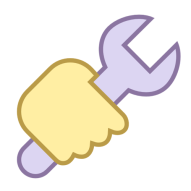
**CFP 8**

**Módulo 2:**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Unidad 4:**

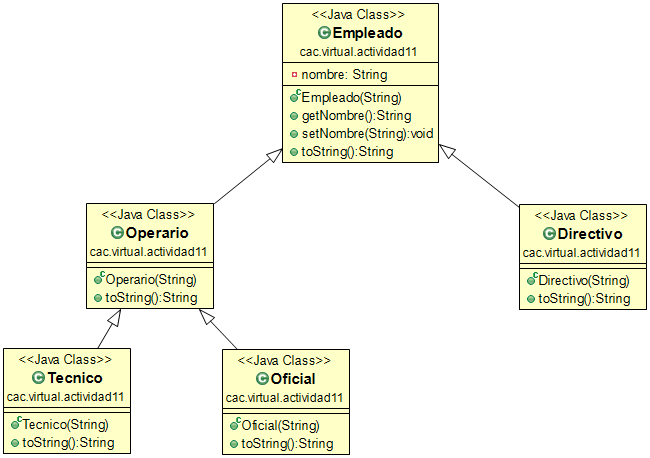
**HERENCIA**

****

## ACTIVIDADES PRÁCTICAS

### Herencia, Subclases y superclases, Sobrecarga, protected vs private, final, Herencia vs Composición, constructores en herencia

Considere la siguiente jerarquía de clases java representada por este diagrama UML para los **ejercicios 1 a 5**:



**1)**

Crear el código para la clase base Empleado. Esta clase contiene:

* Un atributo privado nombre de tipo String que **heredan** el resto de clases.
* Un constructor por defecto.
* Un constructor con parámetros que inicializa el nombre con el String que recibe.
* Método set y get para el atributo nombre.
* Un método toString() que devuelve el String: "Empleado " + nombre.

**Nota:** antes de escribir el resto de clases, que se mencionan en los ejercicios siguientes, considere que debe sobrescribir el método toString() en cada una de ellas y declarar el constructor adecuado de forma tal que al ejecutar las siguientes instrucciones:

* Empleado E1 = new Empleado("Rafael");
* Directivo D1 = new Directivo("Mario");
* Operario OP1 = new Operario("Alfonso");
* Oficial OF1 = new Oficial("Luis");
* Tecnico T1 = new Tecnico("Pablo");
* System.out.println(E1);
* System.out.println(D1);
* System.out.println(OP1);
* System.out.println(OF1);
* System.out.println(T1);

De como resultado lo siguiente:

* Empleado Rafael
* Empleado Mario -> Directivo
* Empleado Alfonso -> Operario
* Empleado Luis -> Operario -> Oficial
* Empleado Pablo -> Operario -> Tecnico

**2)**

Vamos a crear es la clase Operario que **hereda** de forma directa de la clase Empleado. Recuerde que en Java la herencia se expresa mediante la palabra **extends**. *Ejemplo: Operario extends Empleado indica que la clase Operario deriva o hereda de la clase Empleado.* La clase Operario hereda de Empleado el atributo nombre, los métodos get y set y el método toString(). El constructor de la clase recibe el nombre del empleado y lo pasa al constructor de la clase base mediante la instrucción **super**(nombre). En esta clase se modifica (**override**) el método toString() heredado de la clase base para que muestre un mensaje como pide el ejercicio.

**3)**

Escriba el código de la clase Oficial que hereda de la clase Operario. La clase Oficial hereda de Operario el atributo nombre, los métodos get y set y el método toString(). El constructor de la clase recibe el nombre del empleado y lo pasa al constructor de su clase base, en este caso la clase Operario, mediante la instrucción **super**. En esta clase también se modifica el método toString() heredado de su clase base para que muestre un mensaje como pide el ejercicio. Declare a la clase Oficial de tipo **final** y pruebe generar una nueva clase que hereda de ella. ¿Qué es lo que ocurre?

