;
    for (Object o : Juegos_Series) {
        if (o instanceof Videojuego) {
            Videojuego jogoBonito = (Videojuego) o;
            if (jogoBonito.getHoras() > horasMaximas) {
                horasMaximas = jogoBonito.getHoras();
                juego = jogoBonito;
            }
        }
    }
    return juego;
}
```

Interfaz

```java
import java.util.*;

public interface Entregable {
    public void entregado();
    public void devolver();
    public boolean estaEntregado();
    public Object comparativa(ArrayList<Object> a);
}
```

Main

```java
import java.util.*;

public class mainEntregaDoor {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<Videojuego> Juegos = new ArrayList<>();
        ArrayList<Series> Series = new ArrayList<>();

        Videojuego juego1 = new Videojuego(titulo:"The Legend of Zelda: Breath of the Wild",
        horas_estimadas:120, genero:"Aventura", compania:"Nintendo");
        Videojuego juego2 = new Videojuego(titulo:"Super Mario Odyssey",
        horas_estimadas:60, genero:"Plataforma", compania:"Nintendo");
        Videojuego juego3 = new Videojuego(titulo:"Hollow Knight",
        horas_estimadas:28, genero:"Metroidvania", compania:"Team Cherry");
        Videojuego juego4 = new Videojuego(titulo:"The Witcher 3: Wild Hunt",
        horas_estimadas:100, genero:"RPG", compania:"CD Projekt Red");
        Videojuego juego5 = new Videojuego(titulo:"Red Dead Redemption 2",
        horas_estimadas:60, genero:"Acción-Aventura", compania:"Rockstar Games");
        Juegos.addAll(Arrays.asList(juego1, juego2, juego3, juego4, juego5));

        Series serie1 = new Series(titulo:"Breaking Bad", temporadas:5,
        genero:"Droga(Drama)", autor:"Vince Gilligan");
        Series serie2 = new Series(titulo:"Game of Thrones", temporadas:8,
        genero:"Fantasía", autor:"David Benioff, D. B. Weiss");
        Series serie3 = new Series(titulo:"Hunter x Hunter", temporadas:6,
        genero:"Anime", autor:"Yoshihiro Togashi");
        Series serie4 = new Series(titulo:"Friends", temporadas:10,
        genero:"Comedia", autor:"David Crane, Marta Kauffman");
        Series serie5 = new Series(titulo:"Stranger Things", temporadas:4,
        genero:"Ciencia ficción", autor:"Duffer Brothers");
        Series.addAll(Arrays.asList(serie1, serie2, serie3, serie4, serie5));

        ArrayList<Object> Series_Juegos = new ArrayList<>();
        Series_Juegos.addAll(Arrays.asList(serie1, serie2, serie3, serie4, serie5,
        juego1, juego2, juego3, juego4, juego5));

        //  SELECCIONAR ALGUNOS COMO ENTREGADOS
        juego1.entregado();
        juego3.entregado();
        serie2.entregado();
        serie4.entregado();
        serie5.entregado();

        int contadorEntregrados = 0;
        ArrayList<String> titulos = new ArrayList<>();

        for (Object o : Juegos) {
            Videojuego SerieJuego = (Videojuego) o;
            if (SerieJuego.estaEntregado()) {
                titulos.add(SerieJuego.getTitulo());
                contadorEntregrados++;
                SerieJuego.devolver();
            }
        }
        for (Object o : Series) {
            Series SerieJuego = (Series) o;
            if (SerieJuego.estaEntregado()) {
                titulos.add(SerieJuego.getTitulo());
                contadorEntregrados++;
                SerieJuego.devolver();
            }
        }

        if (contadorEntregrados == 0) {
            System.out.println(x:"No hay nada entregado");
        } else {
            System.out.println("Hay "+ contadorEntregrados+
            " entregas los cuales son: \n\t"+titulos+"\n");
        }

        Series serieMayorTemporadas = new Series();
        Videojuego juegoMasLargo = new Videojuego();

        for (Videojuego juego : Juegos) {
            juegoMasLargo = juego.comparativa(Series_Juegos);
        }
        for (Series serie : Series) {
            serieMayorTemporadas = serie.comparativa(Series_Juegos);
        }
        System.out.println(juegoMasLargo.masHorasToString()+
        "\n\n"+serieMayorTemporadas.masTemporadasToString());
    }
}
```

Resultado consola

```
Hay 5 entregas los cuales son:
        [The Legend of Zelda: Breath of the Wild, Hollow Knight, Game of Thrones, Friends, Stranger Things]

El juego con mayor cantidad de horas es:
        'The Legend of Zelda: Breath of the Wild' con un promedio  de horas de --> '120'

La serie con mayor cantidad de temporadas es:
        'Friends' con '10' temporadas.
```

Ejercicio 03:

Ahora tendremos que crear una clase llamada “Libro” en la cual representaremos cuatro atributos: ISBN, título, autor y su número de páginas. Después crearemos sus métodos getters & setters y sobreescribimos el método “toString” para mostrar la información de una manera concreta (“El libro con ISBN creado por el autor tiene páginas”).
En el main crearemos dos libros y mostraremos por pantalla sus respectivos “.toString” e indicaremos cual es el que tiene mayor cantidad de páginas.

Clase “Libros”

ATRIBUTOS

