Explicación

click+source (atajo)

Parte teoría

Métodos

void-> este método realiza un proceso que no se muestra por pantalla

Constructor

Se puede crear a mano o con atajo
Puede haber varios con el mismo nombre siempre y cuando estos no tengan los mismos parámetros.
this.valor=valor; -> con esto asignamos el valor del parámetro **CLIENTE**

Crear alumnos, etc. MAIN

Alumno alu1= new Alumno();-> llamo al constructor pero lo dejo vacio
Alumno alu2= new Alumno(int id, string nombre,double nota);->doy valores

Getters y setter CLIENTE

También se puede hacer con atajos.
Se pone antes de los métodos y después de los constructores.
Son métodos que nos permiten coger los atributos de las clases, por cada atributo se nos crea un getter y un setter.
Get-> Traer, con él obtenemos los datos(vemos los datos)
Setter-> Colocar, establecemos valores (modificaciones de valores y colocarlos)

Mostrar un dato de class (con valor)

System.out.println("Id es:"+ alu2.getId());-> ctrl+espacio opciones autocompletar

Editar dato con setter

alu2.setId(35);-> el nuevo valor sobreescribe el viejo

Mostrar un dato sin valor

alu1.setiD(escribimos valor);-> después hacemos un system y nos saldrá el valor indicado

Mostrar un dato sin valor con scanner

También se puede pedir que lo introduzcas y lo lea para esto hacemos un scanner

```
System.out.println("Ingrese los datos del dependiente 2:");
System.out.print("Nombre: ");
String nombreDep = leerTeclado.nextLine();
```

Principal.java

Paquete y Clase

```
package repasoJava;
public class Principal {
```

(Línea 1-2): Indica que la clase Principal pertenece al paquete repasoJava.
(Línea 3): Declara una clase pública llamada Principal.

Método Principal

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

(Línea 7): Define el método principal main, que es el punto de entrada de la aplicación Java.

Variables y Objetos

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

(Línea 8): Crea un objeto Scanner llamado leerTeclado para leer datos de la entrada estándar (teclado).

Creación de Clientes

```
Cliente client1 = new Cliente();
Cliente client2 = new Cliente("angela", "vivanco", "28472348A", 12);
```

(Línea 11, 12): Crea instancias de la clase Cliente usando diferentes constructores.

Mostrar Datos del Cliente

```
System.out.println("El id del cliente dos es: " + client2.getId());
System.out.println("--------");
client1.setId(3);
System.out.println("El id del cliente uno es: " + client1.getId());
System.out.println("--------");
```

(Línea 15, 16, 19-21): Muestra y modifica los datos de los clientes.

Lectura y Validación de Precio

```
System.out.println("Precio:");
double precioCompra = leerTeclado.nextDouble();
```

(Línea 24-25): Solicita y lee un precio ingresado por el usuario.

```
if (precioCompra > 0) {
    System.out.println("Valido");
} else {
    System.out.println("Invalido");
}
leerTeclado.nextLine();
```

(Línea 28-32): Valida y muestra el precio ingresado.

Creación y Solicitud de Datos del Dependiente

```
Dependiente depd1 = new Dependiente("Juan", "Perez", 1, "Gerente");
```

(Línea 36): Crea una instancia de Dependiente con datos fijos.

```
System.out.println("Ingrese los datos del dependiente 2:");
System.out.print("Nombre: ");
String nombreDep = leerTeclado.nextLine();
```
(Línea 39-40): Solicita y lee los datos del segundo dependiente.

```
Dependiente depd2 = new Dependiente(nombreDep, apellidoDep, idDep, cargoDep);
```
(Línea 46): Crea una instancia de Dependiente con datos ingresados por el usuario.

Mostrar Datos del Dependiente

```
System.out.println("Datos del dependiente 1:");
System.out.println("Nombre: " + depd1.getNombre());
```
(Línea 51-54): Muestra los datos del primer dependiente.

```
System.out.println("Datos del dependiente 2:");
System.out.println("Nombre: " + depd2.getNombre());
```

Cliente.java

Paquete y Clase

```
package repasoJava;
public class Cliente {
```
(Línea 1-2): Indica que la clase Cliente pertenece al paquete repasoJava.
(Línea 3): Declara una clase pública llamada Cliente.

