Informe Programación Estructurada - Programación Orientada A Objetos JAVA

Planear Actividades De Construcción Del Software De Acuerdo Con El Diseño Establecido

Aprendiz:

Santiago Carranza Carrillo

Ficha:

2848530-A

Instructor:

Julio Roberto Galvis Cardozo

SENA

Centro De Diseño Y Metrología

2 De diciembre 2024

Índice

Introducción.....2
Cod Programación Estructurada Notas 7
Cod Programación Orientada a Objetos Empleado 11
Cod Programación Orientada a Objetos Clase Empleado 14
Cod Programación Orientada a Objetos Empresa 17
Cod Programación Orientada A Objetos Estudiante.....21
Cod Programación Orientada a Objetos Clase Estudiante 24
Conclusión 29

Introducción

Este informe presenta una serie de implementaciones en Java destinadas a gestionar empleados y estudiantes, destacando tanto un enfoque estructurado como orientado a objetos. A través de los ejemplos, se ilustran conceptos clave de programación, como el uso de colecciones, la encapsulación, la reutilización de código mediante clases y métodos, y la interacción con el usuario a través de la consola.

En la sección dedicada a la gestión de empleados, se abordan dos enfoques principales: el estructurado, que emplea arreglos dinámicos para almacenar información, y el orientado a objetos, donde entidades como Empleado y Empresa encapsulan datos y lógica específica. De manera similar, los programas relacionados con estudiantes implementan ambos paradigmas para calcular promedios de notas y determinar su estado de aprobación.

Cod Programación Estructurada Empleados

```java
import java.util.ArrayList;
import java.util.Scanner;

public class EmpleadosEstructurada{

  Run | Debug
  public static void main(String[] args) {

    String nombre, cargo;

    double salario, total = 0, minSalario, maxSalario;

    int cantidad, posicion = 0;

    ArrayList<String> nombres = new ArrayList<String>();

    ArrayList<Double> salarios = new ArrayList<Double>();

    ArrayList<String> cargos = new ArrayList<String>();

    Scanner cantidades = new Scanner(System.in);
    Scanner names = new Scanner(System.in);
    Scanner salari= new Scanner(System.in);
    Scanner carg = new Scanner(System.in);

     System.out.println(x:"Ingresar cantidad de empleados");
     cantidad = cantidades.nextInt();

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

          System.out.println(x:"Ingrese el nombre del empleado");
          nombre = names.next();
          System.out.println(x:"Ingrese el cargo del empleado");
          cargo = carg.next();
          System.out.println(x:"Ingrese el salario del empleado");
          salario = salari.nextDouble();


          nombres.add(nombre);
          salarios.add(salario);
          cargos.add(cargo);


     }

    System.out.println("El total de empleados es: " + cantidad);

    System.out.println(x:"Los nombres y salarios son: ");


    for (int i =0; i < cantidad ; i++) {
        System.out.println("Nombres: " + nombres.get(i) + " Salarios: " + salarios.get(i));

      }

    for (int i =0; i < cantidad ; i++) {

          total = total + salarios.get(i);

```

```
58
59          }
60
61          System.out.println("El total de salarios pagados es de: " +total);
62
63          System.out.println(x:"El empleado que mas dinero gana es: ");
64          maxSalario = salarios.get(index:0);
65
66          for (int i =1; i < cantidad ; i++) {
67              if(salarios.get(i) > maxSalario) {
68                  maxSalario = salarios.get(i);
69                  posicion = i;
70              }
71          }
72
73
74          System.out.println("Nombre: " + nombres.get(posicion) + " Cargo: " + cargos.get(posicion) + " Salario: " + salarios.get(posicion));
75
76
77
78          System.out.println(x:"El empleado que menos dinero gana es: ");
79          minSalario = salarios.get(index:0);
80
81
82          for (int i =1; i < cantidad ; i++) {
83
84              if(salarios.get(i) < minSalario) {
85                  minSalario = salarios.get(i);
86                  posicion = i;
87              }
88          }
89
90          System.out.println("Nombre: " + nombres.get(posicion) + " Cargo: " + cargos.get(posicion) + " Salario: " + salarios.get(posicion));
91      }
92
93  }
```

Explicación del código:

1. **Importación de bibliotecas:**

```
import java.util.ArrayList;
import java.util.Scanner;
```

ArrayList: Es una estructura de datos dinámica que permite almacenar listas de elementos. Aquí se usa para almacenar nombres, salarios y cargos de empleados.

