

**Instituto Tecnológico de
Orizaba**



**TECNOLÓGICO
NACIONAL DE MÉXICO**

CARRERA

ING. INFORMATICA

ASIGNATURA

PROGRAMACION ORIENTADA A OBJETOS

TEMA 2

CLASES Y OBJETOS

ALUMN@

MAYTE MELLADO HUERTA

NO.CONTROL

21010202

GRUPO

2a3B

FECHA DE ENTREGA

17/04/2023

INTRODUCCION

El tema del reporte es "Clases y Objetos" y se describen varios subtemas relacionados con este tema. Estos subtemas incluyen:

2.1 Definición de una clase: Se aborda la conceptualización y definición de una clase en la programación orientada a objetos, que es una plantilla o molde para crear objetos.

2.2 Declaración de clases: Se presenta cómo se declara una clase en un lenguaje de programación específico, incluyendo la sintaxis y las convenciones de nomenclatura.

2.3 Miembros de una clase: Se describe qué son los miembros de una clase, como atributos y métodos, y cómo se definen y utilizan en la programación orientada a objetos.

2.4 Ámbito referente a una clase: Se explica el concepto de ámbito o alcance de los miembros de una clase, es decir, dónde pueden ser accesados y utilizados.

2.5 Especificadores de acceso: Se detallan los diferentes especificadores de acceso, como público, privado y protegido, y cómo afectan la visibilidad y manipulación de los miembros de una clase.

2.6 Creación de objetos: Se explica cómo se crean instancias u objetos de una clase, que son las representaciones concretas de la clase en tiempo de ejecución.

2.7 Clases predefinidas: Se mencionan las clases predefinidas o librerías estándar que vienen con un lenguaje de programación y que pueden ser utilizadas para realizar operaciones comunes.

2.8 Definición, creación y reutilización de paquetes/librerías: Se aborda cómo se definen, crean y reutilizan paquetes o librerías de clases en la programación, para organizar y compartir código de manera eficiente.

2.9 Manejo de excepciones: Se describe cómo se manejan las excepciones o errores en la programación orientada a objetos, utilizando mecanismos de manejo de excepciones para gestionar situaciones excepcionales que puedan ocurrir durante la ejecución de un programa.

En resumen, el reporte trata sobre los aspectos fundamentales de las clases y objetos en la programación orientada a objetos, incluyendo su definición, declaración, miembros, ámbito, especificadores de acceso, creación de objetos, clases predefinidas, definición y creación de paquetes/librerías, y manejo de excepciones.

COMPETENCIA ESPECIFICA

Comprende y aplica la estructura de clases para la creación de objetos y utiliza clases predefinidas para facilitar el desarrollo de aplicaciones:

1. Saber diferenciar entre un objeto y una clase, y comprender la relación que existe entre ellos.
2. Conocer y entender las partes que componen una clase, es decir, los atributos y los métodos.
3. Tener un primer contacto con el lenguaje de modelado UML.
4. Saber los conceptos relacionados con un objeto, como son: instancia, estado, comportamiento y mensaje.
5. Reflexionar sobre cómo el paradigma.

MARCO TEÓRICO:

2.1 Definición de una clase:

En la programación orientada a objetos, una clase es una plantilla o molde que define un conjunto de características o atributos (datos) y comportamientos o métodos (funciones) que pueden tener los objetos creados a partir de ella. Una clase se define con un nombre, y puede tener propiedades y métodos asociados.

2.2 Declaración de clases:

La declaración de una clase varía según el lenguaje de programación utilizado, pero generalmente implica la definición del nombre de la clase, la especificación de sus atributos y métodos, y la definición de su estructura. Por ejemplo, en lenguajes como Java o C++, se utiliza una sintaxis específica para declarar una clase, que puede incluir palabras clave como "class", "public", "private" y "protected", entre otros.

2.3 Miembros de una clase:

Una clase puede tener varios miembros, que son las características y comportamientos que la clase define. Los miembros de una clase pueden incluir atributos, que son variables que almacenan datos relacionados con la clase, y métodos, que son funciones que definen las acciones o comportamientos que los objetos de la clase pueden realizar.

2.4 Ámbito referente a una clase:

El ámbito o alcance de los miembros de una clase se refiere a la visibilidad y accesibilidad de estos miembros dentro y fuera de la clase. Los miembros pueden tener distintos niveles de acceso, como público, privado y protegido, que determinan desde dónde se pueden acceder y manipular. Por ejemplo, los miembros públicos son accesibles desde cualquier parte del código, mientras que los miembros privados solo son accesibles dentro de la propia clase.

2.5 Especificadores de acceso:

Los especificadores de acceso son palabras clave utilizadas en la declaración de una clase para establecer el nivel de visibilidad de los miembros de la clase. Por lo general, los especificadores de acceso más comunes son "public", "private" y "protected". "Public" permite que los miembros sean accesibles desde cualquier

parte del código, "private" limita el acceso a los miembros solo dentro de la propia clase, y "protected" permite el acceso desde la clase y sus clases derivadas.

2.6 Creación de objetos:

La creación de objetos se refiere a la instancia o creación concreta de un objeto a partir de una clase. Un objeto es una representación concreta de una clase en tiempo de ejecución, con sus propios valores de atributos y la capacidad de ejecutar los métodos definidos en la clase. La creación de objetos se realiza mediante la instanciación de una clase, utilizando la sintaxis específica del lenguaje de programación.

2.7 Clases predefinidas:

Las clases predefinidas, también conocidas como librerías estándar o clases de la biblioteca, son clases que vienen incluidas en un lenguaje de programación y proporcionan funcionalidades comunes que pueden ser utilizadas sin necesidad de definirlas desde cero. Estas clases predefinidas suelen proporcionar funcionalidades como manipulación de cadenas, operaciones matemáticas, entrada/salida de datos, entre otras.