```java
public class Libros {
    //  ATRIBUTOS
    private Long ISBN;
    private String titulo;
    private String autor;
    private int paginas;
```

CONSTRUCTORES

```java
//   CONSTRUCTORES
public Libros() {
    this.ISBN = 0L;
    this.titulo = "";
    this.autor = "";
    this.paginas = 0;
}

public Libros(Long ISBN, String titulo, String autor, int paginas, boolean error) {
    this.ISBN = ISBN;
    this.titulo = titulo;
    this.autor = autor;
    this.paginas = paginas;
}
```

MÉTODOS

```java
//  METODOS
public void setISBN(Long ISBN) {
    this.ISBN = ISBN;
}

public Long getISBN() {
    return ISBN;
}

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

public String getTitulo() {
    return titulo;
}

public void setAutor(String autor) {
    this.autor = autor;
}

public String getAutor() {
    return autor;
}

public void setPaginas(int pagina) {
    this.paginas = pagina;
}

public int getPaginas() {
    return paginas;
}

@Override
public String toString() {
    return "El libro <" + titulo + "> con ISBN : '" +
    ISBN + "' creado por " + autor + " tiene "+paginas;
}
```

Main

```java
//  MAIN
Run | Debug
public static void main(String[] args) {
    Libros libro1 = new Libros(ISBN:9788408284550L, titulo:"Alas de sangre", autor:"Rebecca Yarros", paginas:896, error:false);
    Libros libro2 = new Libros(ISBN:9788496735712L, titulo:"Wonder", autor:"R.J. PALACIO", paginas:424, error:false);
    System.out.println(libro1.toString());
    System.out.println(libro2.toString());
    if (libro1.getPaginas() > libro2.getPaginas()) {
        System.out.println("El libro con mayor cantidad de páginas es \"" + libro1.getTitulo() + "\" con '" + libro1.getPaginas() + "' páginas, el segundo '"+ libro2.getPaginas() + "'");
    } else {
        System.out.println("El libro con mayor cantidad de páginas es \"" + libro2.getTitulo() + "\" con '" + libro2.getPaginas() + "' páginas, el segundo '"+ libro1.getPaginas() + "'");
    }
}
```

Resultado consola

```
El libro <Alas de sangre> con ISBN : '9788408284550' creado por Rebecca Yarros tiene 896
El libro <Wonder> con ISBN : '9788496735712' creado por R.J. PALACIO tiene 424
El libro con mayor cantidad de páginas es "Alas de sangre" con '896' páginas, el segundo '424'
```

Ejercicio 04:

El cuarto programa que haremos será para calcular una raíz cuadrática de segundo grado en base a los tres valores que insertamos. También deberemos crear seis métodos para recoger el discriminante de la operación, si tiene varias raíces, si tiene una sola, para calcular la raíz, y ara obtener las dos raíces o la única raíz de la operación.

Clase “Raices”

ATRIBUTOS

```java
import java.text.*;
import java.util.*;

public class Raices {
    //   ATRIBUTOS
    private double a;
    private double b;
    private double c;
    private double resultado1;
    private double resultado2;
```

CONSTRUCTORES

```java
//  CONSTRUCTORES
public Raices() {
    this.a = 0;
    this.b = 0;
    this.c = 0;
}

public Raices(double a, double b, double c) {
    this.a = a;
    this.b = b;
    this.c = c;
}
```

MÉTODOS

```java
//METODOS
public double getDiscriminate() {
    return (Math.pow(b, b:2)) - 4 * a *c;
}

public boolean tieneRaices(double discriminenteN) {
    if (discriminenteN > 0) {
        return true;
    } else {
        return false;
    }
}

public boolean tieneRaiz(double discriminenteN) {
    if (discriminenteN == 0) {
        return true;
    } else {
        return false;
    }
}

public String obtenerRaiz() {
    DecimalFormat dosDecimales = new DecimalFormat(pattern:"0.000");
    String respu = dosDecimales.format(resultado1);
    return "\nTiene una unica raiz: " + respu;
}
```

```java
public String obtenerRaices() {
    DecimalFormat dosDecimales = new DecimalFormat(pattern:"0.000");
    String respu1 = dosDecimales.format(resultado1);
    String respu2 = dosDecimales.format(resultado2);
    return "\nPrimera: " + respu1 + "\nSegunda: " + respu2;
}

public void calcular() {
    double B_alcuadrado = Math.pow(b, b:2);
    System.out.println("B_Alcuadrado: "+B_alcuadrado);
    double cuatro_A_C = 4*a*c;
    System.out.println("cuatro_A_C: "+cuatro_A_C);
    double raiz = Math.sqrt(B_alcuadrado - cuatro_A_C);
    System.out.println("raiz: "+raiz);
    double nuevaB = b * (-1);
    System.out.println("nuevaB: "+nuevaB);
    double divisor = 2 * a;
    System.out.println("divisor: "+divisor);

    resultado1 = (nuevaB + raiz) / divisor;
    System.out.println("resultado1: "+ resultado1);
    resultado2 = (nuevaB - raiz) / divisor;
    System.out.println("resultado2: "+ resultado2);