Scanner: Se utiliza para leer entradas del usuario desde la consola.

2. **Declaración de la clase y método principal:**

```
public class EmpleadosEstructurada {
  public static void main(String[] args) {
```

La clase **EmpleadosEstructurada** contiene el método **main**, que es el punto de entrada del programa.

3. **Declaración de variables**

```
String nombre, cargo;
double salario, total = 0, minSalario, maxSalario;
int cantidad, posicion = 0;
```

Variables de texto (nombre, cargo): Para almacenar el nombre y el cargo de un empleado.

Variables numéricas (salario, total, minSalario, maxSalario): Para manejar salarios, el total acumulado, y los salarios mínimo y máximo.

cantidad: Representa el número total de empleados ingresados.

posicion: Guarda la posición en las listas del empleado con el salario mínimo o máximo.

4. **Creación de listas para datos de empleados:**

```
ArrayList<String> nombres = new ArrayList<String>();
ArrayList<Double> salarios = new ArrayList<Double>();
ArrayList<String> cargos = new ArrayList<String>();
```

Se usan tres listas:

nombres: Para almacenar los nombres de los empleados.

salarios: Para los salarios.

cargos: Para los cargos.

5. **Inicialización de objetos:**

```
Scanner cantidades = new Scanner(System.in);
Scanner names = new Scanner(System.in);
Scanner salari = new Scanner(System.in);
Scanner carg = new Scanner(System.in);
```

Estos escáneres se usan para leer diferentes tipos de datos ingresados por el usuario.

6. **Lectura de la cantidad de empleados:**

```
System.out.println("Ingresar cantidad de empleados");
cantidad = cantidades.nextInt();
```

El usuario ingresa el número total de empleados que quiere registrar.

7. **Ciclo para ingresar datos de empleados:**

```
for (int i = 1; i <= cantidad; i++) {
   System.out.println("Ingrese el nombre del empleado");
   nombre = names.next();
   System.out.println("Ingrese el cargo del empleado");
   cargo = carg.next();
   System.out.println("Ingrese el salario del empleado");
   salario = salari.nextDouble();
```

```
    nombres.add(nombre);
    salarios.add(salario);
    cargos.add(cargo);
}
```

Por cada empleado, el programa:

- Solicita su nombre, cargo y salario.
- Agrega estos datos a las listas correspondientes.

8. **Mostrar cantidad de empleados y sus datos:**

```
System.out.println("El total de empleados es: " + cantidad);
for (int i = 0; i < cantidad; i++) {
    System.out.println("Nombres: " + nombres.get(i) + " Salarios: " + salarios.get(i));
}
```

- Se imprime la cantidad total de empleados.
- Luego, se recorren las listas para mostrar los nombres y salarios de cada empleado.

9. **Calcular el total de salarios:**

```
for (int i = 0; i < cantidad; i++) {
    total = total + salarios.get(i);
}
System.out.println("El total de salarios pagados es de: " + total);
```

- El programa suma todos los salarios almacenados en la lista salarios.
- Muestra el total acumulado.

10. **Encontrar el empleado con el salario más alto:**

```
maxSalario = salarios.get(0);
for (int i = 1; i < cantidad; i++) {
    if (salarios.get(i) > maxSalario) {
        maxSalario = salarios.get(i);
        posicion = i;
    }
}
System.out.println("Nombre: " + nombres.get(posicion) + " Cargo: " + cargos.get(posicion) + "
Salario: " + salarios.get(posicion));
```

- Se inicializa maxSalario con el primer salario de la lista.

- Se recorre la lista para encontrar el salario más alto.

- Se almacena la posición del empleado con el salario más alto y se muestra su información.

11. Encontrar el empleado con el salario más bajo:

```
minSalario = salarios.get(0);

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

   if (salarios.get(i) < minSalario) {

      minSalario = salarios.get(i);

      posicion = i;

   }

}

System.out.println("Nombre: " + nombres.get(posicion) + " Cargo: " + cargos.get(posicion) + "
Salario: " + salarios.get(posicion));
```

Conclusión:

Este programa permite registrar datos de empleados (nombre, cargo, salario).

Calcular y mostrar el total de salarios pagados.

Los datos del empleado con el salario más alto.

Los datos del empleado con el salario más bajo.

