

**Instituto Tecnológico de
Orizaba**



**TECNOLÓGICO
NACIONAL DE MÉXICO**

CARRERA

ING. INFORMATICA

ASIGNATURA

PROGRAMACION ORIENTADA A OBJETOS

TEMA 3

MÉTODOS

ALUMN@

MAYTE MELLADO HUERTA

NO.CONTROL

21010202

GRUPO

2a3B

FECHA DE ENTREGA

17/04/2023

INTRODUCCION

El tema principal de este reporte es sobre los métodos en la programación, y se abordan varios subtemas relacionados con los métodos. Estos subtemas incluyen:

3.1 Definición de un método: Se describe qué es un método en el contexto de la programación y cómo se define.

3.2 Estructura de un método: Se explica la estructura básica de un método, incluyendo su nombre, lista de parámetros, tipo de retorno y cuerpo del método.

3.3 Valor de retorno: Se discute cómo un método puede devolver un valor como resultado de su ejecución.

3.4 Declaración de un método: Se detallan dos tipos de métodos: de clase y de instancia, y cómo se declaran en un lenguaje de programación.

3.5 Ámbito y tiempo de vida de variables: Se aborda cómo las variables declaradas en un método tienen un alcance y una duración específica durante la ejecución del programa.

3.6 Argumentos y paso de parámetros: Se explica cómo los métodos pueden recibir argumentos como parámetros y cómo se pasan esos valores al cuerpo del método.

3.7 Puntero this: Se describe el puntero "this" en la programación orientada a objetos, que hace referencia al objeto actual en el cual se está ejecutando el método.

3.8 Sobrecarga de métodos: Se discute cómo es posible tener varios métodos con el mismo nombre pero con diferentes parámetros, lo que se conoce como sobrecarga de métodos.

3.9 Constructores y destructores: Se introduce el concepto de constructores y destructores, que son métodos especiales utilizados para la creación y destrucción de objetos en la programación orientada a objetos.

En resumen, este reporte aborda diversos aspectos relacionados con los métodos en la programación, incluyendo su definición, estructura, valor de retorno, declaración, ámbito de variables, paso de parámetros, uso del puntero "this", sobrecarga de métodos y el uso de constructores y destructores.

COMPETENCIA ESPECIFICA

Comprende y aplica los diferentes tipos de métodos, tomando en cuenta el ámbito y tiempo de vida de los datos durante la ejecución de un programa.

MARCO TEÓRICO:

3.1 Definición de un método: En la programación, un método es una secuencia de instrucciones o bloques de código que realiza una tarea específica. Los métodos son utilizados para encapsular la lógica y funcionalidad de un programa, permitiendo su reutilización y modularidad.

3.2 Estructura de un método: Un método generalmente consta de un nombre, una lista de parámetros (opcional), un tipo de retorno (opcional) y un cuerpo. El nombre del método es único y sirve para invocar su ejecución. Los parámetros son valores que se pasan al método para que los utilice en su procesamiento. El tipo de retorno especifica el tipo de dato que el método devuelve como resultado de su ejecución. El cuerpo del método contiene las instrucciones que se ejecutan cuando el método es invocado.

3.3 Valor de retorno: Un método puede devolver un valor como resultado de su ejecución. Este valor puede ser de un tipo de dato específico, como entero, flotante, cadena, booleano, entre otros. El valor de retorno puede ser utilizado en el programa para tomar decisiones, realizar cálculos u otras operaciones.

3.4 Declaración de un método: En la programación orientada a objetos, existen dos tipos de métodos: de clase y de instancia. Los métodos de clase son aquellos que

pertenecen a la clase en sí misma, mientras que los métodos de instancia son aquellos que pertenecen a una instancia o objeto específico de la clase. Los métodos se declaran con una firma que incluye el nombre del método, la lista de parámetros (si los tiene), el tipo de retorno (si lo tiene) y el cuerpo del método.

3.5 Ámbito y tiempo de vida de variables: Las variables declaradas dentro de un método tienen un alcance y una duración específica durante la ejecución del programa. El alcance se refiere a la parte del programa donde la variable es accesible, y la duración se refiere al tiempo en el cual la variable existe en la memoria. Las variables locales son aquellas declaradas dentro de un método y solo son accesibles dentro del mismo. Las variables de instancia son aquellas declaradas en una clase y son accesibles desde cualquier método de esa clase.

3.6 Argumentos y paso de parámetros: Los métodos pueden recibir argumentos como parámetros, que son valores que se pasan al método al momento de invocarlo. Estos argumentos son utilizados por el método en su procesamiento. Los parámetros pueden ser de diferentes tipos y se pasan al método en el orden especificado en su declaración. El paso de parámetros puede ser por valor, por referencia o por referencia constante, dependiendo del lenguaje de programación utilizado.

3.7 Puntero this: En la programación orientada a objetos, el puntero "this" es una referencia al objeto actual en el cual se está ejecutando el método. Permite acceder a los miembros y métodos del objeto dentro del mismo método. Se utiliza para diferenciar entre variables locales y variables de instancia que tienen el mismo nombre, y para referirse a los miembros del objeto en el cual se está ejecutando el método.

3.8 Sobrecarga de métodos: La sobrecarga de métodos permite tener varios métodos con el mismo nombre, pero con diferentes firmas, es decir, con diferentes listas de parámetros o tipos de retorno. Esto permite tener métodos con funcionalidades similares, pero con diferentes formas de invocarlos, lo cual brinda flexibilidad y versatilidad en la programación.

3.9 Constructores y destructores: Los constructores son métodos especiales que se utilizan para inicializar los objetos de una clase cuando son creados. Los destructores son métodos especiales que se utilizan para realizar acciones de limpieza o liberación de recursos cuando un objeto es destruido o eliminado de la memoria. Estos métodos son parte importante de la programación orientada a objetos y son utilizados para asegurar la correcta creación y destrucción de objetos en un programa.

MATERIAL Y EQUIPO:

- ♥ Computadora
- ♥ NetBeans IDE 8.3
- ♥ PDF y presentaciones dadas por el docente

DESARROLLO DE LA PRACTICA

1. Definición de un método: Describir conceptualmente qué es un método, su propósito y cómo se utiliza en la programación.
2. Estructura de un método: Explicar los elementos que componen la estructura de un método, como el nombre, los parámetros, el tipo de retorno, la visibilidad y otros aspectos relacionados con la sintaxis y organización de un método en un lenguaje de programación específico.
3. Valor de retorno: Explicar cómo se define y utiliza el valor de retorno en un método, su significado y su importancia en la comunicación de datos entre métodos y en la obtención de resultados de un proceso computacional.
4. Declaración de un método: Describir los diferentes tipos de métodos, tanto de clase como de instancia, en la programación orientada a objetos, incluyendo su sintaxis, características y aplicaciones en la resolución de problemas.