Atributos Privados

```
private String nombre;
private String apellido;
private String dni;
private int id;
```
(Línea 4-7): Define los atributos privados de la clase Cliente.

Constructores

```
public Cliente() {
}
```
(Línea 8): Constructor vacío de la clase Cliente.

```
public Cliente(String nombre, String apellido, String dni, int id) {
    this.nombre = nombre;
    this.apellido = apellido;
    this.dni = dni;
    this.id = id;
}
```
(Línea 9-12): Constructor con parámetros que inicializa los atributos de la clase.

Getters y Setters

```
public String getNombre() {
```

```
    return nombre;
}
```
(Línea 13-15): Método getter para obtener el nombre del cliente.

```
public void setNombre(String nombre) {
    this.nombre = nombre;
}
```
(Línea 16-18): Método setter para establecer el nombre del cliente.

```
public int getId() {
    return id;
}
```
(Línea 19-21): Método getter para obtener el ID del cliente.

```
public void setId(int id) {
    this.id = id;
}
```
(Línea 22-24): Método setter para establecer el ID del cliente.

Métodos Adicionales

```
public void mostrarDatos() {
    System.out.println("Nombre: " + nombre);
    System.out.println("Apellido: " + apellido);
    System.out.println("DNI: " + dni);
    System.out.println("ID: " + id);
}
```
(Línea 25-32): Método para mostrar los datos del cliente en la consola.

Dependiente.java

Paquete y Clase

```
package repasoJava;
public class Dependiente {
```
(Línea 1-2): Indica que la clase Dependiente pertenece al paquete repasoJava.
(Línea 3): Declara una clase pública llamada Dependiente.

Atributos Privados

```
private String nombre;
private String apellido;
private int id;
private String cargo;
```
(Línea 4-7): Define los atributos privados de la clase Dependiente.

Constructores

```
public Dependiente(String nombre, String apellido, int id, String cargo) {
    this.nombre = nombre;
```

```
    this.apellido = apellido;
    this.id = id;
    this.cargo = cargo;
}
```
(Línea 8-11): Constructor con parámetros que inicializa los atributos de la clase.

Getters y Setters

```
public String getNombre() {
    return nombre;
}
```
(Línea 12-14): Método getter para obtener el nombre del dependiente.

```
public void setNombre(String nombre) {
    this.nombre = nombre;
}
```
(Línea 15-17): Método setter para establecer el nombre del dependiente.

```
public int getId() {
    return id;
}
```
(Línea 18-20): Método getter para obtener el ID del dependiente.

```
public void setId(int id) {
    this.id = id;
}
```
(Línea 21-23): Método setter para establecer el ID del dependiente.

```
public String getCargo() {
    return cargo;
}
```
(Línea 24-26): Método getter para obtener el cargo del dependiente.

```
public void setCargo(String cargo) {
    this.cargo = cargo;
}
```
(Línea 27-29): Método setter para establecer el cargo del dependiente.

Métodos Adicionales

```
public void mostrarDatos() {
    System.out.println("Nombre: " + nombre);
```

```
    System.out.println("Apellido: " + apellido);
    System.out.println("ID: " + id);
    System.out.println("Cargo: " + cargo);
}
```
(Línea 30-37): Método para mostrar los datos del dependiente en la consola.

Herencia
En el padre creamos las clases con su constructor u los getter y setter.Esto se hace con los valores que tienen en común los códigos hijos.

herencia.java
Paquete y Clase
```
package repasoHerencia;
public class herencia {
```
(Línea 1-2): Indica que la clase herencia pertenece al paquete repasoHerencia.
(Línea 3): Declara una clase pública llamada herencia.

Método Principal
```
public static void main(String[] args) {
```
(Línea 5-7): Define el método principal main, que es el punto de entrada de la aplicación Java.