2.8 Definición, creación y reutilización de paquetes/librerías:

Los paquetes o librerías son conjuntos de clases y otros recursos que se agrupan juntos para facilitar su reutilización en diferentes proyectos. La definición y creación de paquetes/librerías implica la organización de clases y otros recursos en una estructura jerárquica, la definición de interfaces o APIs (Application Programming Interfaces) para su uso, y la creación de documentación y ejemplos de uso para facilitar su reutilización en diferentes proyectos.

2.9 Manejo de excepciones:

El manejo de excepciones se refiere a la forma en que se gestionan y controlan los errores o situaciones excepcionales que puedan ocurrir durante la ejecución de un programa. Las excepciones son eventos inesperados que pueden interrumpir el flujo normal del programa y causar errores. El manejo de excepciones implica la captura,

procesamiento y respuesta adecuada a estas situaciones, utilizando técnicas como la declaración de excepciones, el uso de bloques try-catch, la propagación de excepciones y la definición de excepciones personalizadas.

MATERIAL Y EQUIPO:

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

DESARROLLO DE LA PRACTICA

Paso 1: Definición de una clase

En esta etapa se debe comprender la definición de una clase, que es una plantilla o molde que describe las características y comportamientos de un objeto. Se pueden definir atributos (variables) y métodos (funciones) en una clase para representar su estado y comportamiento.

Paso 2: Declaración de clases

En esta etapa se debe aprender cómo declarar una clase en el lenguaje de programación específico que se esté utilizando. Esto incluye la sintaxis y la estructura básica de una declaración de clase, así como las convenciones de nomenclatura y estilo de codificación.

Paso 3: Miembros de una clase

En esta etapa se debe entender los diferentes tipos de miembros que pueden tener una clase, como atributos y métodos. Los atributos representan las variables que almacenan el estado de un objeto, mientras que los métodos son las funciones que definen su comportamiento.

Paso 4: Ámbito referente a una clase

En esta etapa se debe conocer el concepto de ámbito en el contexto de las clases y objetos. Esto implica entender cómo se acceden y modifican los atributos y métodos de una clase desde diferentes partes del código, y cómo se puede controlar su visibilidad y alcance.

Paso 5: Especificadores de acceso

En esta etapa se debe comprender los especificadores de acceso, que son palabras clave utilizadas para controlar la visibilidad y el alcance de los miembros de una clase. Esto incluye los especificadores de acceso públicos, privados y protegidos, y cómo se utilizan para definir y acceder a los atributos y métodos de una clase.

Paso 6: Creación de objetos

En esta etapa se debe aprender cómo crear objetos a partir de una clase. Esto implica entender el proceso de instanciación, que consiste en crear una copia de una clase en memoria para utilizarla como un objeto independiente con su propio estado y comportamiento.

Paso 7: Clases predefinidas

En esta etapa se debe conocer las clases predefinidas o integradas en el lenguaje de programación que se esté utilizando. Estas son clases que ya están disponibles en el lenguaje y que proporcionan funcionalidades básicas, como la manipulación de cadenas, fechas, números, etc.

Paso 8: Definición, creación y reutilización de paquetes/librerías

En esta etapa se debe entender cómo se definen, crean y utilizan paquetes o librerías, que son conjuntos de clases y otros recursos que se agrupan juntos para facilitar su reutilización en diferentes proyectos. Esto incluye la organización jerárquica de clases en paquetes, la definición de interfaces o APIs, y la creación de documentación y ejemplos de uso.

Paso 9: Manejo de excepciones

En esta etapa se debe comprender cómo se manejan las excepciones, que son situaciones excepcionales o errores que pueden ocurrir durante la ejecución de un programa. Esto implica la captura, procesamiento y respuesta adecuada a estas situaciones utilizando bloques try-catch, declarando y lanzando excepciones, y creando excepciones personalizadas cuando sea necesario.

Advertencias:

- ♥ Es importante seguir las reglas de sintaxis y nomenclatura específicas del lenguaje de programación que se esté utilizando al definir clases, atributos, métodos y otros elementos del código.
- ♥ Se debe tener cuidado al manejar los especificadores de acceso, ya que su uso incorrecto puede causar errores en la ejecución del programa o comprometer la seguridad de los datos.
- ♥ Es fundamental entender el concepto de instanciación y crear objetos correctamente, asegurándose de que cada objeto tenga su propio estado independiente.
- ♥ Al utilizar clases predefinidas o librerías externas, se debe revisar y entender adecuadamente la documentación y los ejemplos de uso para asegurarse de utilizarlas correctamente en el contexto del proyecto.
- ♥ El manejo de excepciones debe realizarse de forma adecuada para garantizar una correcta gestión de errores y una ejecución estable del programa. Es importante identificar y manejar adecuadamente las excepciones que puedan ocurrir en diferentes partes del código.

Precauciones:

- ♥ Leer y comprender completamente los requisitos y restricciones de la práctica antes de comenzar a implementar el código.
- ♥ Seguir los pasos de manera ordenada y verificar que cada etapa se haya completado correctamente antes de avanzar a la siguiente.

- ♥ Hacer pruebas y depuración del código a medida que se va implementando, para identificar y corregir posibles errores.
- ♥ Mantener una buena documentación del código, incluyendo comentarios explicativos y diagramas de clase, para facilitar su comprensión y mantenimiento en el futuro.