5. **Ámbito y tiempo de vida de variables:** Explicar cómo se gestionan las variables en el contexto de un método, incluyendo su ámbito de visibilidad y su tiempo de vida durante la ejecución de un programa.
6. **Argumentos y paso de parámetros:** Describir cómo se pasan argumentos y parámetros a un método, así como los diferentes tipos de paso de parámetros, como por valor, por referencia o por puntero, y su impacto en la manipulación y modificación de datos en un programa.
7. **Puntero "this":** Explicar el concepto de "this" en la programación orientada a objetos, su significado y uso para referirse a la instancia actual de un objeto y acceder a sus propiedades y métodos.
8. **Sobrecarga de métodos:** Describir la sobrecarga de métodos, que permite definir varios métodos con el mismo nombre pero con diferentes parámetros en un mismo objeto o clase, y cómo se resuelve la selección del método adecuado durante la ejecución de un programa.
9. **Constructores y destructores:** Explicar el concepto de constructores y destructores en la programación orientada a objetos, su papel en la creación y destrucción de objetos, así como su sintaxis y uso en la inicialización y liberación de recursos en un programa.

A continuación, se presentarán los diagramas de clase de todos los programas realizados:

| Alfombra |
|--|
| - double largo - double ancho |
| + Alfombra(double largo, double ancho) + double getArea() |

| Salon |
|----------------------------------|
| - double largo - double ancho |

- + Salon(double largo, double ancho)
- + double getArea()
- + double getAreaNoCubierta(Alfombra[] alfombras)

| Empleado |
|---|
| <ul style="list-style-type: none"> - int cveEmpleado - String nombEmp - byte edadEmp - byte hrstrabEmp - double pagohrsEmp - static byte Conta - static short inicial |
| <ul style="list-style-type: none"> + Empleado(String nombEmp,byte edadEmp,byte hrstrabEmp, double pagohrsEmp) + int getCveEmpleado + void setCveEmpleado(int cveEmpleado) + String getNombEmp + void setNombEmp(String nombEmp) + byte getEdadEmp + void setEdadEmp(byte edadEmp) + byte getHrstrabEmp + void setHrstrabEmp(byte hrstrabEmp) + getPagohrsEmp + void setPagohrsEmp(double pagohrsEmp) + static byte getConta + static void setConta(byte Conta) + static short getInicial() + static void setInicial(short inicial) |

```

+ String toString
+ static Empleado mayorEdad(Empleado
emp1,Empleado emp2)
+ static Empleado
mayorHrsTrabajadas(Empleado
emp1,Empleado emp2)

```

| Pelicula |
|---|
| <pre> - String nombre - String director - String genero - int duracion - short año - byte calif + static final String GenAccion = "ACCION",GenComedia = "COMEDIA",GenDrama = "DRAMA",GenSuspenso = "SUSPENSO" + static final int Min60 = 60,Min115 = 115,Min120 = 120,Min150 = 150,Min180 = 180,Min191 = 191,Min210 = 210,Min240 = 240,Min270 = 270; </pre> |
| <pre> + Pelicula(String nombre, String director, String genero, int duracion,short año,byte calif) + String getNombre - void setNombre(String nombre) + String getDirector - void setDirector(String director) + String getGenero - void setGenero(String genero) + int getDuracion - void setDuracion(int duracion) + short getAño - void setAño(short año) + byte getCalif - void setCalif(byte calif) - boolean esPeliculaEpica - String calcularValoracion </pre> |

- boolean esSimilar(Pelicula Similar)
- + void imprimirCartel
- + String msjeTrueFalseePelículaEpica
- + String SimilarTrueFalse(Película Película2)
- + String calcularValoracionn

| ConversorMetros |
|---|
| <ul style="list-style-type: none"> - double metros; - final double METROS_CM = 100; - final double METROS_MILIM = 1000; - final double METROS_PULGADAS = 39.37; - final double METROS_PIES = 3.28; - final double METROS_YARDAS = 1.09361; |
| <ul style="list-style-type: none"> + ConversorMetros(double metros) + double getMetros() + void setMetros(double metros) - double MetrosACentimetros() - double MetrosAMilimetros() - double MetrosAPulgadas() - double MetrosAPies() - double MetrosAYardas() + void Conversor (int opcion) |

| RestPedido |
|---|
| <ul style="list-style-type: none"> - String PrimerPlat - String SegundoPlat - String Bebida - String Postre |
| <ul style="list-style-type: none"> + RestPedido(String primerPlat, String bebida) + RestPedido(String primerPlat, String segundoPlat, String bebida) + RestPedido(String primerPlat, String segundoPlat, String bebida, String postre) + String getPrimerPlat + void setPrimerPlat(String primerPlat) + String getSegundoPlat + void setSegundoPlat(String segundoPlat) + String getBebida + void setBebida(String bebida) + String getPostre() |

```

+ void setPostre(String postre)
- double CostoPrimerPlato(String primerPlat)
- double CostoSegundoPlato(String
segundoPlat)
- double CostoBebida(String bebida)
- double CostoPostre(String postre)
+ void calcularPedidos (String primerPlat,
String bebida)
+ void CalcularPedidos (String primerPlat,
String segundoPlat, String bebida)
+ void calcularPedidos(String primerPlat,
String segundoPlat, String bebida, String
postre)

```

ArticuloCientifico

```

- String titulo;
- String autor;
- String[] palabrasClaves;
- String publicacion;
- int año;
- String resumen;

```

```

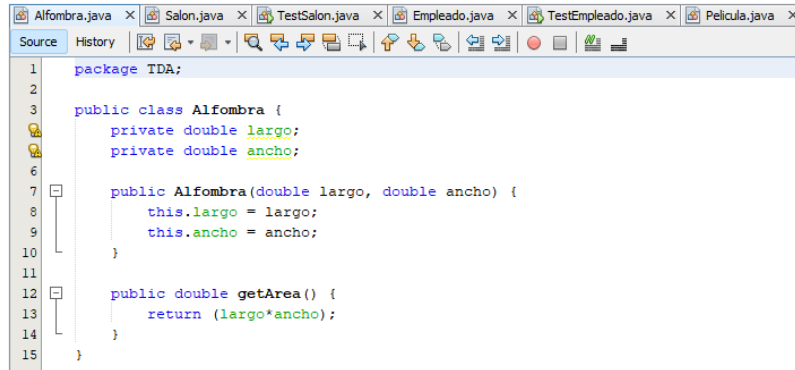
+ ArticuloCientifico(String titulo, String autor)
+ public ArticuloCientifico(String titulo, String
autor, String[] palabrasClaves, String
publicacion, int año)
+ ArticuloCientifico(String titulo, String autor,
String[] palabrasClaves, String publicacion,
int año, String resumen)
+String getTitulo()
+ void setTitulo(String titulo)
+ String getAutor()
+ void setAutor(String autor)
+ String[] getPalabrasClaves()
+ void setPalabrasClaves(String[]
palabrasClaves)
+ String getPublicacion()
+ void setPublicacion(String publicacion)
+ int getAño()
+ void setAño(int año)
+ String getResumen()
+ void setResumen(String resumen)
+ void imprimirArticulo()