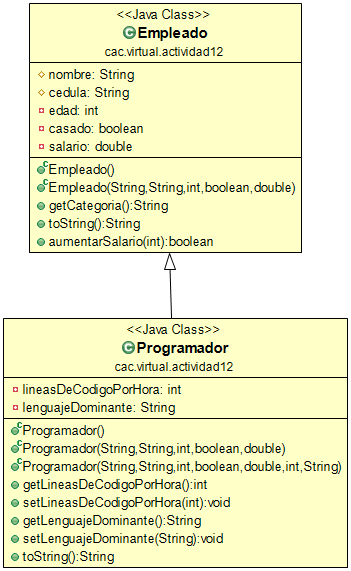
**4)**

Escribir la clase Directivo, similar a la clase Operario. Declare a la clase directivo de tipo **final**.

**5)**

Escribir la clase Tecnico, similar a la clase Oficial. Declare a la clase Tecnico de tipo **final**.

**6)**



La empresa informática “Codo a Codo Tech” necesita llevar un registro de todos sus empleados que se encuentran en la oficina central, para eso ha creado un diagrama de clases que debe incluir la clase Empleado con las siguientes características:

Atributos:

* nombre: tipo cadena (nombre y apellido)
* cedula: tipo cadena
* edad : entero (rango entre 18 y 45 años)
* casado: boolean
* salario: tipo numérico doble

Métodos:

* Cree un constructor con y sin parámetros de entrada
* Cree un método getCategoria que permita mostrar la clasificación según la edad de acuerdo al siguiente algoritmo:
  + Si edad es menor o igual a 21, Principiante
  + Si edad es >= 22 y <= 35, Intermedio
  + Si edad es >35, Senior
* Sobrescriba el método toString() para poder imprimir los datos del empleado por pantalla (puede utilizar salto de línea \n para separar los atributos).
* Cree un método que permita aumentar el salario en un porcentaje que sería pasado como parámetro al método.

**7)**

Teniendo en cuenta la clase creada en el ejercicio anterior, cree la clase Programador (especialización de Empleado). La clase Programador hereda de Empleado todos sus atributos y métodos. Los atributos que serán propios de la clase Programador serán:

* lineasDeCodigoPorHora : tipo entero
* lenguajeDominante: tipo cadena

Para la clase Programador deberá crear también el método constructor con y sin parámetros de entrada. Redefina algunos de los atributos de la clase Empleado con el tipo de modificador de acceso **protected**. ¿Qué es lo puede hacer ahora desde la clase Programador al haber redefinido así la clase Empleado?

**public** **class** Empleado {

**protected** String nombre;

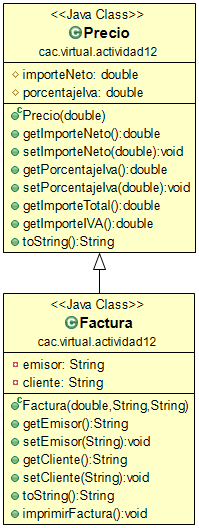
**protected** String cedula;

**private** **int** edad;

**private** **boolean** casado;

**private** **double** salario;

**8)**



Construir una clase denominada Factura, hija de la clase Precio y que posea 2 atributos específicos llamados emisor y cliente (que deberán ser los números de CUIT de ellos) y, al menos, 1 método llamado imprimirFactura que deberá mostrar los datos de esa factura.

La clase Precio tendrá los atributos que se listan a continuación, al menos un constructor, sus getters y setters y el método toString:

**public** **class** Precio {

**protected** **double** importeNeto;

**protected** **double** porcentajeIva;

La clase Factura tendrá los atributos anteriormente mencionados, al menos un constructor que invoque al constructor de la clase Precio, sus getters y setters y el método toString.

**public** **class** Factura **extends** Precio{

**private** String emisor;

**private** String cliente;

En ejemplo de ejecución del método imprimirFactura podría ser:

*Factura:*

*Emisor: 201234567891*

*Cliente: 33123456789*

*Precio:*

*Importe Neto: $100.0*

*IVA 21.0%: $21.0*

*Importe Total: $121.0*

**9)**

Crear una clase llamada Cuentaque tendrá los siguientes atributos: dniTitular y saldo(puede tener decimales). El dniTitular será obligatorio y el saldo es opcional. Crea dos constructores que cumplan con lo anterior. Crea sus métodos get y set (excepto para saldo que no deberá tener set) y toString.