Cod Programación Estructurada Notas

```java
import java.util.Scanner;

public class NotasEstructurada {
    public static void main(String[] args) throws Exception {

    String name;
    Double noteParcial1;
    Double noteParcial2;
    Double noteFinal;

    Scanner nombre = new Scanner(System.in);
    Scanner nota1 = new Scanner(System.in);
    Scanner nota2 = new Scanner(System.in);

for (int i = 1 ; i < 4 ; i++)
{
    System.out.println("Ingrese el nombre del " + i + " Estudiante: ");
    name = nombre.next();
```

```
    System.out.println("Ingrese la nota 1 del parcial del " + i + "
Estudiante: ");
    noteParcial1 = nota1.nextDouble();
    System.out.println("Ingrese la nota 2 del parcial del " + i + "
Estudiante: ");
    noteParcial2 = nota2.nextDouble();

    noteFinal = (noteParcial1 + noteParcial2) / 2;

    System.out.println("Informacion del Estudiante: " + name + " Nota 1: " +
noteParcial1 + " Nota 2: " + noteParcial2 + "  Resultado del promedio de las
2 notas: " + noteFinal );

    if (noteFinal < 3) {

        System.out.println("Usted: " + name + " Reprobo");

    }
    else {
        System.out.println("Usted: " + name + " Aprobo");
    }

}

    }
}
```

Explicación del código:

1. **Importación de la biblioteca Scanner:**

import java.util.Scanner;

Se importa la clase Scanner, que permite leer entradas del usuario desde la consola.

2. **Declaración de la clase y método principal:**

public class NotasEstructurada {

public static void main(String[] args) throws Exception {

Clase NotasEstructurada: Contiene el programa.

Método main: Es el punto de entrada del programa.

throws Exception: Permite manejar excepciones que podrían ocurrir durante la ejecución.

3. **Declaración de variables:**

String name;

Double noteParcial1;

Double noteParcial2;

Double noteFinal;

name: Almacena el nombre del estudiante.

noteParcial1 y noteParcial2: Guardan las dos notas parciales de un estudiante.

noteFinal: Calcula y almacena el promedio de las dos notas parciales.

4. **Creación de objetos:**

Scanner nombre = new Scanner(System.in);

Scanner nota1 = new Scanner(System.in);

Scanner nota2 = new Scanner(System.in);

Tres objetos Scanner se usan para:

- Leer el nombre del estudiante.

- Leer las dos notas parciales.

5. **Ciclo FOR para registrar y procesar datos de tres estudiantes:**

for (int i = 1; i < 4; i++) {

El programa itera 3 veces (una vez por cada estudiante) y en cada iteración, se realizan las siguientes acciones:

Solicitar el nombre del estudiante:

System.out.println("Ingrese el nombre del " + i + " Estudiante: ");

name = nombre.next();

Se solicita el nombre del estudiante y se almacena en la variable name.

Solicitar las notas parciales:

System.out.println("Ingrese la nota 1 del parcial del " + i + " Estudiante: ");

noteParcial1 = nota1.nextDouble();

System.out.println("Ingrese la nota 2 del parcial del " + i + " Estudiante: ");

noteParcial2 = nota2.nextDouble();

Se solicitan las dos notas parciales, que se almacenan en noteParcial1 y noteParcial2.

Calcular el promedio final:

noteFinal = (noteParcial1 + noteParcial2) / 2;

Se calcula el promedio de las dos notas parciales y se almacena en noteFinal.

Mostrar la información del estudiante:

System.out.println("Informacion del Estudiante: " + name +

```
" Nota 1: " + noteParcial1 +
" Nota 2: " + noteParcial2 +
"  Resultado del promedio de las 2 notas: " + noteFinal);
```

Se imprime:

El nombre del estudiante.

Las dos notas parciales.

El promedio final.

Determinar si aprobó o reprobó:

```
if (noteFinal < 3) {
   System.out.println("Usted: " + name + " Reprobo");
} else {
   System.out.println("Usted: " + name + " Aprobo");
}
```

- Si el promedio final es menor a 3, el estudiante reprobó.

- Si el promedio final es 3 o mayor, el estudiante aprobó.

Se imprime el resultado para el estudiante.

6. **Cierre del programa**

El ciclo FOR se repite para procesar los datos de tres estudiantes. Al finalizar, se muestra la información de todos ellos, incluyendo si aprobaron o reprobaron.