}
```

Main

```java
//  MAIN
Run | Debug
public static void main(String[] args) {
    Scanner scaner = new Scanner(System.in);
    System.out.println("Indique los números con los que se "+
    " hara la equacion de segundo grado:\n - A:");
    double a = Double.parseDouble(scaner.next());
    System.out.println(x:"\n - B:");
    double b = Double.parseDouble(scaner.next());
    System.out.println(x:"\n - C:");
    double c = Double.parseDouble(scaner.next());
    System.out.println(x:"\n");
```

```java
    Raices raiz_cuadratica = new Raices(a, b, c);
    double negro = raiz_cuadratica.getDiscriminate();
    raiz_cuadratica.calcular();
    System.out.println("Discriminante: "+negro);
    if(!raiz_cuadratica.tieneRaiz(negro) && !raiz_cuadratica.tieneRaices(negro)) {
        System.out.print(s:"No tiene raices esta ecuación");
    } else if(raiz_cuadratica.tieneRaices(negro)) {
        System.out.print(raiz_cuadratica.obtenerRaices());
    } else {
        System.out.print(raiz_cuadratica.obtenerRaiz());
    }
    scaner.close();
}
```

Resultado consola

```
Indique los números con los que se hara la equacion de segundo grado:
 - A:
1

 - B:
5

 - C:
2
```

```
B_Alcuadrado: 25.0
cuatro_A_C: 8.0
raiz: 4.123105625617661
nuevaB: -5.0
divisor: 2.0
resultado1: -0.4384471871911697
resultado2: -4.561552812808831
Discriminante: 17.0

Primera: -0,438
Segunda: -4,562
```

Ejercicio 05:

En el quinto ejercicio representaremos un colegio creando tres clases principales: una llamada “Aula”; donde definiremos un id, el número de estudiantes que debería haber y la materia que se imparte en el aula; otra profesores y la última de estudiantes. En estas dos últimas no especifico los parametro porque comparten la gran parte (nombre, edad, sexo) por lo que crearemos una superclase con estos llamada “Personas”. Una vez creada, añadiremos el atributo “materia” a la clase-hija “Profesores” y a “Estudiantes” su calificación, “nota”. Con el mismo racionamiento, he creado una interfaz, “MetodosPersonas”, para el método en común que tienen, “falta()”.
Para poder dar clase se tienen que cumplir los siguientes requisitos: la materia del aula y del profesor son la misma, el profesor está disponible y hay más de un 50% de los alumnos en clase.
Finalmente, deberemos crear en el main X cantidad de estudiantes, un profesor y un aula, en base a los valores asignados determinar si se puede o no dar clase y, en caso de poder, mostrar los alumnos aprobados.

Interfaz

```java
public interface MetodosPersonas {
    public boolean falta();
}
```

Clase-Padre “Personas”

```java
public class Personas implements MetodosPersonas {
    //  ATRIBUTOS
    protected String nombre;
    protected int edad;
    protected String sexo;
    protected final String[] sexoOpciones =
    {"Sin seleccionar", "H", "M"};

    //  CONSTRUCTORES
    public Personas() {
        this.nombre = "";
        this.edad = 0;
        this.sexo = sexoOpciones[0];
    }

    public Personas(String nombre, int edad,
    String sexo) {
        this.nombre = nombre;
        this.edad = edad;
        if (sexo.equals(sexoOpciones[1])) {
            this.sexo = sexoOpciones[1];
        } else if (sexo.equals(sexoOpciones[2])) {
            this.sexo = sexoOpciones[2];
        } else {
            this.sexo = sexoOpciones[0];
            System.out.println(x:"Opciones:  'H'  'M'");
        }
    }

    //  METODOS
    public String getNombre() {
        return nombre;
    }

    @Override
    public boolean falta() {
        return false;
    }
}
```

Clase “Profesores”

ATRIBUTOS

```java
import java.util.*;

public class Profesores extends Personas {
    //  ATRIBUTOS
    private String materia;
    private final String[] materiaOpciones =
    {"Sin seleccionar", "matematicas", "filosofia", "fisica"};
    protected boolean ausente;
```

CONSTRUCTORES

```java
//CONSTUCTORES
public Profesores() {
    super();

    this.materia = materiaOpciones[0];
    this.ausente = false;
}

public Profesores(String nombre, int edad, String sexo, String materia) {
    super(nombre, edad, sexo);

    for (int i = 0; i <= 3; i++) {
        if (materia.equals(materiaOpciones[i])) {
            this.materia = materiaOpciones[i];
            break;
        } else {
            this.materia = materiaOpciones[0];
        }
    }

    this.ausente = falta();
}
```

MÉTODOS

```java
// METODOS
public void setMateria(String materia) {
    this.materia = materia;
}

public String getMateria() {
    return materia;
}

public void setAusencia(boolean ausente) {
    this.ausente = ausente;
}

public boolean getAusencia() {
    return ausente;
}

@Override
public boolean falta() {
    Random probabilidad = new Random();
    int numProbabilidad = probabilidad.nextInt(bound:100) + 1;
    if (numProbabilidad <= 20) {
        return true;
    } else {
        return false;
    }
}
```

Clase “Estudiantes”

ATRIBUTOS