Creación de Objetos
```
cliente client = new cliente(null, null, null, 0, 0, null, null, 0);
```
(Línea 9): Crea una instancia de la clase cliente utilizando el constructor con parámetros.

```
administador admin = new administador(null, null, null, 0, 0, null, null);
```
(Línea 12): Crea una instancia de la clase administador utilizando el constructor con parámetros.

Acceso a Métodos
```
client.getNombre();
client.getEdad();
```
(Línea 14-15): Accede a los métodos getNombre() y getEdad() de la instancia de cliente.

```
admin.getApellido();
admin.getCargo();
```
(Línea 17-18): Accede a los métodos getApellido() y getCargo() de la instancia de administador.

Creación de una Lista y Uso de foreach
```
List<cliente> lista = new ArrayList<cliente>();
```
(Línea 20): Crea una lista de objetos de tipo cliente utilizando la interfaz List y la clase

ArrayList.
```
lista.add(new cliente("Angela", "Vivanco", "asds", 2, 3, "sdsd", "sds", 20));
lista.add(new cliente("Paqui", "Diaz", "dfdfs", 3, 3, "sdsd", "sds", 30));
```

```
lista.add(new cliente("Fer", "Vivanco", "frf", 2, 2, "sdsd", "sds", 16));
```
(Línea 22-24): Agrega objetos al cliente a la lista.

Iteración sobre la Lista con un for indice

```
for (int i = 0; i < lista.size(); i++) {
    System.out.println("Prueba indice:" + lista.get(i).getEdad());
}
```
(Línea 26-29): Itera sobre la lista utilizando un bucle for y muestra la edad de cada cliente.

Iteración sobre la Lista con foreach

```
for (cliente cliente : lista) {
    System.out.println("Prueba foreach:" + cliente.getEdad());
}
```
(Línea 31-34): Itera sobre la lista utilizando un bucle foreach y muestra la edad de cada cliente.

Dar de baja y editar

// Método para dar de baja a un cliente

```
    private static void bajaClin(Scanner leerTeclado) {
```

1-Pedimos buscarlo por un atributo Ej.ID

```
        System.out.println("Introduce el ID del cliente a eliminar:");
        int id = leerTeclado.nextInt();
        Cliente clienteAEliminar = null;
```

2-Arraylist sobrecliente

```
        for (Cliente cliente : personas) {
            if (cliente.getId() == id) {
                clienteAEliminar = cliente;
                break;
            }
        }
```

2-Indicamos que nos borre la persona deseada con personas.remove

```
        if (clienteAEliminar != null) {
            personas.remove(clienteAEliminar);
            System.out.println("Cliente eliminado correctamente.");
        } else {
            System.out.println("Cliente no encontrado.");
        }
    }
```

// Método para editar un cliente

1-Pedimos matrícula

```
 System.out.println("Introduce la matrícula del coche a editar:");
        int matricula = leerTeclado.nextInt();
        leerTeclado.nextLine();
```

2-Hacemos arraylist sobre coche y que nos seleccione la matrícula

```
    Coche cocheAEditar = null;
        for (Coche coche : vehiculos) {
            if (coche.getMatricula() == matricula) {
                cocheAEditar = coche;
                break;
             } }
```

3-Hacemos if else para pedir los nuevos datos

```
        if (cocheAEditar != null) {
                    System.out.println("Introduce la nueva marca:");
            cocheAEditar.setMarca(leerTeclado.nextLine());
                    System.out.println("Introduce el nuevo año:");
            cocheAEditar.setAño(leerTeclado.nextInt());
            leerTeclado.nextLine(); // Limpiar el buffer
            System.out.println("Coche actualizado correctamente.");
        } else {
            System.out.println("Coche no encontrado.");}  }
```

Mostrar usuario

// Método para mostrar la lista de usuarios (personas) **ArrayList**

```
    private static void mostrarUsuarios() {
        for (Cliente cliente : personas) {
            System.out.println(cliente);
        }
    }
```

Crear alta

```
private static void crearAlta() {
            System.out.println("Marca coche:");        |
        String marca = leerTeclado.nextLine();        |
                                                      | -> pedimos datos y los leemos
         int año = leerTeclado.nextInt();              |
         leerTeclado.nextLine(); // Limpiar el buffer  |
```