La clase tendrá dos métodos más:

* depositar(double importe): se ingresa un importe a la cuenta, si el importe es negativo o cero, no se hará nada y el método devolverá false (error), caso contrario deberá retornar true (ok) y actualizar el saldo de la cuenta.
* extraer(double importe): se retira una cantidad determinada de la cuenta. Si la resta entre el saldo actual y el importe a extraer da negativo, retornar false; caso contrario, retornar true y actualizar el saldo de la cuenta.

**10)**

Reutilizando la clase Cuenta del ejercicio anterior, modifique los atributos de la clase para que tenga un objeto de tipo Persona como atributo en lugar de dniTitular (**composición**). Modifique el resto del código de la clase Cuenta para que su implementación no se vea alterada.

La clase Persona deberá respetar las siguientes condiciones:

* Sus atributos son: dni, nombre, edad, sexo (M y F, Masculino y Femenino). No queremos que se accedan directamente a ellos. Piensa que modificador de acceso es el más adecuado, también su tipo.
* Por defecto, todos los atributos menos el dni serán valores por defecto según su tipo (0 números, cadena vacía para String).
* Se implantarán varios constructores: un constructor que sólo solicitará el dni, un constructor con el dni y sexo, el resto por defecto, y un constructor con todos los atributos como parámetro.
* Métodos set de cada parámetro, excepto para DNI. getters para todos los parámetros.
* toString(): devuelve toda la información del objeto.
* esMayorDeEdad(): indica si es mayor de edad, devuelve un tipo de dato booleano.

Crea una clase instanciada que pida por teclado los datos necesarios para instanciar 3 objetos de tipo Persona utilizando los 3 constructores anteriormente realizados. En caso de ser mayores de edad, crearles una cuenta con la nueva clase Cuenta creada en este ejercicio y que contenga el correspondiente objeto de tipo Persona. Por último, mostrar la información de cada objeto Cuenta y Persona creado.

**11)**

Escriba un método para realizar la búsqueda del nombre de un cliente (Clase Cliente) guardado en un **ArrayList** de clientes. El cliente a buscar será ingresado por consola por el usuario. El método deberá devolver true en caso de que ese nombre exista, sino false. El método buscarCliente podría estar dentro de una clase llamada Local. La clase Cliente deberá tener al menos los atributos: id (int) y nombre (String), sus correspondientes getters y setters y su constructor Cliente(int id, String nombre).

**12)**

Un supermercado nos pide que hagamos una pequeña aplicación que almacene los productos pasados por el escáner. La aplicación debe almacenar Productos (clase) y la cantidad comprada. Para ello deberá tener un **ArrayList** de compras y cada elemento de la lista deberá tener un objeto de tipo Producto y una cantidad de tipo int. La compra al crearse contendrá un precio (puede generarlo aleatoriamente) y una cantidad asociada que también puede ser generado de esa forma si lo desea. El nombre del producto será básico (producto1, producto2, producto3, etc.). Cree la clase Producto, la clase Compra y la clase Supermercado que contendrá el ArrayList de compras. Calcule el precio total de una lista de entre 1 y 8 productos (aleatorio). Mostrar un ticket con todo lo vendido y el precio final como se hacen en los supermercados. Más o menos con este formato, lo importante son los datos, no el estilo:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Cantidad\*\*\*Precio\*\*\*\*\*Total

Producto1 5 3.5 17.5

Producto2 7 2.5 17.5

Precio final 35

**Nota:** este ejercicio es una simplificación de un caso real. En un sistema real, la compra debería estar asociada a un Cliente y cada compra a su vez tendría un ArrayList de ítems de la compra que a su vez contendría la información del producto y su correspondiente cantidad comprada.