```

| AsistenteBoda |
|--|
| <ul style="list-style-type: none"> - String nombre - byte edad - String sexo - String estadoCivil - static byte total - static byte totalHombres - static byte totalMujeres - static byte totalMayores - static byte totalMenores - static byte totalSolteros - static byte totalCasados - static byte totalViudos - static byte totalDivorciados - static byte totalSeparados |
| <ul style="list-style-type: none"> + AsistenteBoda(String nombre, byte edad, String sexo, String estadoCivil) + String getNombre + void setNombre(String nombre) + byte getEdad() + void setEdad(byte edad) + String getSexo + void setSexo(String sexo) + String getEstadoCivil + void setEstadoCivil(String estadoCivil) + byte getTotalHombres + void setTotalHombres(byte totalHombres) + byte getTotalMujeres + void setTotalMujeres(byte totalMujeres) + byte getTotalCasados + void setTotalCasados(byte totalCasados) + byte getTotalSolteros + void setTotalSolteros(byte totalSolteros) + byte getTotalViudos + void setTotalViudos(byte totalViudos) + byte getTotalSeparados + void setTotalSeparados(byte totalSeparados) + byte getTotalDivorciados + void setTotalDivorciados(byte totalDivorciados) + byte getTotalMayores + void setTotalMayores(byte totalMayores) + void setTotalMenores(byte totalMenores) + static byte getTotal() + static void setTotal(byte total) |

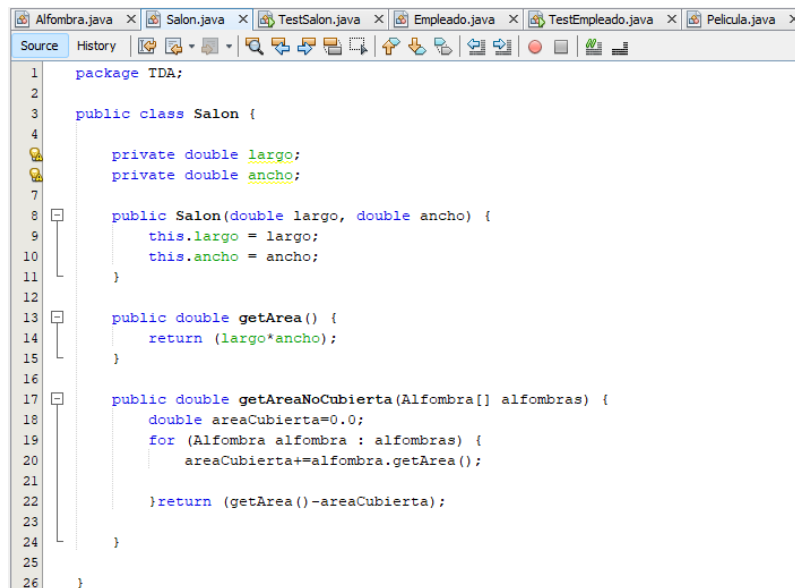
- + void registrarAsistente
- + void mostrarEstadisticas

RESULTADOS



```
1 package TDA;
2
3 public class Alfombra {
4     private double largo;
5     private double ancho;
6
7     public Alfombra(double largo, double ancho) {
8         this.largo = largo;
9         this.ancho = ancho;
10    }
11
12    public double getArea() {
13        return (largo*ancho);
14    }
15 }
```

La clase tiene dos campos privados largo y ancho para almacenar la longitud y el ancho de la alfombra respectivamente.



```
1 package TDA;
2
3 public class Salon {
4
5     private double largo;
6     private double ancho;
7
8     public Salon(double largo, double ancho) {
9         this.largo = largo;
10        this.ancho = ancho;
11    }
12
13    public double getArea() {
14        return (largo*ancho);
15    }
16
17    public double getAreaNoCubierta(Alfombra[] alfombras) {
18        double areaCubierta=0.0;
19        for (Alfombra alfombra : alfombras) {
20            areaCubierta+=alfombra.getArea();
21        }
22        return (getArea()-areaCubierta);
23    }
24 }
25
26 }
```

Este código define una clase Salon que representa un salón con métodos para obtener su área y calcular el área no cubierta por alfombras.

```
Alfombra.java x Salon.java x TestSalon.java x Empleado.java x TestEmpleado.java x Pelicula.java x
Source History
1 package TestClasesObjetos;
2
3 import EntradaSalida.Tools;
4 import TDA.Salon;
5 import TDA.Alfombra;
6
7 public class TestSalon {
8     public static void main(String[] args) {
9         Salon salon = new Salon(7.0, 6.5);
10        Alfombra alfombral = new Alfombra(3.8, 4.6);
11        Alfombra alfombra2 = new Alfombra(4.5, 2.3);
12
13
14        double areaCubierta = (alfombral.getArea()+alfombra2.getArea());
15        double areaNoCubierta = (salon.getArea()-areaCubierta);
16
17        Tools.imprimePantalla("Area del salón es: " + salon.getArea()+"\n"
18            +"Area cubierta por las alfombras es: " + areaCubierta+"\n"
19            +"Area no cubierta por las alfombras es: " + areaNoCubierta);
20
21    }
22 }
23
24
25
```