```java
import java.util.*;

public class Estudiantes extends Personas {
    //  ATRIBUTOS
    private double nota;
    protected boolean ausente;
```

CONSTRUCTORES

```java
// CONSTRUCTORES
public Estudiantes() {
    super();

    this.nota = 0;
    this.ausente = false;
}
```

```java
public Estudiantes(String nombre, int edad, String sexo, double nota) {
    super(nombre, edad, sexo);

    if (nota > 0 && nota < 10) {
        this.nota = nota;
    } else {
        this.nota = 0;
        System.out.println(x:"La nota del alumno debe estar entre '0' y '10'");
    }
    this.ausente = falta();
}
```

MÉTODOS

```java
//METODOS
public void setAusencia(boolean ausente) {
    this.ausente = ausente;
}

public boolean getAusencia() {
    return ausente;
}

public void setNota(int nota) {
    this.nota = nota;
}

public double getNota() {
    return nota;
}
```

```java
public static ArrayList<Estudiantes> alumnosAusentes(
    ArrayList<Estudiantes> estudiantes) {
    ArrayList<Estudiantes> ausentes = new ArrayList<>();
    for(Estudiantes estudiante : estudiantes) {
        if (estudiante.getAusencia()) {
            ausentes.add(estudiante);
        }
    }
    return ausentes;
}

public static int cantAlumnosEnClase(
    ArrayList<Estudiantes> estudiantes) {
    int cantidadEnClase = 0;
    for(Estudiantes estudiante : estudiantes) {
        if (!estudiante.getAusencia()) {
            cantidadEnClase++;
        }
    }
    return cantidadEnClase;
}
```

```java
public static ArrayList<Estudiantes> alumnosAprobados
(ArrayList<Estudiantes> estudiantes) {
    ArrayList<Estudiantes> aprobados = new ArrayList<>();
    for (Estudiantes estudiante : estudiantes) {
        if (estudiante.getNota() >= 5) {
            aprobados.add(estudiante);
        }
    }
    return aprobados;
}

public static ArrayList<Estudiantes> alumnosNOAprobados(
    ArrayList<Estudiantes> estudiantes) {
    ArrayList<Estudiantes> suspendidos = new ArrayList<>();
    for (Estudiantes estudiante : estudiantes) {
        if (estudiante.getNota() < 5) {
            suspendidos.add(estudiante);
        }
    }
    return suspendidos;
}
```

```java
    @Override
    public boolean falta() {
        Random probabilidad = new Random();
        int numProbabilidad = probabilidad.nextInt(bound:100) + 1;
        if (numProbabilidad <= 50) {
            return true;
        } else {
            return false;
        }
    }
}
```

Clase “Aula”

ATRIBUTOS

```java
public class Aula {
    //  ATRIBUTOS
    private int id;
    private int numMaxEstudiantes;
    private String materiaAula;
    private final String[] materiaOpciones = {"Sin seleccionar", "matematicas", "filosofia", "fisica"};
    private boolean sePuedeDarClase;
```

CONSTRUCTORES

```java
//  CONSTRUCTORES
public Aula() {
    this.id = 0;
    this.numMaxEstudiantes = 0;
    this.materiaAula = materiaOpciones[0];
    this.sePuedeDarClase = false;
}

public Aula(int id, int numMaxEstudiantes, String materiaAula, int estudiantesEnClase, Profesores profesor) {
    this.id = id;
    this.numMaxEstudiantes = numMaxEstudiantes;

    //materia
    for (int i = 0; i <= 3; i++) {
        if (materiaAula.equals(materiaOpciones[i])) {
            this.materiaAula = materiaOpciones[i];
            break;
        } else {
            this.materiaAula = materiaOpciones[0];
        }
    }

    this.sePuedeDarClase = posibilidadDeDarClase(estudiantesEnClase, profesor);
}
```

MÉTODOS