**RESPUESTAS:**

**1), 2) , 3) , 4) Y 5)**

public class Empleado {

private String Nombre;

public Empleado() {

}

public Empleado(String Nombre) {

this.Nombre = Nombre;

}

public String getNombre() {

return Nombre;

}

public final void setNombre(String Nombre) {

this.Nombre = Nombre;

}

public String obtenerCargo() {

return "Empleado ";

}

@Override

public String toString() {

return obtenerCargo() + Nombre ;

}

}

public class Directivo extends Empleado {

public Directivo(String Nombre) {

super(Nombre);

}

public final String obtenerCargo() {

return "Directivo";

}

@Override

public String toString() {

return "Empleado " + getNombre() + "---> " + obtenerCargo() ;

}

}

public class Operario extends Empleado{

public Operario(String Nombre) {

super(Nombre);

}

public String obtenerCargo() {

return " Operario";

}

@Override

public String toString() {

return "Empleado " + getNombre() + " ---> "+ obtenerCargo();

}

}

public class Oficial extends Operario {

public Oficial(String Nombre) {

super(Nombre);

}

@Override

public final String obtenerCargo() {

return "Operario";

}

public final String especialidad() {

return "Oficial";

}

@Override

public String toString() {

return "Empleado " + getNombre() + " ---> " + obtenerCargo()+" ---> " + especialidad() ;

}

}

public class Tecnico extends Operario {

public Tecnico(String Nombre) {

super(Nombre);

}

@Override

public final String obtenerCargo() {

return " Operario ";

}

public final String especialidad() {

return " Tecnico ";

}

@Override

public String toString() {

return "Empleado " + getNombre() + " ---> "+ obtenerCargo()+ "--->" + especialidad();

}

}

6) Y 7)

public class Empleado {

protected String nombre;

protected String cedula;

protected int edad;

protected boolean casado;

protected double salario;

public Empleado() {

}

public Empleado(String nombre, String cedula, int edad, boolean casado, double salario) {

this.nombre = nombre;

this.cedula = cedula;

if (edad >= 18 && edad <= 45) {

this.edad = edad;

} else {

System.out.println("La edad debe estar entre 18 y 45 años.");

}

this.casado = casado;

this.salario = salario;

}

public double getSalario() {

return salario;

}

public void setSalario(double salario) {

this.salario = salario;

}

public boolean isCasado() {

return casado;

}

public void setCasado(boolean casado) {

this.casado = casado;

}

public int getEdad() {

return edad;

}

public void setEdad(int edad) {

this.edad = edad;

}

public String getCedula() {

return cedula;

}

public void setCedula(String cedula) {

this.cedula = cedula;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public void aumentarSalario(double porcentajeAumento) {

salario += salario \* (porcentajeAumento / 100);

}

public String getCategoria() {

if (edad <= 21) {

return "Principiante ";

} else { if ( edad >= 22 && edad <= 35 ) {

return " Intermedio ";

} else {

return " Senior ";

}

}

}

@Override

public String toString() {

return " Nombre= " + nombre + " cedula= " + cedula + " edad= " + edad + " casado= " + casado + " salario= " + salario + " categoria: " + getCategoria();

}

}

public class Programador extends Empleado {

private int lineaDeCodigoPorHora;

private String lenguajeDominante;

public Programador() {

}

public Programador(int lineaDeCodigoPorHora, String lenguajeDominante, String nombre, String cedula, int edad, boolean casado, double salario) {

super(nombre, cedula, edad, casado, salario);

this.lineaDeCodigoPorHora = lineaDeCodigoPorHora;

this.lenguajeDominante = lenguajeDominante;

}

public String getLenguajeDominante() {

return lenguajeDominante;

}

public void setLenguajeDominante(String lenguajeDominante) {

this.lenguajeDominante = lenguajeDominante;

}

public int getLineaDeCodigoPorHora() {

return lineaDeCodigoPorHora;

}

public void setLineaDeCodigoPorHora(int lineaDeCodigoPorHora) {

this.lineaDeCodigoPorHora = lineaDeCodigoPorHora;

}

@Override

public String toString() {

return " Linea De Codigo Por Hora= " + lineaDeCodigoPorHora + " Lenguaje Dominante= " + lenguajeDominante ;