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

Libros
- String Titulo - autor Autor - int isbn - int noPaginas
+ Libros() + Libros(String Titulo, Autor autor, int isbn, int noPaginas) + String getTitulo() + void setTitulo(String Titulo) + Autor getAutor() + void setAutor(Autor autor) + int getIsbn() + void setIsbn(int isbn) + int getNoPaginas() + void setNoPaginas(int noPaginas) + String toString()

Autor
- String nomAutor - String apellidoAutor
+ Autor()

```

+ Autor(String nomAutor, String apellidoAutor)
+ String getNomAutor()
+ void setNomAutor(String nomAutor)
+ String getApellidoAutor()
+ void setApellidoAutor(String apellidoAutor)
+ String toString()

```

Persona
<ul style="list-style-type: none"> - String nomPer - long telPers - byte edadPers - String nacioPer - Fecha fechper - char geneper
<pre> + Persona() + Persona(String nomPer,long telPers,byte edadPers,String nacioPer,Fecha fechper,char geneper) + String getNomPer() + void setNomPer(String nomPer) + long getTelPers() + void setTelPers(long telPers) + byte getEdadPers() + void setEdadPers(byte edadPers) + String getNacioPer() + void setNacioPer(String nacioPer) + Fecha getFechper() + void setFechper(Fecha fechper) + char getGeneper() + void setGeneper(char geneper) </pre>

+ String toString()

Fecha

- byte dia - byte mes - short año

+ Fecha() + Fecha (byte dia, byte mes, short año) + byte getDia() + void setDia(byte dia) + byte getMes() + void setMes(byte mes) + short getAño() + void setAño(short año) + String toString()

RentaAutos

- String TipoAuto - byte DiasRenta - double kmRecorridos
--

+ RentaAutos() + RentaAutos(String TipoAuto,byte DiasRenta,double kmRecorridos) + int getDia() + void setDiasRenta(byte DiasRenta) + double getkmRecorridos() + void setKmRecorridos(double kmRecorridos) + String getTipoAuto() + void setTipoAuto(String TipoAuto)
--

TarifaCosto
+ static double tarifa(byte DiasRenta, String TipoAuto)
+ static double calcularCosto(double tarifa, double kmRecorridos, String TipoAuto)

DatosAlumno
- String nombre
- long nControl
- String semestre
- String Periodo
- String carrera
+ DatosAlumno(String nombre, long nControl, String semestre, String periodo, String carrera)
+ String getNombre()
+ void setNombre(String nombre)
+ long getnControl()
+ void setnControl(long nControl)
+ String getSemestre()
+ void setSemestre(String semestre)
+ String getPeriodo()
+ void setPeriodo(String periodo)
+ String getCarrera()
+ void setCarrera(String carrera)
+ String toString()

FechaActual
+ String obtenerFechaActual()

Triangulo
<ul style="list-style-type: none"> - float base - float altura
<ul style="list-style-type: none"> + Triangulo() + Triangulo(float base, float altura) + float getBase() + void setBase(float base) + float getAltura() + void setAltura(float altura) + String TipoTriangulo() + float Perimetro() + float Area()

Rectangulo
<ul style="list-style-type: none"> - double Base - double Altura - double Area - double Perimetro
<ul style="list-style-type: none"> + Rectangulo() + Rectangulo (double Base,double Altura) + double getArea() + double getPerimetro() + double getBase() + void setBase(double Base) + void setAltura(double Altura) + double CalArea() + double CalPerimetro()

Cuadrado

- double Area - double Perimetro - double Lado
--

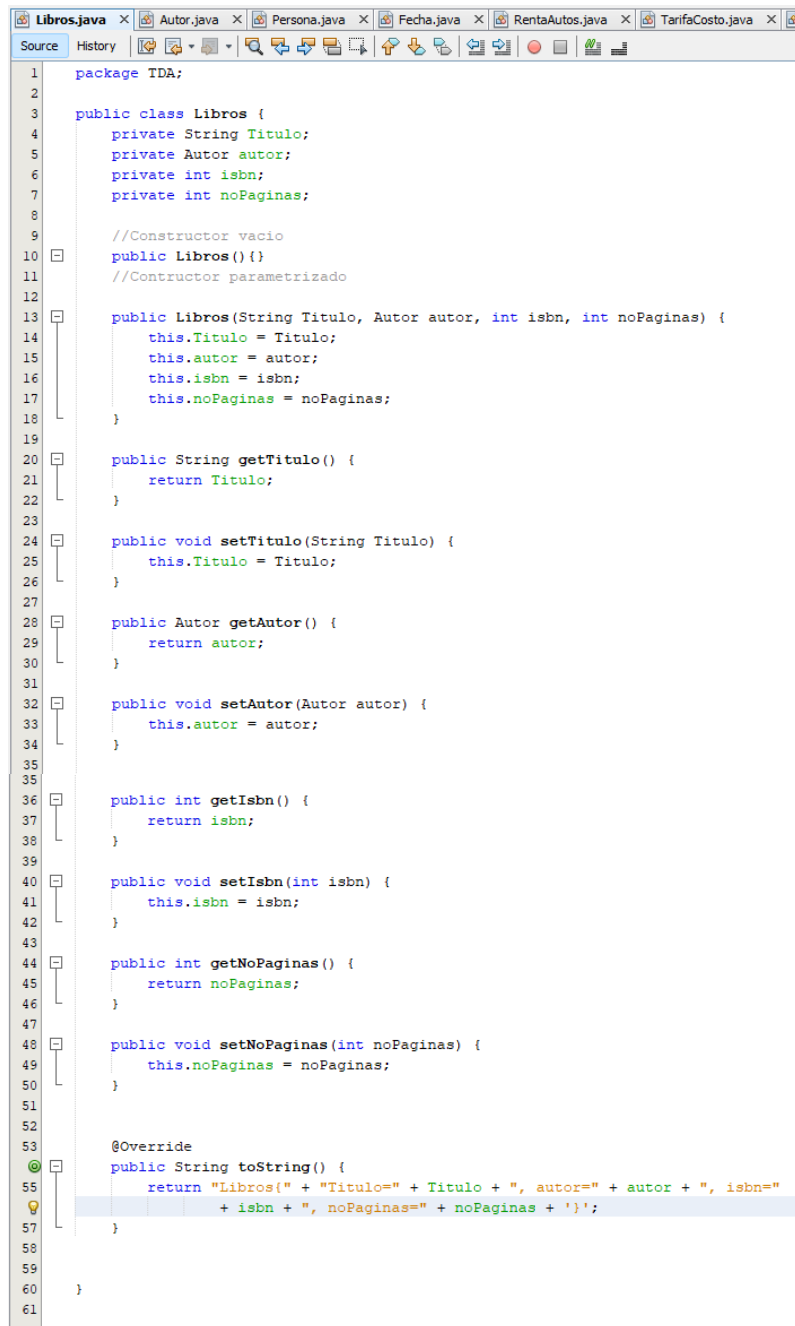
+ Cuadrado() + Cuadrado (double Lado) + double getArea() + double getPerimetro() + double getBase() + void setLado(double Lado) + double CalArea() + double CalPerimetro()