```java
//  METODOS
public void setID(int id) {
    this.id = id;
}

public int getID() {
    return id;
}

public void setNumMaxEstudiantes(int numMaxEstudiantes) {
    this.numMaxEstudiantes = numMaxEstudiantes;
}
public int getNumMaxEstudiantes() {
    return numMaxEstudiantes;
}

public void setMateriaAula(String materiaAula) {
    this.materiaAula = materiaAula;
}

public String getMateriaAula() {
    return materiaAula;
}
```

```java
public boolean sePuedeDarClase() {
    return sePuedeDarClase;
}

public boolean posibilidadDeDarClase(int estudiantesEnClase, Profesores profesor) {
    if (!profesor.getMateria().equals(materiaAula)) {
        System.out.println("Materia profesor: " + profesor.getMateria() +
        "\nMateria aula: " + materiaAula + "\n");
        return false;
    } else if (profesor.getAusencia()) {
        System.out.println(x:"Profesor ausente\n");
        return false;
    } else if (estudiantesEnClase < (numMaxEstudiantes / 2)) {
        System.out.println("Cantidad de alumnos insuficientes.\nAlumnos asignados al aula: " +
        numMaxEstudiantes + "\nAlumnos en el aula: " + estudiantesEnClase +
        "\nMínimo requerido: " + (numMaxEstudiantes / 2) + "\n");
        return false;
    } else {
        return true;
    }
}
```

Main

```java
import java.text.*;
import java.util.*;

public class mainColegio {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<Estudiantes> estudiantes = new ArrayList<>();
        DecimalFormat formato = new DecimalFormat(pattern:"0.00");
        Random nota = new Random();

        double nota1 = nota.nextDouble() * 10;
        Estudiantes estudiante1 = new Estudiantes(nombre:"Juan", edad:15, sexo:"H", nota1);
                                Del estudiante2 al estudiante14
        double nota15 = nota.nextDouble() * 10;
        Estudiantes estudiante15 = new Estudiantes(nombre:"Luisa", edad:14, sexo:"M", nota15);

        estudiantes.addAll(Arrays.asList(estudiante1, ..., estudiante15));

        Profesores profesor = new Profesores(nombre:"Margarita", edad:42, sexo:"M", materia:"filosofia");
```

```java
    int cantidadEnClase = Estudiantes.cantAlumnosEnClase(estudiantes);
    ArrayList<Estudiantes> ausentes = Estudiantes.alumnosAusentes(estudiantes);
    ArrayList<Estudiantes> aprobados = Estudiantes.alumnosAprobados(estudiantes);
    ArrayList<Estudiantes> suspendidos = Estudiantes.alumnosNOAprobados(estudiantes);

    Aula aulaFilosofia = new Aula(id:1, estudiantes.size(), materiaAula:"filosofia", cantidadEnClase, profesor);
    if (aulaFilosofia.sePuedeDarClase()) {
        System.out.println("\t- - ALUMNOS APROBADOS ["+aprobados.size()+"] - -\n");
        for (Estudiantes alumno : aprobados) {
            if (ausentes.contains(alumno)) {
                System.out.println("¡AUSENTE! El alumno "+alumno.getNombre()+" está aprobado con un "+formato.format(alumno.getNota()));
            } else {
                System.out.println("El alumno "+alumno.getNombre()+" está aprobado con un "+formato.format(alumno.getNota()));
            }
        }

        System.out.println("\n\t- - ALUMNOS SUSPENDIDOS ["+suspendidos.size()+"] - -\n");
        for (Estudiantes alumno : suspendidos) {
            if (ausentes.contains(alumno)) {
                System.out.println("¡AUSENTE! El alumno "+alumno.getNombre()+" está suspendido con un "+formato.format(alumno.getNota()));
            } else {
                System.out.println("El alumno "+alumno.getNombre()+" está suspendido con un "+formato.format(alumno.getNota()));
            }
        }

        System.out.println("\n\t- - ALUMNOS AUSENTES ["+ausentes.size()+"] - -\n");
        for (Estudiantes alumno : ausentes) {
            System.out.println("El alumno "+alumno.getNombre()+" está ausente");
        }

        System.out.println(x:"\nSe puede dar clase");
    } else {
        System.out.println(x:"No se puede dar clase");
    }
}
```

Resultado consola

```
                                                - - ALUMNOS APROBADOS [10] - -

                    TIPOS DE ERRORES                        ¡AUSENTE! El alumno Juan está aprobado con un 6,46
                                                            El alumno Elisabeth está aprobado con un 5,69
                                                            ¡AUSENTE! El alumno Maria está aprobado con un 7,22
                                                            El alumno Pedro está aprobado con un 8,90
                    Profesor ausente                        El alumno Carlos está aprobado con un 7,62
                                                            ¡AUSENTE! El alumno Ana está aprobado con un 5,97
                    No se puede dar clase                   ¡AUSENTE! El alumno Pablo está aprobado con un 9,16
                                                            El alumno Sara está aprobado con un 8,85
            Cantidad de alumnos insuficientes.              El alumno Natalia está aprobado con un 5,42
            Alumnos asignados al aula: 15                   El alumno Luisa está aprobado con un 5,27
            Alumnos en el aula: 2
            Mínimo requerido: 7                                     - - ALUMNOS SUSPENDIDOS [5] - -

            No se puede dar clase                           El alumno Fran está suspendido con un 4,70
                                                            ¡AUSENTE! El alumno Laura está suspendido con un 3,57
                                                            ¡AUSENTE! El alumno Elena está suspendido con un 0,27
                                                            ¡AUSENTE! El alumno Diego está suspendido con un 0,93
aulaFilosofia.setMateriaAula(materiaAula:"matematicas");    El alumno Javier está suspendido con un 3,67