Cod Programación Orientada a Objetos Empleado

```java
import java.util.Scanner;


public class ejercicioEmpleadoPOO2 {
    Run | Debug
    public static void main(String[] args) throws Exception {

        int cantidad;
        String nombre;
        String cargo;
        double salario;

        Empresa empresa = new Empresa();


        Scanner cantidades = new Scanner(System.in);

        Scanner nombres = new Scanner(System.in);

        Scanner cargos = new Scanner(System.in);

        Scanner salarios = new Scanner(System.in);

        System.out.println(x:"Ingrese cantidad de empleados");
        cantidad = cantidades.nextInt();

        for (int i = 0; i <= cantidad ; i++) {
            // Solicitar los datos del empleado
            System.out.println(x:"Ingresar nombre del empleado");
            nombre = nombres.next();
            System.out.println(x:"Ingresar cargo del empleado");
            cargo = cargos.next();
            System.out.println(x:"Ingresar salario del empleado");
            salario = salarios.nextInt();

            empresa.contratarEmpleado(new Empleado(cargo, nombre, salario));



        }

        System.out.println("Numero total de empleados: " + empresa.getTotalEmpleados());

        System.out.println(x:"Nombres y salarios de los empleados son: ");
        empresa.nombreSalario();

        System.out.println("Total de dinero pagado a todos es de: " + empresa.getTotalSalarios());

        empresa.empleadoMayorSalario();
        empresa.empleadoMenorSalario();
        }
    }

```

Explicación del código:

1. Importación de Librerías

import java.util.Scanner;

Se importa la clase Scanner que permite la entrada de datos por parte del usuario.

2. Definición de la Clase Principal

public class ejercicioEmpleadoPOO2 {

public static void main(String[] args) throws Exception {

La clase principal es ejercicioEmpleadoPOO2.

El método main es el punto de entrada de la aplicación.

3. Declaración de Variables

```
int cantidad;

String nombre;

String cargo;

double salario;
```

Variables para capturar datos de los empleados:

Cantidad: Cantidad de empleados a ingresar.

Nombre: Nombre del empleado.

Cargo: Cargo que ocupa el empleado.

Salario: Salario que percibe el empleado.

4. Instanciar la Clase Empresa

```
Empresa empresa = new Empresa();
```

Se crea una instancia de la clase Empresa. Aunque la clase Empresa no está definida en el fragmento que proporcionaste, por el nombre podemos inferir que se usa para manejar la lógica relacionada con los empleados, como contratación, contar empleados, calcular salarios, etc.

5. Configuración de Objetos Scanner

```
Scanner cantidades = new Scanner(System.in);

Scanner nombres = new Scanner(System.in);

Scanner cargos = new Scanner(System.in);

Scanner salarios = new Scanner(System.in);
```

Se crean múltiples instancias de Scanner para capturar entradas. Sin embargo, es redundante crear varias instancias de Scanner. Se puede usar una sola instancia para todo el proceso.

6. Solicitar la Cantidad de Empleados

```
System.out.println("Ingrese cantidad de empleados");

cantidad = cantidades.nextInt();
```

Se solicita al usuario que indique cuántos empleados desea ingresar.

7. Bucle para Capturar Datos de Empleados

```
for (int i = 0; i <= cantidad ; i++) {

   System.out.println("Ingresar nombre del empleado");
```

```
    nombre = nombres.next();
    System.out.println("Ingresar cargo del empleado");
    cargo = cargos.next();
    System.out.println("Ingresar salario del empleado");
    salario = salarios.nextInt();
    empresa.contratarEmpleado(new Empleado(cargo, nombre, salario));
}
```

Entrada de datos:

El usuario introduce el nombre, cargo y salario de cada empleado.

Se crea una nueva instancia de la clase Empleado con los datos ingresados.

Método contratarEmpleado:

Se asume que este método es parte de la clase Empresa. Este método debería agregar el empleado a una lista interna que la clase Empresa mantenga.

8. Mostrar Información Básica

```
System.out.println("Numero total de empleados: " + empresa.getTotalEmpleados());
```

Se imprime el número total de empleados. Esto se realiza a través del método getTotalEmpleados() de la clase Empresa.

```
System.out.println("Nombres y salarios de los empleados son: ");
empresa.nombreSalario();
```

Se asume que nombreSalario() imprimirá la información de los empleados (nombres y salarios). Este método debe estar implementado en la clase Empresa.

```
System.out.println("Total de dinero pagado a todos es de: " + empresa.getTotalSalarios());
```

Se calcula y muestra la suma total de los salarios a través del método getTotalSalarios() de la clase Empresa.