Añadimos un nuevo Coche (con sus atributos)

```
         vehiculos.add(new Coche(marca, modelo, combustible, color, carnet, año, matricula));
         System.out.println("\nAlta guardada");
    }
    private static void salir() {
        System.out.println("Saliendo...");
    }
```

Creamos variable coche, con su constructor (para poder guardar todo lo que hemos creado)

```
        static class Coche {
        private String marca, modelo, combustible, color, tipoCarnet;
        private int año, matricula;
```

Constructor

```
        public Coche(String marca, String modelo, String combustible, String color, String
tipoCarnet, int año,
                int matricula) {

            this.marca = marca;
            this.modelo = modelo;
            this.combustible = combustible;
            this.color = color;
            this.tipoCarnet = tipoCarnet;
            this.año = año;
            this.matricula = matricula;
        }
```

Clase Menu

Esta clase es la principal y contiene el menú de opciones para gestionar los objetos de tipo Cliente.

Atributos estáticos

ArrayList<Cliente> personas: Lista para almacenar objetos de tipo Cliente.
int opcion: Variable para almacenar la opción del menú seleccionada por el usuario.
Scanner leerTeclado: Objeto para leer la entrada del usuario desde el teclado.

Método main

Bucle Principal: Se ejecuta mientras la variable salir sea false. Muestra el menú inicial y espera la opción del usuario para llamar al método correspondiente
Método menuInicial: Muestra el menú principal y lee la opción del usuario.
Métodos del Menú Principal
gestCliente(Scanner leerTeclado): Maneja las opciones relacionadas con los clientes.
gestAdministrador(Scanner leerTeclado): Método reservado para gestionar administradores (sin implementar).
gestJefe(Scanner leerTeclado): Método reservado para gestionar jefes (sin implementar).
mostrarUsuarios(): Muestra la lista de todos los clientes almacenados en personas.
salir(): Imprime un mensaje de despedida y termina el programa.

Métodos del Submenú de Clientes

gestCliente(Scanner leerTeclado): Muestra las opciones relacionadas con los **clientes (altas, editar, bajas, mostrar).** Utiliza un bucle do-while para manejar estas opciones.

Métodos para Gestionar Clientes

crearAlta(Scanner leerTeclado):
Solicita al usuario si desea crear un nuevo cliente o volver al menú anterior.
Llama al método **datosNuevoCliente** para obtener los datos del nuevo cliente y agregarlo a la lista personas.
datosNuevoCliente(Scanner leerTeclado):
Solicita los datos del nuevo **cliente (nombre, apellido, DNI, teléfono, ID, ciudad, país y edad).**
Crea un nuevo objeto Cliente con estos datos y lo añade a la lista personas.
bajaClin(Scanner leerTeclado):

Solicita al usuario el ID del cliente a eliminar.
Busca el cliente en la lista personas por ID y lo elimina si lo encuentra.
editarClin(Scanner leerTeclado):

Solicita al usuario el ID del cliente a editar.
Busca el cliente en la lista personas por ID.
Si encuentra el cliente, solicita los nuevos datos y actualiza el objeto Cliente correspondiente.

Clase Cliente
Esta clase define los objetos que se almacenarán en la lista personas. Cada cliente tiene los siguientes atributos:

String nombre
String apellido
String dni
int telefono
int id
String ciudad
String pais
int edad

Constructores y Métodos
Constructor: Inicializa todos los atributos del cliente.
Getters y Setters: Métodos para obtener y establecer los valores de los atributos.
toString(): Devuelve una representación en cadena del objeto Cliente.

Ejecución del Programa
Inicio: El programa empieza mostrando el menú principal.
Selección de Opción: El usuario selecciona una opción (gestionar clientes, administradores, jefes, mostrar usuarios o salir).
Submenú de Clientes: Si se selecciona la opción de gestionar clientes, se muestra un submenú con opciones para añadir, editar, eliminar o mostrar clientes.
Operaciones de Cliente: Dependiendo de la opción seleccionada en el submenú de clientes, se ejecuta la operación correspondiente.
Salida: Si se selecciona la opción de salir, el programa imprime un mensaje de despedida y termina.