                                                                    - - ALUMNOS AUSENTES [7] - -

            Materia profesor: filosofia
            Materia aula: matematicas                       El alumno Juan está ausente
                                                            El alumno Maria está ausente
            No se puede dar clase                           El alumno Laura está ausente
                                                            El alumno Ana está ausente
                                                            El alumno Pablo está ausente
                                                            El alumno Elena está ausente
                                                            El alumno Diego está ausente

                                                            Se puede dar clase
```

Ejercicio 06:

Por último, tendremos que generar tres clases diferenciadas para llegar a crear un cine conformado de una sala. Las clases que crearemos serán: “Sala”, “Pelicula” y “Espectador”, también un main; en la clase “Sala” crearemos unos atributos para recoger los datos de; su cantidad de filas y columnas, la película que se está reproduciendo y el precio de entrada. Para la clase “Pelicula”: título, duración; esta la desglosaré en tres más: horas, minutos y segundos; edad y director. Y, para la última clase “Espectadores”, los atributos que crearemos serán: nombre, edad y dinero. Una vez las clases sean creadas, en el main, tendremos que generar una cantidad de espectadores, junto con una película y una sala y revisar que el espectador que quiere entrar, puede. Para entrar lo que deberemos revisar será que su edad sea superior a la mínima, su dinero superior al precio de la entradas e independientemente del espectador, la película que se reproduce en la sala y la “película que se puede ver” deben ser la misma. Por último haremos una generación de texto por consola simulando que los vamos sentando uno a uno.

Clase “Espectador”

ATRIBUTOS

```java
public class Espectador {
    // ATRIBUTOS
    private String nombre;
    private int edad;
    private double dinero;
```

CONSTRUCTORES

```java
// CONSTRUCTORES
public Espectador() {
    this.nombre = "";
    this.edad = 0;
    this.dinero = 0.0;
}

public Espectador(String nombre, int edad, double dinero) {
    this.nombre = nombre;
    this.edad = edad;
    this.dinero = dinero;
}
```

MÉTODOS

```java
    // METODOS
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public double getDinero() {
        return dinero;
    }

    public void setDinero(double dinero) {
        this.dinero = dinero;
    }
}
```

Clase “Pelicula”

ATRIBUTOS

```java
public class Pelicula {
    //  ATRIBUTOS
    private String titulo;
    private String duracion;
    private int duracion_horas;
    private int duracion_minutos;
    private int duracion_segundos;
    private int edad_minima;
    private String director;
```

CONSTRUCTORES

```java
//  CONSTRUCTORES
public Pelicula() {
    this.titulo = "";
    this.duracion = "";
    this.duracion_horas = 0;
    this.duracion_minutos = 0;
    this.duracion_segundos = 0;
    this.edad_minima = 0;
    this.director = "";
}

public Pelicula(String titulo, String duracion, int edad_minima, String director) {
    this.titulo = titulo;
    this.duracion = duracion;
    String[] duracionSeparado = duracion.split(regex:":");
    if (duracionSeparado.length == 3) {
        this.duracion_horas = Integer.parseInt(duracionSeparado[0]);
        this.duracion_minutos = Integer.parseInt(duracionSeparado[1]);
        this.duracion_segundos = Integer.parseInt(duracionSeparado[2]);
    } else if (duracionSeparado.length == 2) {
        this.duracion_minutos = Integer.parseInt(duracionSeparado[0]);
        this.duracion_segundos = Integer.parseInt(duracionSeparado[1]);
    } else {
        this.duracion_minutos = Integer.parseInt(duracionSeparado[0]);
    }
    this.edad_minima = edad_minima;
    this.director = director;
}
```

MÉTODOS

```java
//  METODOS
public String getTitulo() {
    return titulo;
}

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

public String getDuracion() {
    return duracion;
}

public void setDuracion(String duracion) {
    this.duracion = duracion;
}

public int getDuracion_Horas() {
    return duracion_horas;
}

public void setDuracion_Horas(int horas) {
    this.duracion_horas = horas;
}

public int getDuracion_Minutos() {
    return duracion_minutos;
}

public void setDuracion_Minutos(int minutos) {
    this.duracion_minutos = minutos;
}

public int getDuracion_Segundos() {
    return duracion_segundos;
}

public void setDuracion_Segundos(int segundos) {
    this.duracion_segundos = segundos;
}

public int getEdad_Minima() {
    return edad_minima;
}

public void setEdad_Minima(int edad_minima) {
    this.edad_minima = edad_minima;
}

public String getDirector() {
    return director;
}

public void setDirector(String director) {
    this.director = director;
}
}
```

Clase “Sala”

ATRIBUTOS

```java
import java.util.*;

public class Sala {
    //  ATRIBUTOS
    private int filasAsientos;
    private int columnasAsientos;
    private boolean[][] asientosAsignados;
    private Pelicula peliculaReproducida;
    private double precioEntrada;
```

CONSTRUCTORES