Este código prueba el cálculo del área total del salón, el área total cubierta por las alfombras y el área no cubierta por las alfombras utilizando las clases Salon y Alfombra, y muestra los resultados en pantalla.

```
Alfombra.java x Salon.java x TestSalon.java x Empleado.java x TestEmpleado.java x Pelicula.java x
Source History
1 package TDA;
2
3 public class Empleado {
4
5     private int cveEmpleado;
6     private String nombEmp;
7     private byte edadEmp;
8     private byte hrstrabEmp;
9     private double pagohrsEmp;
10    private static byte Conta=0;
11    private static short inicial=1000;
12
13    public Empleado(String nombEmp,byte edadEmp,byte hrstrabEmp,
14        double pagohrsEmp){
15        this.cveEmpleado=inicial;
16        this.nombEmp=nombEmp;
17        this.edadEmp=edadEmp;
18        this.hrstrabEmp=hrstrabEmp;
19        this.pagohrsEmp=pagohrsEmp;
20        Conta++;
21        inicial+=2;
22    }
23
24
25    public int getCveEmpleado() {
26        return cveEmpleado;
27    }
28
29    public void setCveEmpleado(int cveEmpleado) {
30        this.cveEmpleado = cveEmpleado;
31    }
32
33    public String getNombEmp() {
34        return nombEmp;
35    }
36
37 }
```

```

36
37 public void setNombEmp(String nombEmp) {
38     this.nombEmp = nombEmp;
39 }
40
41 public byte getEdadEmp() {
42     return edadEmp;
43 }
44
45 public void setEdadEmp(byte edadEmp) {
46     this.edadEmp = edadEmp;
47 }
48
49 public byte getHrstrabEmp() {
50     return hrstrabEmp;
51 }
52
53 public void setHrstrabEmp(byte hrstrabEmp) {
54     this.hrstrabEmp = hrstrabEmp;
55 }
56
57 public double getPagohrsEmp() {
58     return pagohrsEmp;
59 }
60
61 public void setPagohrsEmp(double pagohrsEmp) {
62     this.pagohrsEmp = pagohrsEmp;
63 }
64
65 public static byte getConta() {
66     return Conta;
67 }
68
69 public static void setConta(byte Conta) {
70     Empleado.Conta = Conta;
71 }
72
73 public static short getInicial() {
74     return inicial;
75 }
76
77 public static void setInicial(short inicial) {
78     Empleado.inicial = inicial;
79 }
80
81
82 @Override
83 public String toString() {
84     return "Empleado{" + "cveEmpleado=" + cveEmpleado + ", nombEmp="
85         + nombEmp + ", edadEmp=" + edadEmp + ", hrstrabEmp="
86         + hrstrabEmp + ", pagohrsEmp=" + pagohrsEmp + '}';
87 }
88
89
90 public static Empleado mayorEdad(Empleado emp1, Empleado emp2) {
91     return (emp1.edadEmp >= emp2.edadEmp) ? emp1 : emp2;
92 }
93
94 public static Empleado mayorHrsTrabajadas(Empleado emp1, Empleado emp2) {
95     return (emp1.hrstrabEmp >= emp2.hrstrabEmp) ? emp1 : emp2;
96 }
97 }

```

El código define una clase llamada Empleado que representa a un empleado con varios atributos, como clave de empleado, nombre, edad, horas trabajadas y pago por hora.

```

1 package TestClasesObjetos;
2
3 import TDA.Empleado;
4 import EntradaSalida.Tools;
5
6 public class TestEmpleado {
7
8     public static void main(String arg[]){
9
10         Empleado Carlos=new Empleado("Carlos Perez", (byte)25, (byte)40, 180.50);
11         Empleado Sonia=new Empleado("Sonia Alvarez", (byte)19, (byte)45, 180.50);
12         Empleado Alma=new Empleado("Alma Alvarez", (byte)19, (byte)45, 180.50);
13
14         String cad=" Empleado con mayor edad: "+Empleado.mayorEdad(Carlos, Sonia)+
15             "\n";
16         cad+="Empleado que trabaja mas horas: "+Empleado.mayorHrsTrabajadas(Carlos,
17             Sonia);
18         Tools.imprimePantalla(cad+"\n"+Carlos.toString()+"\n"+Sonia.toString()+
19             "\n"+Alma.toString()+"\n Total de objetos creados:"+Empleado.getConta());
20     }
21 }
22

```

El código crea tres objetos de la clase Empleado con diferentes atributos: Carlos, Sonia y Alma. Luego, se llama a los métodos estáticos mayorEdad y mayorHrsTrabajadas de la clase Empleado para comparar los empleados en base a su edad y horas trabajadas, respectivamente, y se almacena el resultado en una cadena de texto (cad).

```
...ava TestEmpleado.java x Pelicula.java x TestPelicula.java x ConversorMetros.java x TestConversorMetros.j
Source History
1 package TDA;
2
3 import EntradaSalida.Tools;
4
5 public class Pelicula {
6
7     private String nombre;
8     private String director;
9     private String genero;
10    private int duracion;
11    private short año;
12    private byte calif;
13
14    public static final String GenAccion = "ACCION", GenComedia = "COMEDIA",
15        GenDrama = "DRAMA", GenSuspense = "SUSPENSO";
16    public static final int Min60 = 60, Min115 = 115, Min120 = 120, Min150 = 150,
17        Min180 = 180, Min191 = 191, Min210 = 210, Min240 = 240, Min270 = 270;
18
19
20    public Pelicula(String nombre, String director, String genero, int duracion,
21        short año, byte calif) {
22        this.nombre = nombre;
23        this.director = director;
24        this.genero = genero;
25        this.duracion = duracion;
26        this.año = año;
27        this.calif = calif;
28    }
29
30    public String getNombre() {
31        return nombre;
32    }
33
34    private void setNombre(String nombre) {
35        this.nombre = nombre;
36    }
37
38    public String getDirector() {
39        return director;
40    }
41
42    private void setDirector(String director) {
43        this.director = director;
44    }
45
46    public String getGenero() {
47        return genero;
48    }
49
50    private void setGenero(String genero) {
51        this.genero = genero;
52    }
53
54    public int getDuracion() {
55        return duracion;
56    }
57
58    private void setDuracion(int duracion) {
59        this.duracion = duracion;
60    }
61
62    public short getAño() {
63        return año;
64    }
65
66    private void setAño(short año) {
67        this.año = año;
68    }
69
```



```

70 public byte getCalif() {
71     return calif;
72 }
73
74 private void setCalif(byte calif) {
75     this.calif = calif;
76 }
77
78 private boolean esPeliculaEpica() {
79     return duracion >= 180;
80 }
81
82 private String calcularValoracion() {
83     if (calif >= 0 && calif <= 2) {
84         return "Muy mala";
85     } else if (calif > 2 && calif <= 5) {
86         return "Mala";
87     } else if (calif > 5 && calif < 7) {
88         return "Regular";
89     } else if (calif >= 7 && calif <= 8) {
90         return "Buena";
91     } else if (calif > 8 && calif <= 10) {
92         return "Excelente";
93     } else {
94         return "Valoración inválida";
95     }
96 }
97
98 private boolean esSimilar(Pelicula Pelicula2) {
99     return (this.genero.equalsIgnoreCase(Pelicula2.genero) &&
100         this.calif == Pelicula2.calif);
101 }
102
103 public void imprimirCartel() {
104     String Valoracion= Tools.imprimeFrecuencia(calif);
105     Tools.imprimePantalla("-----" + nombre + " -----"+"\n"
106         +Valoracion+"\n Director: " + director+"\n Género: " + genero+
107         "\n Duración: " + duracion + " minutos"+" \n Año: " + año+
108         "\n Calificación: " + calif);
109 }
110
111 public String msjeTrueFalseesPeliculaEpica() {
112     return (esPeliculaEpica())?"Pelicula épica":"No fue Pelicula Epica";
113 }
114
115 public String SimilarTrueFalse(Pelicula Pelicula2) {
116     return esSimilar(Pelicula2)?"Pelicula Similar":"No fue Pelicula Similar";
117 }
118
119 public String calcularValoracionn(){
120     return calcularValoracion();
121 }
122 }