9. Ejecutar Acciones con Base en los Empleados

```
empresa.empleadoMayorSalario();
empresa.empleadoMenorSalario();
```

empleadoMayorSalario(): Debería identificar el empleado con el salario más alto y mostrar su información.

empleadoMenorSalario(): Tambien, debe identificar el empleado con el salario más bajo.

Cod Programación Orientada a Objetos Clase Empleado

```java
public class Empleado {

    public String nombre;
    public String cargo;
    public double salario;


    public Empleado(String cargo, String nombre, double salario) {
        this.cargo = cargo;
        this.nombre = nombre;
        this.salario = salario;
    }
    public String getNombre() {
        return nombre;
    }

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

    public String getCargo() {
        return cargo;
    }

    public void setCargo(String cargo) {
        this.cargo = cargo;
    }

    public double getSalario() {
        return salario;
    }

    public void setSalario(double salario) {
        this.salario = salario;
    }

}
```

1. Definición de la Clase

public class Empleado {

La clase se llama Empleado. Esta clase es un modelo o plantilla para crear objetos que representen empleados en una aplicación.

2. Atributos de la Clase

public String nombre;

public String cargo;

public double salario;

Estos son los atributos o propiedades de la clase:

public String nombre: Almacena el nombre del empleado.

public String cargo: Almacena el cargo que el empleado tiene en la empresa.

public double salario: Almacena el salario del empleado.

3. Constructor

```
public Empleado(String cargo, String nombre, double salario) {
   this.cargo = cargo;
   this.nombre = nombre;
   this.salario = salario;
}
```

El constructor es un método especial que se ejecuta automáticamente cuando se crea una instancia de la clase Empleado.

Parámetros: Recibe cargo, nombre, y salario como entrada.

this: Se usa para referenciar los atributos actuales de la clase y evitar ambigüedades entre los

4. Getters and Setters

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

getNombre(): Devuelve el valor del atributo nombre.

setNombre(String nombre): Permite modificar el valor del atributo nombre.

Getter y Setter para cargo

```
public String getCargo() {
   return cargo;
}
public void setCargo(String cargo) {
   this.cargo = cargo;
}
```

getCargo(): Devuelve el valor del atributo cargo.

setCargo(String cargo): Permite modificar el valor del atributo cargo.

Getter y Setter para salario

```
public double getSalario() {
   return salario;
}
public void setSalario(double salario) {
   this.salario = salario;
}
```

getSalario(): Devuelve el valor del atributo salario.

setSalario(double salario): Permite modificar el valor del atributo salario.

Cod Programación Orientada a Objetos Empresa

```java
import java.util.ArrayList;

public class Empresa {

    public ArrayList<Empleado> empleados;

    public Empresa() {
        empleados = new ArrayList<Empleado>();
    }

    public void contratarEmpleado(Empleado e) {
        empleados.add(e);
    }

    public int getTotalEmpleados() {
        return empleados.size();
    }

    public void nombreSalario() {
        for (Empleado e : empleados) {
            System.out.println("Nombre: " + e.getNombre() + " Salario: " + e.getSalario());
        }
    }

    public double getTotalSalarios() {
        double total = 0;
        for (Empleado e : empleados)

        {
            total = total + e.getSalario();
        }
        return total;
    }

    public void empleadoMayorSalario() {
        Empleado empMayorSalario = empleados.get(0);
        double maxSalario = empleados.get(0).getSalario();
        for (Empleado e : empleados) {
            if (e.getSalario() > maxSalario) {
                maxSalario = e.getSalario();
                empMayorSalario = e;
            }
        }
        System.out.println(x:"El empleado que mas dinero gana es: ");
        System.out.print("Nombre: " + empMayorSalario.getNombre());
        System.out.print(" Cargo: " + empMayorSalario.getCargo());
        System.out.println(" Salario: " + empMayorSalario.getSalario());
    }

    public void empleadoMenorSalario() {
        Empleado empMenorSalario = empleados.get(0);
        double menSalario = empleados.get(0).getSalario();

        for (Empleado e : empleados) {
            if (e.getSalario() < menSalario) {
                menSalario = e.getSalario();
                empMenorSalario = e;
            }
        }
        System.out.println(x:"El empleado que menos dinero gana es: ");
        System.out.print("Nombre: " + empMenorSalario.getNombre());
        System.out.print(" Cargo: " + empMenorSalario.getCargo());
        System.out.println(" Salario: " + empMenorSalario.getSalario());
    }

}
```

Explicacion del codigo:

```
public class Empresa {
   public ArrayList<Empleado> empleados;
```

ArrayList<Empleado> empleados:

Una lista dinámica que almacena objetos de tipo Empleado.