Circulo

- double Perimetro - double Area - double Radio

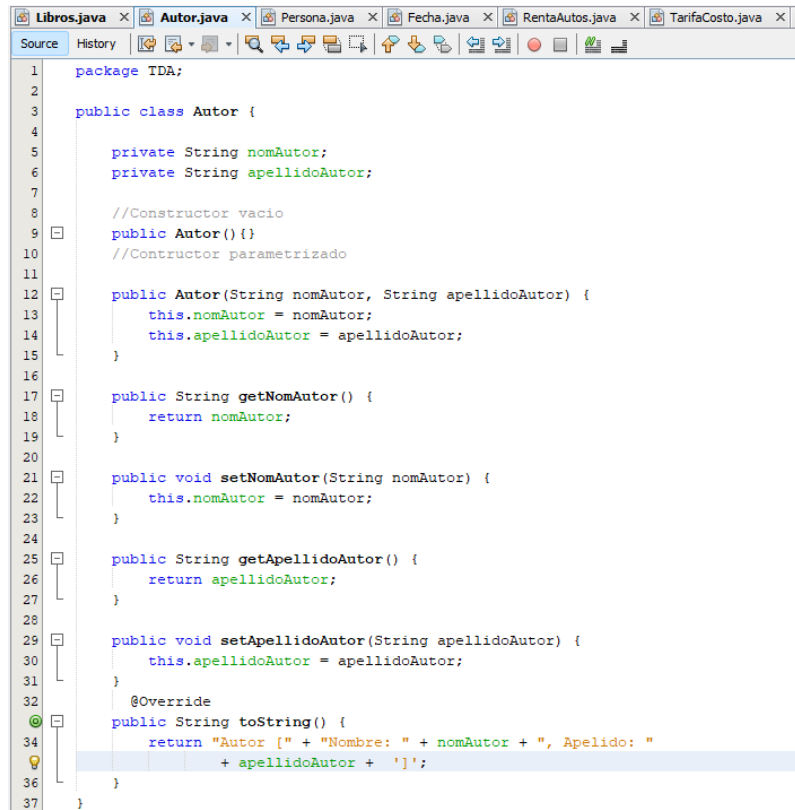
+ Circulo() + Circulo (double Radio) + double getArea() + double getPerimetro() + double getRadio() + void setRadio(double Radio) + double CalArea() + double CalPerimetro()

RESULTADOS



```
1 package TDA;
2
3 public class Libros {
4     private String Titulo;
5     private Autor autor;
6     private int isbn;
7     private int noPaginas;
8
9     //Constructor vacio
10    public Libros() {}
11    //Constructor parametrizado
12
13    public Libros(String Titulo, Autor autor, int isbn, int noPaginas) {
14        this.Titulo = Titulo;
15        this.autor = autor;
16        this.isbn = isbn;
17        this.noPaginas = noPaginas;
18    }
19
20    public String getTitulo() {
21        return Titulo;
22    }
23
24    public void setTitulo(String Titulo) {
25        this.Titulo = Titulo;
26    }
27
28    public Autor getAutor() {
29        return autor;
30    }
31
32    public void setAutor(Autor autor) {
33        this.autor = autor;
34    }
35
36    public int getIsbn() {
37        return isbn;
38    }
39
40    public void setIsbn(int isbn) {
41        this.isbn = isbn;
42    }
43
44    public int getNoPaginas() {
45        return noPaginas;
46    }
47
48    public void setNoPaginas(int noPaginas) {
49        this.noPaginas = noPaginas;
50    }
51
52    @Override
53    public String toString() {
54        return "Libros(" + "Titulo=" + Titulo + ", autor=" + autor + ", isbn="
55            + isbn + ", noPaginas=" + noPaginas + ')';
56    }
57
58 }
59
60
61
```

Este código proporciona una estructura básica para representar y manipular información de libros en un programa Java, con métodos para obtener y establecer los atributos del libro, así como para obtener una representación en forma de cadena del objeto.



```

1 package TDA;
2
3 public class Autor {
4
5     private String nomAutor;
6     private String apellidoAutor;
7
8     //Constructor vacio
9     public Autor() {}
10    //Constructor parametrizado
11
12    public Autor(String nomAutor, String apellidoAutor) {
13        this.nomAutor = nomAutor;
14        this.apellidoAutor = apellidoAutor;
15    }
16
17    public String getNomAutor() {
18        return nomAutor;
19    }
20
21    public void setNomAutor(String nomAutor) {
22        this.nomAutor = nomAutor;
23    }
24
25    public String getApellidoAutor() {
26        return apellidoAutor;
27    }
28
29    public void setApellidoAutor(String apellidoAutor) {
30        this.apellidoAutor = apellidoAutor;
31    }
32    @Override
33    public String toString() {
34        return "Autor [" + "Nombre: " + nomAutor + ", Apellido: "
35            + apellidoAutor + ']';
36    }
37 }