}

}

public class Test {

public static void main(String[] args) {

// Creación de un objeto Empleado usando el constructor con parámetros

Empleado empleado1 = new Empleado("Juan Perez", "1234567890", 30, false, 300000.0);

System.out.println("Datos del empleado:");

System.out.println(empleado1); // Llamada al método toString() para imprimir los datos del empleado

// Accediendo a un método de la clase Empleado

empleado1.aumentarSalario(10); // Aumentar salario en un 10%

// Mostrando los datos actualizados del empleado

System.out.println("Datos actualizados del empleado:");

System.out.println(empleado1);

// Creación de un objeto Programador usando el constructor con parámetros

Programador programador1 = new Programador(100, "Java", "Carlos Gutierrez ", "098765432",45, false, 300000);

System.out.println("Datos del programador:");

System.out.println(programador1); // Imprimiendo los datos del programador

// Accediendo a un método de la clase Empleado desde Programador

programador1.aumentarSalario(15); // Aumentar salario en un 15%

// Mostrando los datos actualizados del programador

System.out.println("Datos actualizados del programador:");

System.out.println(programador1);

}

}

8)

public class Precio {

protected double importeNeto;

protected double porcentajeIva;

public Precio() {

}

public Precio(double importeNeto, double porcentajeIva) {

this.importeNeto = importeNeto;

this.porcentajeIva = porcentajeIva;

}

public double getPorcentajeIva( double Iva) {

return porcentajeIva ;

}

public void setPorcentajeIva(double porcentajeIva) {

this.porcentajeIva = porcentajeIva \* importeNeto ;

}

public double getImporteNeto() {

return importeNeto;

}

public void setImporteNeto(double importeNeto) {

this.importeNeto = importeNeto;

}

@Override

public String toString() {

return "Importe Neto: $" + importeNeto + "\n" +

"IVA " + porcentajeIva + "%: $" + (importeNeto \* porcentajeIva / 100) + "\n" +

"Importe Total: $" + (importeNeto + (importeNeto \* porcentajeIva / 100)) + "\n";

}

}

public class Factura extends Precio {

private String emisor;

private String cliente;

public Factura() {

}

public Factura(String emisor, String cliente, double importeNeto, double porcentajeIva) {

super(importeNeto, porcentajeIva);

this.emisor = emisor;

this.cliente = cliente;

}

public String getCliente() {

return cliente;

}

public void setCliente(String cliente) {

this.cliente = cliente;

}

public String getEmisor() {

return emisor;

}

public void setEmisor(String emisor) {

this.emisor = emisor;

}

public void imprimirFactura() {

System.out.println("Factura:");

System.out.println(" Emisor: " + emisor);

System.out.println(" Cliente: " + cliente);

System.out.println("Precio:");

System.out.println(super.toString());

}

@Override

public String toString() {

return " emisor= " + emisor + " cliente= " + cliente ;

}

}

public class Test {

public static void main(String[] args) {

Factura f1 = new Factura( " Viajes S.A. ", "Carlos Gutierrez" ,3000 , 1.21 );

System.out.println("Factura ");

System.out.println(f1);

Precio p1 = new Precio(3000 , 21 );

System.out.println(" Precio ");

System.out.println(p1);

}

}

9)

import java.util.Scanner;

public class Cuenta {

private String dniTitular;

private double saldo;

// Constructor con saldo opcional

public Cuenta(String dniTitular, double saldo) {

this.setDniTitular(dniTitular);

this.saldo = saldo;

}

// Constructor sin saldo (saldo por defecto a 0)

public Cuenta(String dniTitular) {

this(dniTitular, 0.00);

}

//gettters y setters para dniTitular

public String getDniTitular() {

return dniTitular;

}

public void setDniTitular(String dniTitular) {

this.dniTitular = dniTitular;

}

//get para saldo

public double getSaldo() {

return saldo;