Es una estructura de datos que permite agregar empleados dinámicamente.

public Empresa() Constructor:

Se inicializa la lista de empleados como una nueva instancia de ArrayList.

```
public Empresa() {
   empleados = new ArrayList<Empleado>();
}
```

Método para Contratar Empleados

```
public void contratarEmpleado(Empleado e) {
   empleados.add(e);
}
```

Recibe un objeto de tipo Empleado como parámetro.

Agrega el empleado a la lista de empleados usando el método add() de ArrayList.

Obtener el Total de Empleados

```
public int getTotalEmpleados() {
   return empleados.size();
}
```

Devuelve la cantidad de empleados actuales en la lista.

empleados.size() obtiene el tamaño de la lista.

Imprimir Nombres y Salarios

```
public void nombreSalario() {
   for (Empleado e : empleados) {
      System.out.println("Nombre: " + e.getNombre() + " Salario: " + e.getSalario());
   }
}
```

Para cada empleado en la lista, imprime su nombre y salario en consola utilizando los métodos getNombre() y getSalario() de la clase Empleado.

Calcular el Total de Salarios

```
public double getTotalSalarios() {
   double total = 0;
   for (Empleado e : empleados) {
      total = total + e.getSalario();
   }
   return total;
}
```

Variable total: Se inicializa en 0.

Bucle for: Recorre todos los empleados de la lista.

Suma el salario de cada empleado a la variable total.

Devuelve el total de todos los salarios.

Empleado con el Mayor Salario

```
public void empleadoMayorSalario() {
   Empleado empMayorSalario = empleados.get(0);
   double maxSalario = empleados.get(0).getSalario();
   for (Empleado e : empleados) {
      if (e.getSalario() > maxSalario) {
         maxSalario = e.getSalario();
         empMayorSalario = e;
      }
   }
   System.out.println("El empleado que mas dinero gana es: ");
   System.out.print("Nombre: " + empMayorSalario.getNombre());
   System.out.print(" Cargo: " + empMayorSalario.getCargo());
   System.out.println(" Salario: " + empMayorSalario.getSalario());
}
```

Se asume que el primer empleado es el que tiene el salario más alto inicialmente:

```
Empleado empMayorSalario = empleados.get(0);
```

```
double maxSalario = empleados.get(0).getSalario();
```

Si el salario de algún empleado es mayor que el valor almacenado en maxSalario, actualiza las referencias:

```
if (e.getSalario() > maxSalario) {
   maxSalario = e.getSalario();
   empMayorSalario = e;
}
```

Muestra el nombre, cargo y salario del empleado con el mayor salario.

Empleado con el Menor Salario

```
public void empleadoMenorSalario() {
   Empleado empMenorSalario = empleados.get(0);
   double menSalario = empleados.get(0).getSalario();
   for (Empleado e : empleados) {
      if (e.getSalario() < menSalario) {
         menSalario = e.getSalario();
         empMenorSalario = e;
      }
   }
   System.out.println("El empleado que menos dinero gana es: ");
   System.out.print("Nombre: " + empMenorSalario.getNombre());
   System.out.print(" Cargo: " + empMenorSalario.getCargo());
   System.out.println(" Salario: " + empMenorSalario.getSalario());
}
```

Se asume que el primer empleado es el que tiene el salario más bajo inicialmente:

```
Empleado empMenorSalario = empleados.get(0);
double menSalario = empleados.get(0).getSalario();
```

Bucle para comparar salarios:

Recorre la lista de empleados.

Si el salario de algún empleado es menor que el valor almacenado en menSalario, actualiza las referencias:

```
if (e.getSalario() < menSalario) {
```

```
    menSalario = e.getSalario();
    empMenorSalario = e;
}
```

Luego muestra el nombre, cargo y salario del empleado con el menor salario.