```

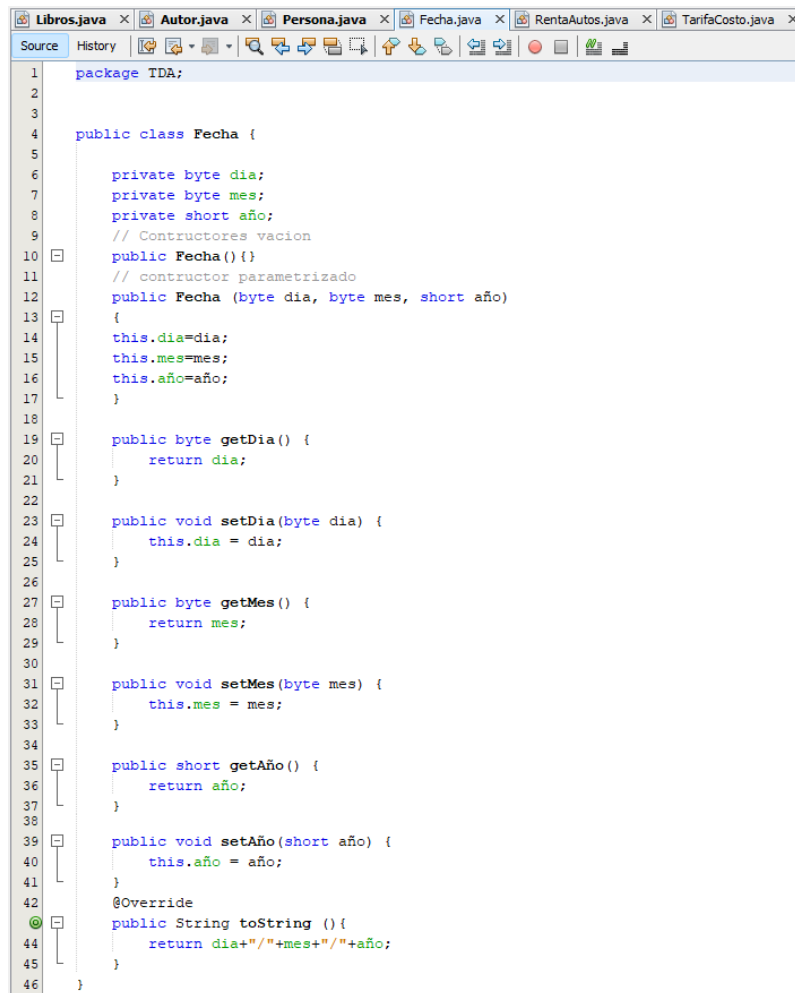
Este código proporciona una estructura básica para representar y manipular información de autores de libros en un programa Java, con métodos para obtener y establecer los atributos del autor, así como para obtener una representación en forma de cadena del objeto.


```

Libros.java x Autor.java x Persona.java x Fecha.java x RentaAutos.java x TarifaCosto.java x
Source History
1 package TDA;
2
3 public class Persona {
4     // Atributos
5     private String nomPer;
6     private long telPers;
7     private byte edadPers;
8     private String nacioPer;
9     private Fecha fechper;
10    private char geneper;
11    //Constructor vacio
12    public Persona() {}
13    //Constructor parametrizado
14
15    public Persona(String nomPer,long telPers,byte edadPers,String nacioPer,
16        Fecha fechper,char geneper){
17        this.nomPer=nomPer;
18        this.telPers=telPers;
19        this.edadPers=edadPers;
20        this.fechper=fechper;
21        this.geneper=geneper;
22        this.nacioPer=nacioPer;
23    }
24
25    public String getNomPer() {
26        return nomPer;
27    }
28
29    public void setNomPer(String nomPer) {
30        this.nomPer = nomPer;
31    }
32
33    public long getTelPers() {
34        return telPers;
35    }
36
37    public void setTelPers(long telPers) {
38        this.telPers = telPers;
39    }
40
41    public byte getEdadPers() {
42        return edadPers;
43    }
44
45    public void setEdadPers(byte edadPers) {
46        this.edadPers = edadPers;
47    }
48
49    public String getNacioPer() {
50        return nacioPer;
51    }
52
53    public void setNacioPer(String nacioPer) {
54        this.nacioPer = nacioPer;
55    }
56
57    public Fecha getFechper() {
58        return fechper;
59    }
60
61    public void setFechper(Fecha fechper) {
62        this.fechper = fechper;
63    }
64
65    public char getGeneper() {
66        return geneper;
67    }
68
69    public void setGeneper(char geneper) {
70        this.geneper = geneper;
71    }
72
73    @Override
74    public String toString() {
75        return "Persona [" + "Nombre: " + nomPer + ", Telefono: " + telPers +
76            ", Edad: " + edadPers + ", Nacionalidad: " + nacioPer +
77            ", Fecha de Cumpleaños: " + fechper + ", Genero: "
78            + geneper + ']';
79    }
80
81 }

```

Este código proporciona una estructura básica para representar y manipular información de personas en un programa Java, con métodos para obtener y establecer los atributos de la persona, así como para obtener una representación en forma de cadena del objeto.



```
1 package TDA;
2
3
4 public class Fecha {
5
6     private byte dia;
7     private byte mes;
8     private short año;
9     // Constructores vacion
10    public Fecha(){}
11    // constructor parametrizado
12    public Fecha (byte dia, byte mes, short año)
13    {
14        this.dia=dia;
15        this.mes=mes;
16        this.año=año;
17    }
18
19    public byte getDia() {
20        return dia;
21    }
22
23    public void setDia(byte dia) {
24        this.dia = dia;
25    }
26
27    public byte getMes() {
28        return mes;
29    }
30
31    public void setMes(byte mes) {
32        this.mes = mes;
33    }
34
35    public short getAño() {
36        return año;
37    }
38
39    public void setAño(short año) {
40        this.año = año;
41    }
42    @Override
43    public String toString () {
44        return dia+"/"+mes+"/"+año;
45    }
46 }
```