```

La clase tiene métodos para acceder y modificar los atributos, así como realizar algunas operaciones relacionadas con la película, como verificar si es una película épica, calcular su valoración y comparar si es similar a otra película en términos de género y calificación. Además, la clase tiene constantes para definir géneros de películas y duraciones mínimas.

```

...ave TestEmpleado.java x Pelicula.java x TestPelicula.java x ConversorMetros.java x TestConversorMetros
Source History
1 package TestClasesObjetos;
2
3 import TDA.Pelicula;
4 import EntradaSalida.Tools;
5
6 public class TestPelicula {
7     public static void main(String[] args) {
8         Pelicula peli1 = new Pelicula("Gandhi ", "Richard Attenborough",
9             Pelicula.GenDrama, Pelicula.Min191, (short) 1982, (byte)8);
10        Pelicula peli2 = new Pelicula("Thor ", "Kenneth Branagh",
11            Pelicula.GenAccion, Pelicula.Min115, (short) 2011, (byte)7 );
12
13        peli1.imprimirCartel();
14        peli2.imprimirCartel();
15
16        Tools.imprimePantalla("¿Es " + peli1.getNombre()+" una película épica? "
17            + peli1.msjeTrueFalseePeliculaEpica());
18        Tools.imprimePantalla("¿Es " + peli2.getNombre()+" una película épica? "
19            + peli2.msjeTrueFalseePeliculaEpica());
20
21        Tools.imprimePantalla("Valoración de " + peli1.getNombre() + ": "
22            + peli1.calcularValoracionn());
23        Tools.imprimePantalla("Valoración de " + peli2.getNombre() + ": "
24            + peli2.calcularValoracionn());
25
26        Tools.imprimePantalla("¿Son similares " + peli1.getNombre() + " y "
27            + peli2.getNombre() + "? " +peli1.SimilarTrueFalse(peli2) );
28    }
29
30
31 }

```

El código es una prueba de la clase Pelicula.

```

...ava TestEmpleado.java x Pelicula.java x TestPelicula.java x ConversorMetros.java x TestConversorMetros
Source History
1 package TDA;
2
3 import EntradaSalida.Tools;
4
5 public class ConversorMetros {
6
7     private double metros;
8     private final double METROS_CM = 100;
9     private final double METROS_MILIM = 1000;
10    private final double METROS_PULGADAS = 39.37;
11    private final double METROS_PIES = 3.28;
12    private final double METROS_YARDAS = 1.09361;
13
14    public ConversorMetros(double metros) {
15        this.metros = metros;
16    }
17
18
19    public double getMetros() {
20        return metros;
21    }
22
23    public void setMetros(double metros) {
24        this.metros = metros;
25    }
26
27    private double MetrosACentimetros() {
28        return (metros * METROS_CM);
29    }
30
31    private double MetrosAMilimetros() {
32        return (metros * METROS_MILIM);
33    }
34
35    private double MetrosAPulgadas() {
36        return (metros * METROS_PULGADAS);
37    }
38
39    private double MetrosAPies() {
40        return (metros * METROS_PIES);
41    }
42
43    private double MetrosAYardas() {
44        return (metros / METROS_YARDAS);
45    }
46
47    public void Conversor (int opcion){
48
49        do {
50            opcion = Tools.leerInt("Selecciona una opción:\n1. "
51                + "Convertir a centímetros\n2. Convertir a milímetros\n"
52                + "3. Convertir a pulgadas\n4. Convertir a pies\n"
53                + "5. Convertir a yardas\n6. Salir");
54            switch(opcion) {
55                case 1:
56                    Tools.imprimePantalla(metros + " metros = "
57                        + MetrosACentimetros() + " centímetros");
58                    break;
59                case 2:
60                    Tools.imprimePantalla(metros + " metros = "
61                        + MetrosAMilimetros() + " milímetros");
62                    break;
63                case 3:
64                    Tools.imprimePantalla(metros + " metros = "
65                        + MetrosAPulgadas() + " pulgadas");
66                    break;
67                case 4:
68                    Tools.imprimePantalla(metros + " metros = "
69                        + MetrosAPies() + " pies");
70                    break;
71                case 5:
72                    Tools.imprimePantalla(metros + " metros = "
73                        + MetrosAYardas() + " yardas");
74                    break;

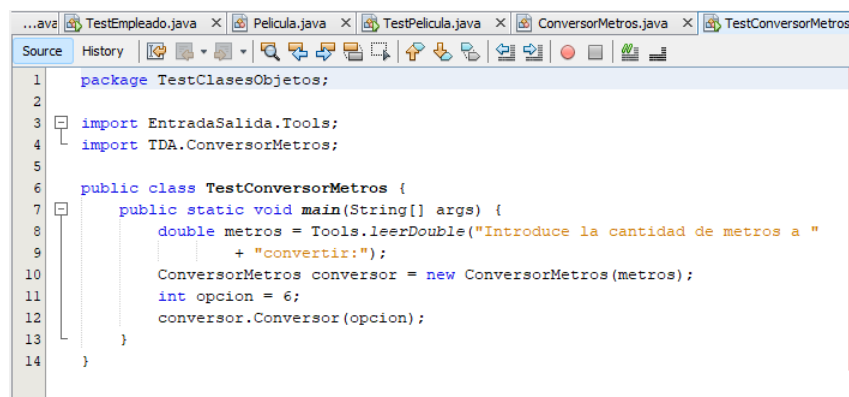
```

```

75         case 6:
76             Tools.imprimePantalla("Saliendo ...");
77             break;
78         default:
79             Tools.salidaError("Opción inválida");
80     }
81     } while (opcion != 6);
82 }
83 }
84 }