Cod Programación Orientada A Objetos Estudiante

```
import java.util.Scanner;



public class ejercicioEstudiantePOO2 {

    public static void main(String[] args) throws Exception {

        Estudiante est;


        String nombre;

        double nota1, nota2;


        Scanner name = new Scanner(System.in);

        Scanner not1 = new Scanner(System.in);

        Scanner not2 = new Scanner(System.in);


        for (int i = 1; i < 4; i++) {

        System.out.println("Ingrese el nombre del " + i + " Estudiante: ");

        nombre = name.next();

        System.out.println("Ingrese la nota 1 del parcial del " + i + "
Estudiante: ");

        nota1 = not1.nextDouble();

        System.out.println("Ingrese la nota 2 del parcial del " + i + "
Estudiante: ");

        nota2 = not2.nextDouble();
```

```
        est = new Estudiante(nombre);
        est.asignarNota1(nota1);
        est.asignarNota2(nota2);
        est.calcularNotaFinal();

        System.out.println("Querido estudiante: " + est.nombre + " Su nota 1 fue de: " + est.obtenerNota1() + " Su nota 2 fue de: " + est.obtenerNota2() + " El promedio de sus 2 notas parciales es de: " + est.obtenerNotaFinal() + " Usted" + est.obtenerMensaje());
        }
    }
}
```

Explicacion del codigo:

1. Declaraciones de Variables y Objetos

Estudiante est;

String nombre;

double nota1, nota2;

Estudiante est: Declaración de la variable que contendrá una instancia de la clase Estudiante.

String nombre: Para almacenar el nombre de cada estudiante ingresado por el usuario.

double nota1, nota2: Para capturar las dos notas de cada estudiante.

2. Creación de Escáneres para la Entrada del Usuario

Scanner name = new Scanner(System.in);

Scanner not1 = new Scanner(System.in);

Scanner not2 = new Scanner(System.in);

Aquí se crean tres objetos Scanner para leer la entrada del usuario:

name: Para capturar los nombres de los estudiantes.

not1: Para capturar la primera nota.

not2: Para capturar la segunda nota.

3. Bucle para Solicitar Información de Tres Estudiantes

```
for (int i = 1; i < 4; i++) {
```

El bucle recorre tres iteraciones (i = 1, 2, 3) para capturar información de 3 estudiantes.

4. Captura de Datos de Entrada

```
System.out.println("Ingrese el nombre del " + i + " Estudiante: ");
nombre = name.next();
System.out.println("Ingrese la nota 1 del parcial del " + i + " Estudiante: ");
nota1 = not1.nextDouble();
System.out.println("Ingrese la nota 2 del parcial del " + i + " Estudiante: ");
nota2 = not2.nextDouble();
```

Cada iteración del bucle solicita lo siguiente:

Nombre del estudiante: Se captura con name.next().

Primera nota parcial: Se captura con not1.nextDouble().

Segunda nota parcial: Se captura con not2.nextDouble().

5. Crear el Objeto Estudiante

```
est = new Estudiante(nombre);
```

Se crea una nueva instancia de la clase Estudiante con el nombre ingresado por el usuario.

6. Asignar las Notas

```
est.asignarNota1(nota1);
est.asignarNota2(nota2);
```

Se asignan las dos notas ingresadas por el usuario al objeto est mediante sus respectivos métodos:

asignarNota1(nota1): Asigna la primera nota.

asignarNota2(nota2): Asigna la segunda nota.

7. Calcular el Promedio

```
est.calcularNotaFinal();
```

Calcula el promedio de las dos notas para el estudiante.

8. Mostrar la Información Final

```
System.out.println("Querido estudiante: " + est.nombre +
                   " Su nota 1 fue de: " + est.obtenerNota1() +
```

```
" Su nota 2 fue de: " + est.obtenerNota2() +
" El promedio de sus 2 notas parciales es de: " +
est.obtenerNotaFinal() +
" Usted" + est.obtenerMensaje());
```

Cod Programación Orientada a Objetos Clase Estudiante

```
public class Estudiante {

	public String nombre;
	public double nota1;
	public double nota2;
	public double notaFinal;

	public Estudiante(String nombre) {
		this.nombre = nombre;
	}

	public Estudiante(String nombre, double np1, double np2) {
		this.nombre = nombre;
		nota1 = np1;
		nota2 = np2;
	}

	public void asignarNota1(double np1) {
		nota1 = np1;
	}
```

```
	public void asignarNota2(double np2) {
		nota2 = np2;
	}

	public double obtenerNota1() {
		return nota1;
	}

	public double obtenerNota2() {
		return nota2;
	}

	public void calcularNotaFinal() {
		notaFinal = (nota1 + nota2) / 2;
	}

	public double obtenerNotaFinal() {

		return notaFinal;
	}

	public String obtenerMensaje() {
		if (notaFinal < 3)
			return " ha reprobado lo lamento";
		else
			return " ha aprobado felicidades";
```

```
        }
}
```

Explicacion del codigo:

1. Declaración de la clase y atributos

```
public class Estudiante {


    public String nombre;

    public double nota1;

    public double nota2;

    public double notaFinal;
```

public String nombre;: Almacena el nombre del estudiante.

public double nota1;: Almacena la primera nota parcial del estudiante.

public double nota2;: Almacena la segunda nota parcial del estudiante.

public double notaFinal;: Almacena el promedio de las dos notas parciales.