```java
//  CONSTRUCTORES
public Sala() {
    this.filasAsientos = 0;
    this.columnasAsientos = 0;
    asientosAsignados = new boolean[0][0];
    this.peliculaReproducida = new Pelicula();
    this.precioEntrada = 0.0;
}

public Sala(int filas, int columnas, Pelicula pelicula, double precio) {
    this.filasAsientos = filas;
    this.columnasAsientos = columnas;
    asientosAsignados = new boolean[filas][columnas];
    this.peliculaReproducida = pelicula;
    this.precioEntrada = precio;
}
```

MÉTODOS

```java
//  METODOS
public int getFilas() {
    return filasAsientos;
}

public void setFilas(int filas) {
    this.filasAsientos = filas;
}

public int getColumnas() {
    return columnasAsientos;
}

public void setColumnas(int columnas) {
    this.columnasAsientos = columnas;
}

public Pelicula getPelicula() {
    return peliculaReproducida;
}

public void setPelicula(Pelicula pelicula) {
    this.peliculaReproducida = pelicula;
}

public double getPrecio() {
    return precioEntrada;
}

public void setPrecio(double precio) {
    this.precioEntrada = precio;
}

public String asignarAsiento(Espectador espectador, ArrayList<Espectador> espectadores,
HashMap<String, Espectador> espectador_asiento, ArrayList<String> posicionAsientos) {
    Random numAleatorio = new Random();
    String texto = "";
    boolean repetir = true;
    while (repetir) {
        int numFila = numAleatorio.nextInt(filasAsientos);
        int numColumna = numAleatorio.nextInt(columnasAsientos);
        if (espectadores.size()>(filasAsientos*columnasAsientos)) {
            texto = "La capacidad de la sala es menor a la cantidad de espectadores.\n " +
            "Capacidad de la sala: " + (filasAsientos*columnasAsientos) +
            "\n Número de espectadores: " + espectadores.size();
            repetir = false;
        } else if (!asientosAsignados[numFila][numColumna]) {
            int letraASCII = 64 + (numColumna + 1);
            String posicionSentado = String.valueOf(numFila + 1) + "" + String.valueOf((char) letraASCII);
            posicionAsientos.add(posicionSentado);
            texto = "\"" + espectador.getNombre()+"\" irá en el asiento "+ posicionSentado + "\n";
            espectador_asiento.put(posicionSentado, espectador);
            asientosAsignados[numFila][numColumna] = true;
            repetir = false;
        }
    }
    return texto;
}

public String puedeEntrar(Espectador espectador) {
    if (espectador.getEdad() < peliculaReproducida.getEdad_Minima()) {
        return "edad";
    } else if (espectador.getDinero() < precioEntrada) {
        return "dinero";
    } else {
        return "si";
    }
}
}
```

Main

```java
import java.util.*;

public class mainCine {
    Run | Debug
    public static void main(String[] args) {
        Espectador espectador1 = new Espectador(nombre:"Jorge", edad:25, dinero:100);
        ...
        Espectador espectador25 = new Espectador(nombre:"Lucas", edad:10, dinero:90);

        ArrayList<Espectador> espectadores = new ArrayList<>();
        espectadores.addAll(Arrays.asList(espectador1, ..., espectador25));

        Pelicula pelicula = new Pelicula(titulo:"Alicia", duracion:"120:51",
        edad_minima:15, director:"Almodobar");
        Sala sala = new Sala(filas:8, columnas:9, pelicula, precio:20.20);

        ArrayList<String> posicionAsientos = new ArrayList<>();
        StringBuilder visionFinalAsientos = new StringBuilder();
        StringBuilder asientoAsignado = new StringBuilder();
        HashMap<String, Espectador> espectador_asiento = new HashMap<>();
        for (Espectador espectador : espectadores) {
            if (sala.puedeEntrar(espectador).equals(anObject:"si")) {
                asientoAsignado.append(sala.asignarAsiento(espectador, espectadores,
                espectador_asiento, posicionAsientos));
                if (asientoAsignado.toString().equals("La capacidad de la sala es " +
                "menor a la cantidad de espectadores.\n" +
                " Capacidad de la sala: " + (sala.getFilas()*sala.getColumnas()) +
                "\n Número de espectadores: " + espectadores.size())) {
                }
            } else if (sala.puedeEntrar(espectador).equals(anObject:"dinero")) {
                System.out.println("El espectador " + espectador.getNombre() +
                " no tiene el dinero suficiente. [" + espectador.getDinero() + "€]");
            } else if (sala.puedeEntrar(espectador).equals(anObject:"edad")) {
                System.out.println("El espectador " + espectador.getNombre() +
                " no tiene la edad minima. " + espectador.getEdad() + " años");
            } else {
                System.out.println("El espectador " + espectador.getNombre() +
                " no cumple con algúnos de los requisitos");
            }
        }
        System.out.println(asientoAsignado.toString());

        crearVisionFinal(sala, visionFinalAsientos, espectador_asiento, posicionAsientos);
        imprimirLetraPorLetra(visionFinalAsientos.toString());