```

El código define una clase llamada `ConversorMetros` que representa un objeto para convertir una longitud en metros a diferentes unidades de medida, como centímetros, milímetros, pulgadas, pies y yardas.



```

1 package TestClasesObjetos;
2
3 import EntradaSalida.Tools;
4 import TDA.ConversorMetros;
5
6 public class TestConversorMetros {
7     public static void main(String[] args) {
8         double metros = Tools.leerDouble("Introduce la cantidad de metros a "
9             + "convertir:");
10        ConversorMetros conversor = new ConversorMetros(metros);
11        int opcion = 6;
12        conversor.Conversor(opcion);
13    }
14 }

```

El código define una clase de prueba llamada `TestConversorMetros` que utiliza la clase `ConversorMetros` para realizar conversiones de longitud de metros a diferentes unidades de medida.

```

...ave TestEmpleado.java x Pelicula.java x TestPelicula.java x ConversorMetros.java x TestConversorMetros.java
Source History
1 package TDA;
2
3 import EntradaSalida.Tools;
4
5 public class RestPedido {
6     private String PrimerPlat;
7     private String SegundoPlat;
8     private String Bebida;
9     private String Postre;
10    public RestPedido(String primerPlat, String bebida) {
11        this.PrimerPlat = primerPlat;
12        this.Bebida = bebida;
13    }
14    public RestPedido(String primerPlat, String segundoPlat, String bebida){
15        this.PrimerPlat = primerPlat;
16        this.SegundoPlat = segundoPlat;
17        this.Bebida = bebida;
18    }
19    public RestPedido(String primerPlat, String segundoPlat, String bebida,
20        String postre) {
21        this.PrimerPlat = primerPlat;
22        this.SegundoPlat = segundoPlat;
23        this.Bebida = bebida;
24        this.Postre = postre;
25    }
26    public String getPrimerPlat() {
27        return PrimerPlat;
28    }
29    public void setPrimerPlat(String primerPlat) {
30        PrimerPlat = primerPlat;
31    }
32    public String getSegundoPlat() {
33        return SegundoPlat;
34    }
35    public void setSegundoPlat(String segundoPlat) {
36        SegundoPlat = segundoPlat;
37    }
38    public String getBebida() {
39        return Bebida;
40    }
41    public void setBebida(String bebida) {
42        Bebida = bebida;
43    }
44    public String getPostre() {
45        return Postre;
46    }
47    public void setPostre(String postre) {
48        Postre = postre;
49    }
50
51    private double CostoPrimerPlato(String primerPlat) {
52        double total = 0;
53        switch (primerPlat) {
54            case "Crema de espinacas": total = 75.0; break;
55            case "Ensalada de verduras": total = 95.0; break;
56            case "Crema de brócoli": total = 70.0; break;
57            case "Caldo tlalpeño": total = 123.90; break;
58            case "Sopa mixteca": total = 99.90; break; }
59        return total;
60    }
61
62    private double CostoSegundoPlato(String segundoPlat) {
63        double total=0;
64        switch (segundoPlat) {
65            case "Filete de Pescado": total = 80.0; break;
66            case "Milanesa de Pollo": total = 70.0; break;
67            case "Bistec a la mexicana": total = 85.0; break;
68            case "Pollo en escabeche": total = 75.0; break;
69            case "Carne asada": total = 90.0; break;
70            case "Lomo relleno": total = 95.0; break;
71            case "Pollo a la plancha": total = 80.0; break; }
72        return total;
73    }
74

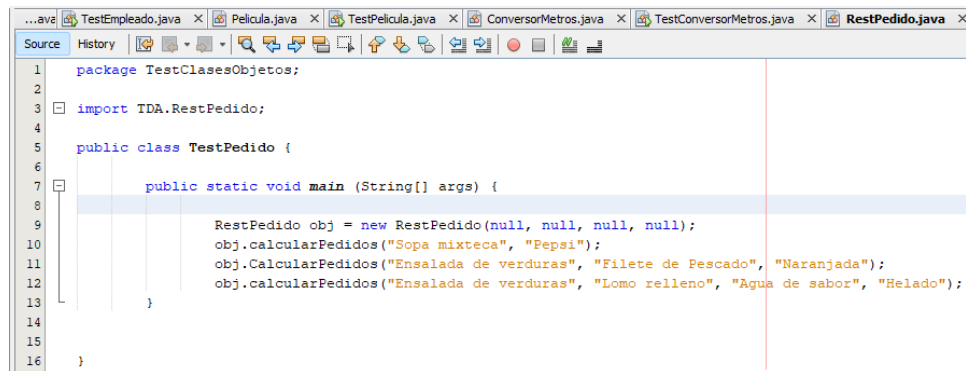
```

```

73 private double CostoBebida(String bebida) {
74     double total=0;
75     switch (bebida) {
76         case "Coca Cola":
77         case "Pepsi": total += 20.0; break;
78         case "Naranjada": case "Limonada":
79         case "Agua de sabor": total += 15.0; break;
80     }
81     return total;
82 }
83 private double CostoPostre(String postre) {
84     double total=0;
85     switch (postre) {
86         case "Pastel helado": total += 40.0; break;
87         case "Helado": total += 30.0; break;
88         case "Fresas con crema": total += 35.0; break;
89         case "Plátanos fritos": total += 40.0; break;
90         case "Flan casero": total += 25.0; break;
91         case "Gelatina": total += 20.0; break; default: break; }
92     return total;
93 }
94 public void calcularPedidos (String primerPlat, String bebida) {
95     double total =CostoPrimerPlato(primerPlat)+CostoBebida(bebida);
96     Tools.imprimeSalida("Total a pagar"+total);
97 }
98 public void CalcularPedidos (String primerPlat, String segundoPlat,
99     String bebida) {
100     double total =CostoPrimerPlato(primerPlat)+CostoSegundoPlato
101     (segundoPlat)+CostoBebida(bebida);
102     Tools.imprimeSalida("Total a pagar"+total);
103 }
104 public void calcularPedidos(String primerPlat, String segundoPlat,
105     String bebida, String postre) {
106     double total =CostoPrimerPlato(primerPlat)+CostoSegundoPlato
107     (segundoPlat)+CostoBebida(bebida)+CostoPostre(postre);
108     Tools.imprimeSalida("Total a pagar"+total);
109 }
110 }

```

El código define una clase llamada "RestPedido" que representa un pedido en un restaurante.



```

1 package TestClasesObjetos;
2
3 import TDA.RestPedido;
4
5 public class TestPedido {
6
7     public static void main (String[] args) {
8
9         RestPedido obj = new RestPedido(null, null, null, null);
10        obj.calcularPedidos("Sopa mixteca", "Pepsi");
11        obj.CalcularPedidos("Ensalada de verduras", "Filete de Pescado", "Naranjada");
12        obj.calcularPedidos("Ensalada de verduras", "Lomo relleno", "Agua de sabor", "Helado");
13    }
14
15 }
16