Estas variables son públicas, lo que significa que se pueden acceder directamente desde fuera de la clase, aunque por buenas prácticas en la programación orientada a objetos es mejor usar variables privadas y proveer métodos getter y setter.

2. Constructores

```
public Estudiante(String nombre) {
  this.nombre = nombre;
}


public Estudiante(String nombre, double np1, double np2) {
  this.nombre = nombre;
  nota1 = np1;
  nota2 = np2;
}
```

Primer constructor:

Recibe solo el nombre del estudiante y lo asigna al atributo nombre.

```
public Estudiante(String nombre)
```

Segundo constructor:

Recibe el nombre y dos notas parciales y las asigna a sus atributos correspondientes. Esto permite crear un objeto Estudiante con las notas ya predefinidas.

```
public Estudiante(String nombre, double np1, double np2)
```

Los constructores permiten crear objetos de la clase Estudiante de diferentes formas según los datos disponibles en ese momento.

3. Métodos para asignar las notas

```
public void asignarNota1(double np1) {
   nota1 = np1;
}

public void asignarNota2(double np2) {
   nota2 = np2;
}
```

asignarNota1(double np1):

Asigna el valor np1 a la variable nota1.

asignarNota2(double np2):

Asigna el valor np2 a la variable nota2.

Estos métodos permiten establecer las notas del estudiante dinámicamente después de crear el objeto.

4. Obtener las notas individuales

```
public double obtenerNota1() {
   return nota1;
}

public double obtenerNota2() {
   return nota2;
}
```

obtenerNota1(): Devuelve el valor de la primera nota.

obtenerNota2(): Devuelve el valor de la segunda nota.

Estos métodos permiten acceder a las notas individuales de un estudiante desde fuera de la clase.

5. Calcular el promedio de las dos notas

```
public void calcularNotaFinal() {
   notaFinal = (nota1 + nota2) / 2;
}
```

Este método calcula el promedio de las dos notas parciales (nota1 y nota2) y lo almacena en el atributo notaFinal.

El cálculo se realiza como la suma de ambas notas dividido entre 2.

6. Obtener el promedio calculado

```
public double obtenerNotaFinal() {
   return notaFinal;
}
```

Devuelve el valor de la variable notaFinal, que contiene el promedio calculado.

7. Generar un mensaje según el promedio

```
public String obtenerMensaje() {
   if (notaFinal < 3)
      return " ha reprobado lo lamento";
   else
      return " ha aprobado felicidades";
}
```

Este método genera un mensaje personalizado basado en el promedio del estudiante:

Si el promedio es menor que 3, devuelve el mensaje:

" ha reprobado lo lamento".

Si el promedio es 3 o mayor, devuelve el mensaje:

" ha aprobado felicidades".

Conclusión

El análisis y desarrollo de los códigos presentados permiten observar las ventajas y diferencias entre los paradigmas de programación estructurada y orientada a objetos. Mientras que el enfoque estructurado resulta efectivo para soluciones rápidas y directas, la programación orientada a objetos (POO) proporciona una mejor organización, escalabilidad y reutilización del código, lo que facilita el mantenimiento y la expansión de los sistemas desarrollados.

En el caso de la gestión de empleados, la implementación con POO sobresale por su habilidad para manejar varios empleados de manera ordenada, simplificando operaciones como el cálculo de salarios totales y la identificación de empleados con los salarios más altos y más bajos. De manera similar, en el contexto de los estudiantes, la programación orientada a objetos permite encapsular la lógica para el cálculo de notas y la generación de mensajes personalizados, dejando al programa principal únicamente la tarea de controlar el flujo de la aplicación.

En general, estos programas demuestran claridad, eficiencia y flexibilidad, siguiendo buenas prácticas de programación y estableciendo una base robusta para resolver problemas más complejos en el futuro.