}

// Método para depositar dinero en la cuenta

public boolean depositar(double importe) {

if (importe > 0) {

saldo += importe;

return true; // Depósito exitoso

} else {

return false; // Error: importe negativo o cero

}

}

// Método para extraer dinero de la cuenta

public boolean extraer(double importe) {

if (saldo - importe >= 0) {

saldo -= importe;

return true; // Extracción exitosa

} else {

return false; // Error: saldo insuficiente

}

}

// toString

@Override

public String toString() {

return "Cuenta{DNI num: '" + dniTitular + "', saldo = $" + saldo + "}";

}

// Método para obtener entrada de texto del usuario

private static String obtenerEntradaTexto(String mensaje) {

Scanner lector = new Scanner(System.in);

System.out.print(mensaje + ": ");

return lector.next();

}

public static void main(String[] args) {

// Obtener datos por teclado

String dniTitular = obtenerEntradaTexto("Ingrese el DNI del titular");

double saldoInicial = Double.parseDouble(obtenerEntradaTexto("Ingrese el saldo inicial"));

// Crear una cuenta con los datos ingresados

Cuenta miCuenta = new Cuenta(dniTitular, saldoInicial);

// Mostrar información de la cuenta

System.out.println("Informacion inicial de la cuenta:");

System.out.println(miCuenta);

System.out.println("---------------------------------");

// Depositar dinero en la cuenta

double deposito = Double.parseDouble(obtenerEntradaTexto("Ingrese el monto a depositar"));

if (miCuenta.depositar(deposito)) {

System.out.println("Deposito exitoso. Saldo actual: $" + miCuenta.getSaldo());

System.out.println("---------------------------------");

} else {

System.out.println("Error al depositar. Importe no valido.");

}

// Extraer dinero de la cuenta

double extraccion = Double.parseDouble(obtenerEntradaTexto("Ingrese el monto a extraer"));

if (miCuenta.extraer(extraccion)) {

System.out.println("Extraccion exitosa. Saldo actual: $" + miCuenta.getSaldo());

System.out.println("---------------------------------");

} else {

System.out.println("Error al extraer. Saldo insuficiente.");

}

}

}

12)

public class Producto {

private String nombre;

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

}

public class Compra {

private String nombreProducto;

private int cantidad;

private double precio;

public Compra(String nombreProducto, int cantidad, double precio) {

this.nombreProducto = nombreProducto;

this.cantidad = cantidad;

this.precio = precio;

}

public double getPrecioTotal() {

return precio \* cantidad;

}

public String toString() {

return String.format("%-15s %5d %10.2f %10.2f", nombreProducto, cantidad, precio, getPrecioTotal());

}

}

import java.util.ArrayList;

public class Supermercado {

private ArrayList<Compra> compras;

public Supermercado() {

compras = new ArrayList<>();

}

public void agregarCompra(String nombreProducto, int cantidad, double precio) {

compras.add(new Compra(nombreProducto, cantidad, precio));

}

public double calcularPrecioTotal() {

double total = 0;

for (Compra compra : compras) {

total += compra.getPrecioTotal();

}

return total;

}

public void mostrarTicket() {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Cantidad\*\*\*Precio\*\*\*\*\*Total");

for (Compra compra : compras) {

System.out.println(compra.toString());

}

System.out.println("Precio final: " + calcularPrecioTotal());

}

}

import java.util.Random;

public class Test {

public static void main(String[] args) {

Supermercado supermercado = new Supermercado();

Random random = new Random();

int cantidadProductos = random.nextInt(8) + 1;

for (int i = 1; i <= cantidadProductos; i++) {

String nombreProducto = "Producto" + i;

int cantidad = random.nextInt(10) + 1; // Cantidad entre 1 y 10

double precio = random.nextDouble() \* 10; // Precio entre 0 y 10

supermercado.agregarCompra(nombreProducto, cantidad, precio);

}

supermercado.mostrarTicket();

}

}

9)

import java.util.Scanner;

public class Cuenta {

private String dniTitular;

private double saldo;

// Constructor con saldo opcional

public Cuenta(String dniTitular, double saldo) {

this.setDniTitular(dniTitular);

this.saldo = saldo;

}

// Constructor sin saldo (saldo por defecto a 0)

public Cuenta(String dniTitular) {

this(dniTitular, 0.00);

}

// Métodos get y set para dniTitular

public String getDniTitular() {

return dniTitular;