Este código proporciona una estructura básica para representar y manipular fechas en un programa Java, con métodos para obtener y establecer los atributos de la fecha, así como para obtener una representación en forma de cadena del objeto.



```

1 package TDA;
2
3 public class RentaAutos {
4
5     private String TipoAuto;
6     private byte DiasRenta;
7     private double kmRecorridos;
8
9     //Constructor vacio
10    public RentaAutos() {}
11
12    //Constructor parametrizado
13    public RentaAutos(String TipoAuto, byte DiasRenta, double kmRecorridos) {
14
15        this.TipoAuto = TipoAuto;
16        this.DiasRenta = DiasRenta;
17        this.kmRecorridos = kmRecorridos;
18    }
19
20    public int getDia() {
21        return DiasRenta;
22    }
23
24    public void setDiasRenta(byte DiasRenta) {
25        this.DiasRenta = DiasRenta;
26    }
27
28    public double getkmRecorridos() {
29        return kmRecorridos;
30    }
31
32    public void setKmRecorridos(double kmRecorridos) {
33        this.kmRecorridos = kmRecorridos;
34    }
35
36    public String getTipoAuto() {
37        return TipoAuto;
38    }
39
40    public void setTipoAuto (String TipoAuto) {
41        this.TipoAuto = TipoAuto;
42    }
43
44 }

```

Este código proporciona una estructura básica para representar y manipular la renta de autos en un programa Java, con métodos para obtener y establecer los atributos de la renta de autos.

```

Libros.java x Autor.java x Persona.java x Fecha.java x RentaAutos.java x TarifaCosto.java
Source History
1 package TDA;
2 public class TarifaCosto {
3     public static double tarifa(byte DiasRenta, String TipoAuto){
4         int tarifa=0;
5         if(TipoAuto.equalsIgnoreCase("chico"))
6             tarifa=200;
7         else
8             if(TipoAuto.equalsIgnoreCase("mediano"))
9                 tarifa = 350;
10            else
11                if(TipoAuto.equalsIgnoreCase("grande"))
12                    tarifa = 450;
13            tarifa*=DiasRenta;
14
15            return tarifa;
16        }
17    public static double calcularCosto(double tarifa, double kmRecorridos,
18        String TipoAuto){
19        double extra = 0;
20        double kms=0;
21        if(kmRecorridos>10)
22            extra = 2.5;
23
24        if(TipoAuto.equalsIgnoreCase("chico"))
25            kms = kmRecorridos*20;
26        else
27            if(TipoAuto.equalsIgnoreCase("mediano"))
28                kms = kmRecorridos*30;
29            else
30                if(TipoAuto.equalsIgnoreCase("grande"))
31                    kms = kmRecorridos*40;
32
33        double Costo = tarifa + (tarifa*extra) + kms;
34
35        return Costo;
36    }
37 }
38

```

Esta clase proporciona métodos para calcular la tarifa y el costo de una renta de autos, basados en los días de renta, tipo de auto y kilómetros recorridos.

```

Libros.java x Autor.java x Persona.java x Fecha.java x RentaAutos.java x TarifaCosto.java
Source History
1 package TDA;
2
3 import TDA.FechaActual;
4
5 public class DatosAlumno {
6
7     //Parametros
8     private String nombre;
9     private long nControl;
10    private String semestre;
11    private String Periodo;
12    private String carrera;
13
14    public DatosAlumno(String nombre, long nControl, String semestre,
15        String periodo, String carrera) {
16        this.nombre = nombre;
17        this.nControl = nControl;
18        this.semestre = semestre;
19        this.Periodo = periodo;
20        this.carrera = carrera;
21    }
22
23    public String getNombre() {
24        return nombre;
25    }
26
27    public void setNombre(String nombre) {
28        this.nombre = nombre;
29    }
30
31    public long getnControl() {
32        return nControl;
33    }
34
35    public void setnControl(long nControl) {
36        this.nControl = nControl;
37    }
38
39    public String getSemestre() {
40        return semestre;
41    }
42
43    public void setSemestre(String semestre) {
44        this.semestre = semestre;
45    }
46
47    public String getPeriodo() {
48        return Periodo;
49    }
50
51    public void setPeriodo(String periodo) {
52        this.Periodo = periodo;
53    }
54
55    public String getCarrera() {
56        return carrera;
57    }
58
59    public void setCarrera(String carrera) {
60        this.carrera = carrera;
61    }
62
63    @Override
64    public String toString() {
65        return "Orizaba Ver. " + FechaActual.obtenerFechaActual() +
66            " Semestre: " + semestre + "\n" + " Nombre: " + nombre +
67            " Numero de control: " + nControl + "\n"
68            + " Carrera: " + carrera + " Periodo: " + Periodo;
69    }
70 }

```

Esta clase que representa los datos de un alumno, incluyendo su nombre, número de control, semestre, periodo y carrera.

```
Libros.java x Autor.java x Persona.java x Fecha.java x RentaAutos.java x TarifaCosto.java
Source History
1 package TDA;
2
3 import java.util.Date;
4 import java.text.SimpleDateFormat;
5 public class FechaActual {
6     public static String obtenerFechaActual() {
7         Date fecha = new Date();
8         SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
9         return formatoFecha.format(fecha);
10    }
11 }
```

Esta clase contiene un método estático que devuelve la fecha actual en formato de texto.