    }
```

```java
    public static void crearVisionFinal(Sala sala, StringBuilder visionFinalAsientos,
    HashMap<String, Espectador> espectador_asiento, ArrayList<String> posicionAsientos) {
        int fila = sala.getFilas();
        int columna = sala.getColumnas();
        int filaMostrada = fila;
        int letraAscii = 65;
        for (int i = 0; i < fila; i++) {
            visionFinalAsientos.append(str:"|");
            for (int j = 0; j < columna; j++) {
                char letraColumna = (char) letraAscii;
                String asientoActual = String.valueOf(filaMostrada) + letraColumna;
                boolean esta = false;
                for (String asientoOcupado : posicionAsientos) {
                    if (asientoActual.equals(asientoOcupado)) {
                        for (String asientoHash : espectador_asiento.keySet()) {
                            if (asientoHash.equals(asientoActual)) {
                                visionFinalAsientos.append("- - " +
                                espectador_asiento.get(asientoHash).getNombre() + " [" +
                                filaMostrada + letraColumna + "] - -|");
                                esta = true;
                            }
                        }
                    }
                }
                if (!esta) {
                    visionFinalAsientos.append("    Libre [" +
                    filaMostrada + letraColumna + "]    |");
                }
                letraAscii++;
            }
            visionFinalAsientos.append(str:"\n");
            letraAscii = 65;
            filaMostrada--;
        }
    }

    public static void imprimirLetraPorLetra(String asientos) {
        for (int i = 0; i < asientos.length(); i++) {
            char letra = asientos.charAt(i);
            System.out.print(letra);
            try {
                Thread.sleep(millis:15);
            } catch (InterruptedException e) {
                System.out.println(x:"\n\nFin del programa");
            }
        }
    }
}
```

Resultado consola

```
El espectador Sofía no tiene el dinero suficiente. [10.0?]
El espectador Lucas no tiene la edad minima. 10 años
"Jorge" irá en el asiento 5E
"Maria" irá en el asiento 6A
"Julia" irá en el asiento 4D
"Kevin" irá en el asiento 5C
"Laura" irá en el asiento 7D
"Pedro" irá en el asiento 3B
"David" irá en el asiento 2I
"Carla" irá en el asiento 7E
"Pablo" irá en el asiento 6E
"Sonia" irá en el asiento 3A
"Rubén" irá en el asiento 1D
"Paula" irá en el asiento 4C
"Oscar" irá en el asiento 3G
"Elena" irá en el asiento 6H
"Mario" irá en el asiento 8C
"Paula" irá en el asiento 7H
"Amber" irá en el asiento 1F
"Ramón" irá en el asiento 1A
"Eider" irá en el asiento 2A
"Diego" irá en el asiento 1I
"Carol" irá en el asiento 5G
"Ángel" irá en el asiento 1G
"Mikel" irá en el asiento 3C
|    Libre [8A]    |    Libre [8B]    |- - Mario [8C] - -|    Libre [8D]    |    Libre [8E]    |    Libre [8F]    |    Libre [8G]    |    Libre [8H]    |    Libre [8I]    |
|    Libre [7A]    |    Libre [7B]    |    Libre [7C]    |- - Laura [7D] - -|- - Carla [7E] - -|    Libre [7F]    |    Libre [7G]    |- - Paula [7H] - -|    Libre [7I]    |
|- - Maria [6A] - -|    Libre [6B]    |    Libre [6C]    |    Libre [6D]    |- - Pablo [6E] - -|    Libre [6F]    |    Libre [6G]    |- - Elena [6H] - -|    Libre [6I]    |
|    Libre [5A]    |    Libre [5B]    |- - Kevin [5C] - -|    Libre [5D]    |- - Jorge [5E] - -|    Libre [5F]    |- - Carol [5G] - -|    Libre [5H]    |    Libre [5I]    |
|    Libre [4A]    |    Libre [4B]    |- - Paula [4C] - -|- - Julia [4D] - -|    Libre [4E]    |    Libre [4F]    |    Libre [4G]    |    Libre [4H]    |    Libre [4I]    |
|- - Sonia [3A] - -|- - Pedro [3B] - -|- - Mikel [3C] - -|    Libre [3D]    |    Libre [3E]    |    Libre [3F]    |- - Oscar [3G] - -|    Libre [3H]    |    Libre [3I]    |
|- - Eider [2A] - -|    Libre [2B]    |    Libre [2C]    |    Libre [2D]    |    Libre [2E]    |    Libre [2F]    |    Libre [2G]    |    Libre [2H]    |- - David [2I] - -|
|- - Ramón [1A] - -|    Libre [1B]    |    Libre [1C]    |- - Rubén [1D] - -|    Libre [1E]    |- - Amber [1F] - -|- - Ángel [1G] - -|    Libre [1H]    |- - Diego [1I] - -|
```

- Para entender mejor cómo hacer el ejercicio 4:
  Ecuaciones de segundo grado
- Para ver API's de java:
  Java API
- El copiloto de confianza como ayuda siempre, importante:
  ChatGPT