}

public void setDniTitular(String dniTitular) {

this.dniTitular = dniTitular;

}

// Método get para saldo

public double getSaldo() {

return saldo;

}

// Método para depositar dinero en la cuenta

public boolean depositar(double importe) {

if (importe > 0) {

saldo += importe;

return true; // Depósito exitoso

} else {

return false; // Error: importe negativo o cero

}

}

// Método para extraer dinero de la cuenta

public boolean extraer(double importe) {

if (saldo - importe >= 0) {

saldo -= importe;

return true; // Extracción exitosa

} else {

return false; // Error: saldo insuficiente

}

}

private static String obtenerEntradaTexto(String mensaje) {

Scanner lector = new Scanner(System.in);

System.out.print(mensaje + ": ");

return lector.next();

}

// toString

@Override

public String toString() {

return "Cuenta{DNI num: '" + dniTitular + "', saldo = $" + saldo + "}";

}

public static void main(String[] args) {

// Obtener datos por teclado

String dniTitular = obtenerEntradaTexto("Ingrese el DNI del titular");

double saldoInicial = Double.parseDouble(obtenerEntradaTexto("Ingrese el saldo inicial"));

// Crear una cuenta con los datos ingresados

Cuenta miCuenta = new Cuenta(dniTitular, saldoInicial);

// Mostrar información de la cuenta

System.out.println("Informacion inicial de la cuenta:");

System.out.println(miCuenta);

System.out.println("---------------------------------");

// Depositar dinero en la cuenta

double deposito = Double.parseDouble(obtenerEntradaTexto("Ingrese el monto a depositar"));

if (miCuenta.depositar(deposito)) {

System.out.println("Deposito exitoso. Saldo actual: $" + miCuenta.getSaldo());

System.out.println("---------------------------------");

} else {

System.out.println("Error al depositar. Importe no valido.");

}

// Extraer dinero de la cuenta

double extraccion = Double.parseDouble(obtenerEntradaTexto("Ingrese el monto a extraer"));

if (miCuenta.extraer(extraccion)) {

System.out.println("Extraccion exitosa. Saldo actual: $" + miCuenta.getSaldo());

System.out.println("---------------------------------");

} else {

System.out.println("Error al extraer. Saldo insuficiente.");

}

}

}

10)

import java.util.Scanner;

public class Cuenta {

private Persona titular;

private double saldo;

// Constructor con saldo opcional

public Cuenta(Persona titular, double saldo) {

this.titular = titular;

this.saldo = saldo;

}

// Constructor sin saldo (saldo por defecto a 0)

public Cuenta(Persona titular) {

this(titular, 0.0);

}

// getters y setters para titular

public Persona getTitular() {

return titular;

}

public void setTitular(Persona titular) {

this.titular = titular;

}

// getter para saldo

public double getSaldo() {

return saldo;

}

// Método para depositar dinero en la cuenta

public boolean depositar(double importe) {

if (importe > 0) {

saldo += importe;

return true; // Depósito exitoso

} else {

return false; // Error: importe negativo o cero

}

}

// Método para extraer dinero de la cuenta

public boolean extraer(double importe) {

if (saldo - importe >= 0) {

saldo -= importe;

return true; // Extracción exitosa

} else {

return false; // Error: saldo insuficiente

}

}

// toString

@Override

public String toString() {

return "Cuenta { Titular = " + titular + ", Saldo = " + saldo + "}";

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Crear tres objetos de tipo Persona con datos ingresados por teclado

System.out.println("Ingrese los datos para la primera persona:");

Persona persona1 = ingresarDatosPersona();

System.out.println("Ingrese los datos para la segunda persona:");

Persona persona2 = ingresarDatosPersona();

System.out.println("Ingrese los datos para la tercera persona:");

Persona persona3 = ingresarDatosPersona();

// Verificar si son mayores de edad y crear una cuenta si es el caso

if (persona1.esMayorDeEdad()) {

Cuenta cuenta1 = new Cuenta(persona1);

System.out.println("Cuenta creada para la primera persona: ");

System.out.println(cuenta1);