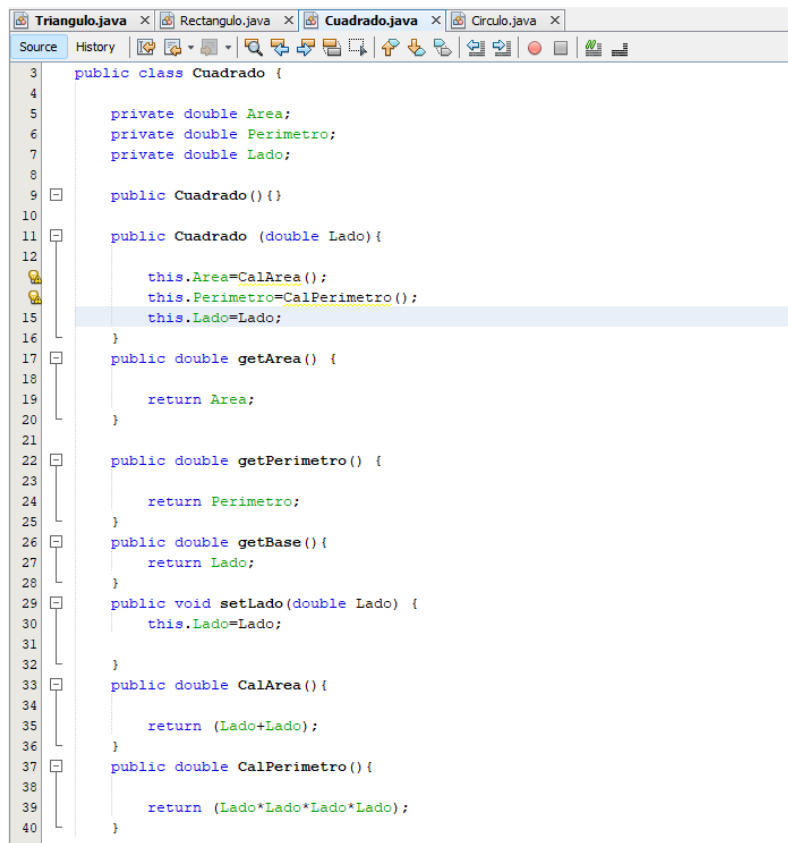
```
Triangulo.java x Rectangulo.java x Cuadrado.java x Circulo.java x
Source History
1 package TDA;
2
3 public class Triangulo {
4     private float base;
5     private float altura;
6
7     public Triangulo() {
8     }
9
10    public Triangulo(float base, float altura) {
11        this.base = base;
12        this.altura = altura;
13    }
14
15    public float getBase() {
16        return base;
17    }
18
19    public void setBase(float base) {
20        this.base = base;
21    }
22
23    public float getAltura() {
24        return altura;
25    }
26
27    public void setAltura(float altura) {
28        this.altura = altura;
29    }
30
31    public String TipoTriangulo() {
32        String tipo = "";
33        if (base == altura) {
34            tipo = "Equilátero";
35        } else if (base == altura || altura == (base * Math.sqrt(3) / 2)) {
36            tipo = "Isósceles";
37        } else {
38            tipo = "Escaleno";
39        }
40        return tipo;
41    }
42
43    public float Perimetro() {
44        float P = 0;
45        P = base + altura + (float) Math.sqrt(Math.pow(base, 2) +
46            Math.pow(altura, 2));
47        return P;
48    }
49
50    public float Area() {
51        float A = 0;
52        A = (base * altura) / 2;
53        return A;
54    }
55
56 }
```

Esta clase que representa un triángulo con propiedades como base, altura, métodos para calcular su tipo, perímetro y área.



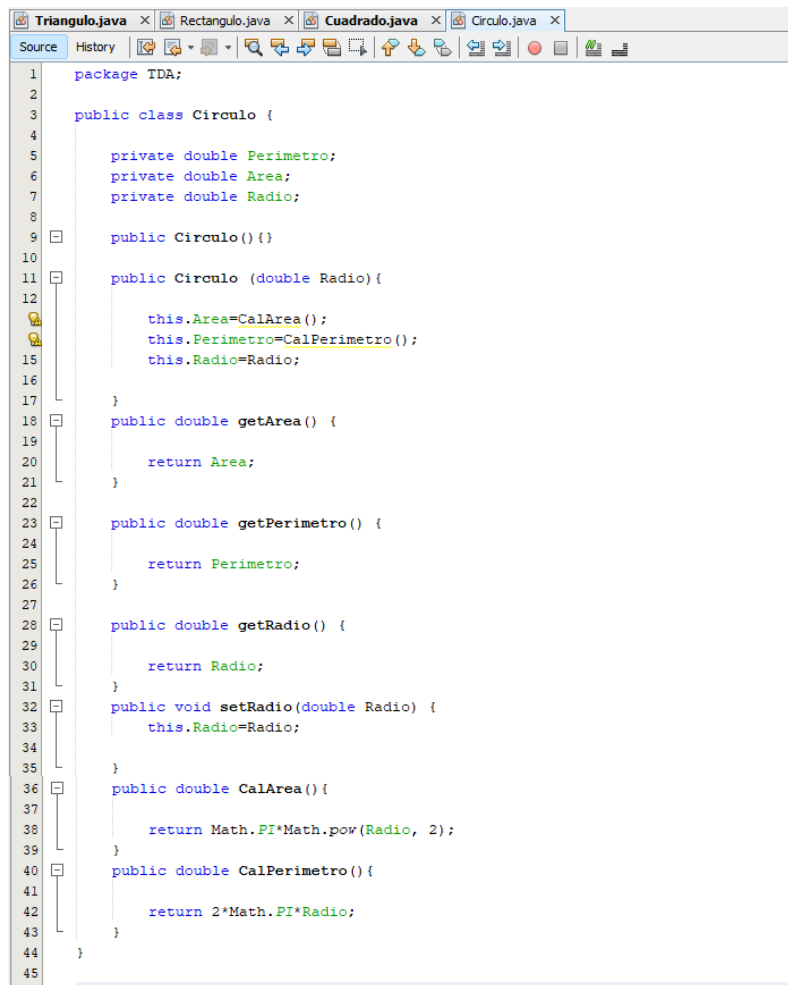
```
1 package TDA;
2
3 public class Rectangulo {
4     private double Base;
5     private double Altura;
6     private double Area;
7     private double Perimetro;
8
9     public Rectangulo() {}
10
11     public Rectangulo (double Base, double Altura) {
12
13         this.Area = CalArea();
14         this.Perimetro = CalPerimetro();
15         this.Base = Base;
16         this.Altura = Altura;
17     }
18
19     public double getArea() {
20
21         return Area;
22     }
23
24     public double getPerimetro() {
25
26         return Perimetro;
27     }
28     public double getBase() {
29         return Base;
30     }
31     public void setBase(double Base) {
32         this.Base = Base;
33     }
34
35     public double getAltura() {
36         return Base;
37     }
38     public void setAltura(double Altura) {
39         this.Altura = Altura;
40     }
41
42     public double CalArea() {
43
44         return (Altura * Base);
45     }
46     public double CalPerimetro() {
47
48         return (2 * Altura + Base);
49     }
50 }
51
```