```

El código es una clase de prueba llamada TestPedido que utiliza la clase RestPedido.

```

...ave TestConversorMetros.java X RestPedido.java X TestPedido.java X ArtículoCientifico.java X TestArtic
Source History
1 package TDA;
2
3 import EntradaSalida.Tools;
4
5 public class ArtículoCientifico {
6     private String titulo;
7     private String autor;
8     private String[] palabrasClaves;
9     private String publicacion;
10    private int año;
11    private String resumen;
12
13    public ArtículoCientifico(String titulo, String autor) {
14        this.titulo = titulo;
15        this.autor = autor;
16    }
17
18    public ArtículoCientifico(String titulo, String autor,
19        String[] palabrasClaves, String publicacion, int año) {
20        this(titulo, autor);
21        this.palabrasClaves = palabrasClaves;
22        this.publicacion = publicacion;
23        this.año = año;
24    }
25
26    public ArtículoCientifico(String titulo, String autor,
27        String[] palabrasClaves, String publicacion, int año, String resumen) {
28        this(titulo, autor, palabrasClaves, publicacion, año);
29        this.resumen = resumen;
30    }
31
32    public String getTitulo() {
33        return titulo;
34    }
35
36    public void setTitulo(String titulo) {
37        this.titulo = titulo;
38    }
39
40    public String getAutor() {
41        return autor;
42    }
43
44    public void setAutor(String autor) {
45        this.autor = autor;
46    }
47
48    public String[] getPalabrasClaves() {
49        return palabrasClaves;
50    }
51
52    public void setPalabrasClaves(String[] palabrasClaves) {
53        this.palabrasClaves = palabrasClaves;
54    }
55
56    public String getPublicacion() {
57        return publicacion;
58    }
59
60    public void setPublicacion(String publicacion) {
61        this.publicacion = publicacion;
62    }
63
64    public int getAño() {
65        return año;
66    }
67
68    public void setAño(int año) {
69        this.año = año;
70    }
71
72    public String getResumen() {
73        return resumen;
74    }

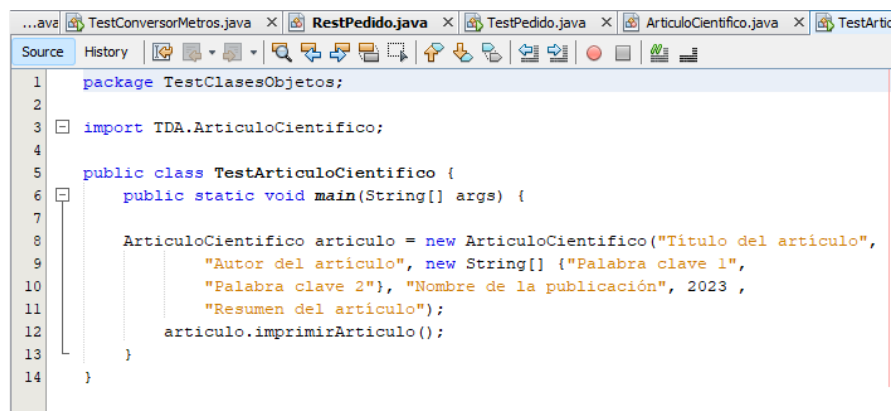
```

```

75
76     public void setResumen(String resumen) {
77         this.resumen = resumen;
78     }
79
80     public void imprimirArticulo() {
81         String mensaje = "Título: " + titulo + "\n" + "Autor: " + autor + "\n";
82         if (palabrasClaves != null) {
83             mensaje += "Palabras clave:\n";
84             for (String palabra : palabrasClaves) {
85                 mensaje += "- " + palabra + "\n";
86             }
87         }
88         mensaje += "Publicación: " + publicacion + "\n" + "Año: " + año + "\n";
89         if (resumen != null) {
90             mensaje += "Resumen: " + resumen + "\n";
91         }
92         Tools.imprimePantalla(mensaje);
93     }
94 }

```

El código define una clase llamada `ArticuloCientifico` que representa un artículo científico.



```

...ava TestConversorMetros.java x RestPedido.java x TestPedido.java x ArticuloCientifico.java x TestArtic
Source History
1 package TestClasesObjetos;
2
3 import TDA.ArticuloCientifico;
4
5 public class TestArticuloCientifico {
6     public static void main(String[] args) {
7
8         ArticuloCientifico articulo = new ArticuloCientifico("Título del artículo",
9             "Autor del artículo", new String[] {"Palabra clave 1",
10             "Palabra clave 2"}, "Nombre de la publicación", 2023 ,
11             "Resumen del artículo");
12         articulo.imprimirArticulo();
13     }
14 }

```

El código define una clase de prueba llamada `TestArticuloCientifico` que crea una instancia de la clase `ArticuloCientifico` con valores de ejemplo para los atributos del artículo, como el título, autor, palabras clave, publicación, año y resumen.


```
...ave TestConversorMetros.java x RestPedido.java x TestPedido.java x ArtículoCientífico.java x TestArticu
Source History
1 package TDA;
2
3 import EntradaSalida.Tools;
4
5 public class AsistenteBoda {
6     private String nombre;
7     private byte edad;
8     private String sexo;
9     private String estadoCivil;
10
11     private static byte total=0;
12     private static byte totalHombres=0;
13     private static byte totalMujeres=0;
14     private static byte totalMayores=0;
15     private static byte totalMenores=0;
16     private static byte totalSolteros=0;
17     private static byte totalCasados=0;
18     private static byte totalViudos=0;
19     private static byte totalDivorciados=0;
20     private static byte totalSeparados=0;
21
22     public AsistenteBoda(String nombre, byte edad, String sexo,
23         String estadoCivil) {
24         this.nombre = nombre;
25         this.edad = edad;
26         this.sexo = sexo;
27         this.estadoCivil = estadoCivil;
28         this.totalHombres = 0;
29         this.totalMujeres = 0;
30         this.totalCasados = 0;
31         this.totalSolteros = 0;
32         this.totalViudos = 0;
33         this.totalSeparados = 0;
34         this.totalDivorciados = 0;
35     }
36
37     public String getNombre() {
38         return nombre;
39     }
40
41     public void setNombre(String nombre) {
42         this.nombre = nombre;
43     }
44
45     public byte getEdad() {
46         return edad;
47     }
48
49     public void setEdad(byte edad) {
50         this.edad = edad;
51     }
52
53     public String getSexo() {
54         return sexo;
55     }
56
57     public void setSexo(String sexo) {
58         this.sexo = sexo;
59     }
60
61     public String getEstadoCivil() {
62         return estadoCivil;
63     }
64
65     public void setEstadoCivil(String estadoCivil) {
66         this.estadoCivil = estadoCivil;
67     }
68
69     public byte getTotalHombres() {
70         return totalHombres;
71     }
```

```

72
73 public void setTotalHombres(byte totalHombres) {
74     this.totalHombres = totalHombres;
75 }
76
77 public byte getTotalMujeres() {
78     return totalMujeres;
79 }
80
81 public void setTotalMujeres(byte totalMujeres) {
82     this.totalMujeres = totalMujeres;
83 }
84
85 public byte getTotalCasados() {
86     return totalCasados;
87 }
88
89 public void setTotalCasados(byte totalCasados) {
90     this.totalCasados = totalCasados;
91 }
92
93 public byte getTotalSolteros() {
94     return totalSolteros;
95 }
96
97 public void setTotalSolteros(byte totalSolteros) {
98     this.totalSolteros = totalSolteros;
99 }
100
101 public byte getTotalViudos() {
102     return totalViudos;
103 }
104
105 public void setTotalViudos(byte totalViudos) {
106     this.totalViudos = totalViudos;
107 }
108
109 public byte getTotalSeparados() {
110     return totalSeparados;
111 }
112
113 public void setTotalSeparados(byte totalSeparados) {
114     this.totalSeparados = totalSeparados;
115 }
116
117 public byte getTotalDivorciados() {
118     return totalDivorciados;
119 }
120
121 public void setTotalDivorciados(byte totalDivorciados) {
122     this.totalDivorciados = totalDivorciados;
123 }
124
125 public byte getTotalMayores() {
126     return totalMayores;
127 }
128
129 public void setTotalMayores(byte totalMayores) {
130     this.totalMayores = totalMayores;
131 }
132 public void setTotalMenores(byte totalMenores) {
133     this.totalMenores = totalMenores;
134 }
135
136 public static byte getTotal() {
137     return total;
138 }
139
140 public static void setTotal(byte total) {
141     AsistenteBoda.total = total;
142 }