System.out.println("----------------------------------------------");

}

if (persona2.esMayorDeEdad()) {

Cuenta cuenta2 = new Cuenta(persona2);

System.out.println("Cuenta creada para la segunda persona: ");

System.out.println(cuenta2);

System.out.println("----------------------------------------------");

}

if (persona3.esMayorDeEdad()) {

Cuenta cuenta3 = new Cuenta(persona3);

System.out.println("Cuenta creada para la tercera persona: ");

System.out.println(cuenta3);

System.out.println("----------------------------------------------");

}

}

// Método para ingresar datos de una persona por teclado

private static Persona ingresarDatosPersona() {

Scanner scanner = new Scanner(System.in);

System.out.print("Ingrese el DNI: ");

String dni = scanner.nextLine();

System.out.print("Ingrese el nombre: ");

String nombre = scanner.nextLine();

System.out.print("Ingrese la edad: ");

int edad = scanner.nextInt();

scanner.nextLine(); // Limpia el buffer de entrada

System.out.print("Ingrese el sexo (M/F): ");

char sexo = scanner.next().charAt(0);

System.out.println("----------------------------------------------");

return new Persona(dni, nombre, edad, sexo);

}

}

import java.util.Scanner;

class Persona {

private String dni;

private String nombre;

private int edad;

private char sexo;

// Constructor solo con DNI

public Persona(String dni) {

this.dni = dni;

}

// Constructor con DNI y sexo, el resto por defecto

public Persona(String dni, char sexo) {

this(dni);

this.sexo = sexo;

}

// Constructor con todos los atributos como parámetro

public Persona(String dni, String nombre, int edad, char sexo) {

this(dni, sexo);

this.nombre = nombre;

this.edad = edad;

}

// setters para cada parámetro, excepto para DNI

public void setNombre(String nombre) {

this.nombre = nombre;

}

public void setEdad(int edad) {

this.edad = edad;

}

public void setSexo(char sexo) {

this.sexo = sexo;

}

// getters para cada parámetro

public String getDni() {

return dni;

}

public String getNombre() {

return nombre;

}

public int getEdad() {

return edad;

}

public char getSexo() {

return sexo;

}

// toString

@Override

public String toString() {

return "Persona {DNI = '" + dni + "', Nombre = '" + nombre + "', Edad = " + edad + ", Sexo = " + sexo + "}";

}

// Método para verificar si es mayor de edad

public boolean esMayorDeEdad() {

return edad >= 18;

}

}

11)

import java.util.Scanner;

import java.util.ArrayList;

public class Test {

public static void main(String[] args) {

Local local = new Local();

// Agregar clientes al local

local.agregarCliente(new Cliente(1, "Juan"));

local.agregarCliente(new Cliente(2, "María"));

local.agregarCliente(new Cliente(3, "Pedro"));

// Solicitar el nombre del cliente a buscar

Scanner scanner = new Scanner(System.in);

System.out.print("Ingrese el nombre del cliente a buscar: ");

String nombreBuscar = scanner.nextLine();

// Realizar la búsqueda y mostrar el resultado

boolean clienteEncontrado = local.buscarCliente(nombreBuscar);

if (clienteEncontrado) {

System.out.println("El cliente con nombre '" + nombreBuscar + "' existe en la lista.");

} else {

System.out.println("El cliente con nombre '" + nombreBuscar + "' NO existe en la lista.");

}

}

}

class Cliente {

private int id;

private String nombre;

public Cliente(int id, String nombre) {

this.id = id;

this.nombre = nombre;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

}

class Local {

private ArrayList<Cliente> clientes;

public Local() {

this.clientes = new ArrayList<>();

}

// Agregar un cliente al ArrayList

public void agregarCliente(Cliente cliente) {

clientes.add(cliente);

}

// Método para buscar un cliente por nombre

public boolean buscarCliente(String nombre) {

for (Cliente cliente : clientes) {

if (cliente.getNombre().equalsIgnoreCase(nombre)) {

return true; // El nombre del cliente existe en la lista

}

}

return false; // No se encontró el nombre del cliente en la lista

}

}