Esta clase que representa un rectángulo con propiedades como base, altura, área y perímetro, y métodos para calcular el área y perímetro del rectángulo.



```
3 public class Cuadrado {
4
5     private double Area;
6     private double Perimetro;
7     private double Lado;
8
9     public Cuadrado() {}
10
11     public Cuadrado (double Lado) {
12         this.Area=CalArea();
13         this.Perimetro=CalPerimetro();
14         this.Lado=Lado;
15     }
16
17     public double getArea() {
18         return Area;
19     }
20
21
22     public double getPerimetro() {
23         return Perimetro;
24     }
25
26     public double getBase() {
27         return Lado;
28     }
29
30     public void setLado(double Lado) {
31         this.Lado=Lado;
32     }
33
34     public double CalArea() {
35         return (Lado*Lado);
36     }
37
38     public double CalPerimetro() {
39         return (Lado*Lado*Lado);
40     }
41 }
```

Esta clase que representa un cuadrado con propiedades como el área, perímetro y lado, y métodos para calcular el área y perímetro del cuadrado.



```
1 package TDA;
2
3 public class Circulo {
4
5     private double Perimetro;
6     private double Area;
7     private double Radio;
8
9     public Circulo() {}
10
11     public Circulo (double Radio) {
12
13         this.Area=CalArea();
14         this.Perimetro=CalPerimetro();
15         this.Radio=Radio;
16     }
17
18     public double getArea() {
19
20         return Area;
21     }
22
23     public double getPerimetro() {
24
25         return Perimetro;
26     }
27
28     public double getRadio() {
29
30         return Radio;
31     }
32     public void setRadio(double Radio) {
33         this.Radio=Radio;
34     }
35
36     public double CalArea() {
37
38         return Math.PI*Math.pow(Radio, 2);
39     }
40     public double CalPerimetro() {
41
42         return 2*Math.PI*Radio;
43     }
44 }
45
```

Esta clase que representa un círculo con propiedades como el área, perímetro y radio, y métodos para calcular el área y perímetro del círculo.

CONCLUSIONES

En este reporte, se han abordado diversos subtemas relacionados con las clases y objetos en la programación orientada a objetos. Se ha definido una clase como un modelo o plantilla que define la estructura y comportamiento de un objeto, y se ha discutido la declaración de clases, los miembros de una clase, el ámbito referente a una clase, los especificadores de acceso y la creación de objetos.

Además, se ha abordado el tema de las clases predefinidas, que son clases incluidas en los lenguajes de programación que ofrecen funcionalidades comunes y facilitan el desarrollo de programas. También se ha mencionado la importancia de la definición, creación y reutilización de paquetes/librerías, que permiten organizar y modularizar el código de manera eficiente.

Por último, se ha destacado la importancia del manejo de excepciones en la programación orientada a objetos, como una práctica necesaria para gestionar y manejar situaciones excepcionales o errores que puedan ocurrir durante la ejecución de un programa.

En resumen, el estudio de las clases y objetos en la programación orientada a objetos es fundamental para comprender cómo se estructura y organiza el código en este paradigma de programación. La comprensión de los conceptos abordados en este reporte, como la definición de clases, la creación de objetos, el uso de clases predefinidas, la gestión de paquetes/librerías y el manejo de excepciones, permitirá a los programadores desarrollar programas eficientes y robustos en el ámbito de la programación orientada a objetos.

BIBLIOGRAFÍA

Gaddis, T. (2019). Starting Out with Java: From Control Structures through Data Structures. Pearson.

Liang, Y. (2018). Introduction to Java Programming, Comprehensive Version. Pearson.

Deitel, P. & Deitel, H. (2017). Java: How to Program, Early Objects. Pearson.

Schildt, H. (2018). Java: A Beginner's Guide, Eighth Edition. McGraw-Hill Education.

Li, W. (2019). Java: A Detailed Approach to Practical Coding. Packt Publishing.

Savitch, W. (2018). Java: An Introduction to Problem Solving and Programming, 8th Edition. Pearson.

Horstmann, C. & Cornell, G. (2017). Core Java, Volume I--Fundamentals, 11th Edition. Prentice Hall.

Eckel, B. (2017). Thinking in Java, 4th Edition. Prentice Hall.

Gosling, J., Joy, B., & Steele, G. (2014). The Java Language Specification, Java SE 8 Edition. Addison-Wesley Professional.