```

```

143
144 public void registrarAsistente() {
145     total++;
146     if (sexo.equalsIgnoreCase("masculino")) {
147         totalHombres++;
148     } else {
149         totalMujeres++;
150     }
151
152     if (edad >= 18) {
153         totalMayores++;
154     } else {
155         totalMenores++;
156     }
157
158     if (estadoCivil.equalsIgnoreCase("soltero")) {
159         totalSolteros++;
160     } else if (estadoCivil.equalsIgnoreCase("casado")) {
161         totalCasados++;
162     } else if (estadoCivil.equalsIgnoreCase("viudo")) {
163         totalViudos++;
164     } else if (estadoCivil.equalsIgnoreCase("divorciado")) {
165         totalDivorciados++;
166     } else if (estadoCivil.equalsIgnoreCase("separado")) {
167         totalSeparados++;
168     }
169 }
170
171 public void mostrarEstadisticas() {
172     Tools.imprimePantalla("Total de asistentes: " + total+
173         "\nTotal de hombres: " + totalHombres+
174         "\nTotal de mujeres: " + totalMujeres+
175         "\nTotal de mayores de edad: " + totalMayores+
176         "\nTotal de menores de edad: " + totalMenores+
177         "\nTotal de solteros: " + totalSolteros+
178         "\nTotal de casados: " + totalCasados+
179         "\nTotal de viudos: " + totalViudos+
180         "\nTotal de divorciados: " + totalDivorciados+
181         "\nTotal de separados: " + totalSeparados);
182 }
183 }

```

El código implementa una clase llamada "AsistenteBoda" que lleva el registro de los asistentes a una boda, almacenando información como nombre, edad, sexo y estado civil.

```

...ave TestConversorMetros.java x RestPedido.java x TestPedido.java x ArtículoCientífico.java x TestArtículoCientífico.java x
Source History
1 package TestClasesObjetos;
2
3 import EntradaSalida.Tools;
4 import TDA.AsistenteBoda;
5
6 public class TestAsistenteBoda {
7     public static void main(String[] args) {
8         AsistenteBoda asistente1 = new AsistenteBoda("Juan", (byte) 25, "Masculino", "Soltero");
9         AsistenteBoda asistente2 = new AsistenteBoda("Maria", (byte) 30, "Femenino", "Casado");
10        AsistenteBoda asistente3 = new AsistenteBoda("Pedro", (byte) 17, "Masculino", "Soltero");
11
12        asistente1.registrarAsistente();
13        asistente2.registrarAsistente();
14        asistente3.registrarAsistente();
15
16        asistente1.mostrarEstadisticas();
17        asistente2.mostrarEstadisticas();
18        asistente3.mostrarEstadisticas();
19    }
20 }
21
22

```

El código es una prueba de una clase llamada "AsistenteBoda".

CONCLUSIONES

En conclusión, los métodos son una parte fundamental de la programación, permitiendo encapsular la lógica y funcionalidad en bloques de código reutilizables. Durante la realización de la práctica, se pudo observar que la definición y estructura de un método son elementos esenciales para su correcto funcionamiento. Además, se identificó la importancia del valor de retorno en los métodos, ya que permite obtener resultados o información específica de la ejecución del método.

En cuanto a la declaración de los métodos, se observó que existen dos tipos principales: los métodos de clase y los métodos de instancia, cada uno con sus características y formas de utilización. Además, se identificó la importancia del ámbito y tiempo de vida de las variables en un método, ya que afecta a su accesibilidad y utilización dentro del mismo.

Se analizó también la importancia de los argumentos y el paso de parámetros en los métodos, permitiendo la comunicación y transferencia de datos entre diferentes métodos o partes de un programa. Se mencionó la relevancia del puntero "this" en los métodos, que hace referencia al objeto actual y permite acceder a sus propiedades y métodos.

Se destacó la sobrecarga de métodos como una técnica que permite definir varios métodos con el mismo nombre pero con diferentes parámetros, lo cual aumenta la flexibilidad y versatilidad de un programa. Por último, se revisó la importancia de los constructores y destructores en la creación y destrucción de objetos en la programación orientada a objetos.

BIBLIOGRAFÍA

Stroustrup, B. (2013). C++ Programming Language: The Definitive Reference (C++ Reference Series). Addison-Wesley Professional.

Schildt, H. (2017). Java: The Complete Reference. McGraw-Hill Education.

Matthes, E. (2019). Python Crash Course: A Hands-On, Project-Based Introduction to Programming. No Starch Press.

Wiener, R., & Pinson, L. J. (2017). Fundamentals of OOP and Data Structures in Java. Springer.

Mueller, J. P. (2017). C# 7.0 All-in-One For Dummies. For Dummies.