Ministerio de Producción
Presidencia de la Nación

Ministerio de Educación y Deportes

Subsecretaría de Servicios Tecnológicos y Productivos

Programa 111 mil
VOS PODÉS SER UNO.

CONSOLIDACIÓN DE TEMAS

Programación

- Enunciado.
- Diseño de la solución.
- Atributos de las clases.
- ArrayList vs. Arreglo.
- Constructores.
- Getters y Setters.
- Comportamiento.

Enunciado

La empresa “Soft en Java” desea implementar un sistema para registrar sus clientes y facturas. En particular desea 4 funcionalidades:

- Conocer nombre, apellido y dirección de sus clientes.
- Saber cuanto se le facturó a cada cliente.
- Saber a que cliente se le facturó más.

Enunciado a diseño

La **empresa** “Soft en Java” desea implementar un sistema para registrar sus **clientes** y **facturas**. En particular desea 4 funcionalidades:

- Conocer nombre, apellido y dirección de sus clientes.
- Saber cuanto se le facturó a cada cliente.
- Saber a que cliente se le facturó más.

Enunciado a diseño

Cliente:

Factura:

Empresa:

Enunciado a diseño

Cliente:

- Nombre: String
- Apellido: String
- Dirección: String

Factura:

Empresa:

Enunciado a diseño

Cliente:

- Nombre: String
- Apellido: String
- Dirección: String

Factura:

- Cliente: Cliente
- Monto: int

Empresa:

Enunciado a diseño

Cliente:

- Nombre: String
- Apellido: String
- Dirección: String

Factura:

- Cliente: Cliente
- Monto: int

Empresa:

- Clientes: Cliente[]
- Facturas: Factura[]

Enunciado a diseño

Cliente:

- Nombre: String
- Apellido: String
- Dirección: String

Factura:

- Cliente: Cliente
- Monto: int

Empresa:

- Clientes: ¿Cliente[]?
- Facturas: ¿Factura[]?

Enunciado a diseño

Cliente:

- Nombre: String
- Apellido: String
- Dirección: String

Factura:

- Cliente: Cliente
- Monto: int

Empresa:

- Clientes: ArrayList<Cliente>
- Facturas: ArrayList<Factura>

ArrayList vs. Arreglo

Java.util.ArrayList es una clase provista por Java que provee una funcionalidad similar a los arreglos, salvo que permite agrega un nuevo elemento al final cambiando su tamaño cuando es requerido.

ArrayList vs. Arreglo: Equivalencias

Operación en arreglo	Operación en ArrayList	Notas
Cliente[] c = new Cliente[10];	ArrayList<Cliente> c = new ArrayList<>();	Creación. En el caso del ArrayList el tamaño inicial del arraylist es 0.
c[2] = cliente;	c.set(2, cliente);	Cambia el valor en la posición dada.
cliente = c[4];	cliente = c.get(4);	Obtiene el valor en la posición dada.
int l = c.length;	int l = c.size();	Obtiene el tamaño de la estructura.
¡NO SE PUEDE HACER!	c.add(cliente);	Agrega un cliente al final de la estructura. Cambia el tamaño de la misma.

Funcionalidad

Cliente:

Factura:

Empresa:

Funcionalidad

Cliente:

- Getters y Setters.

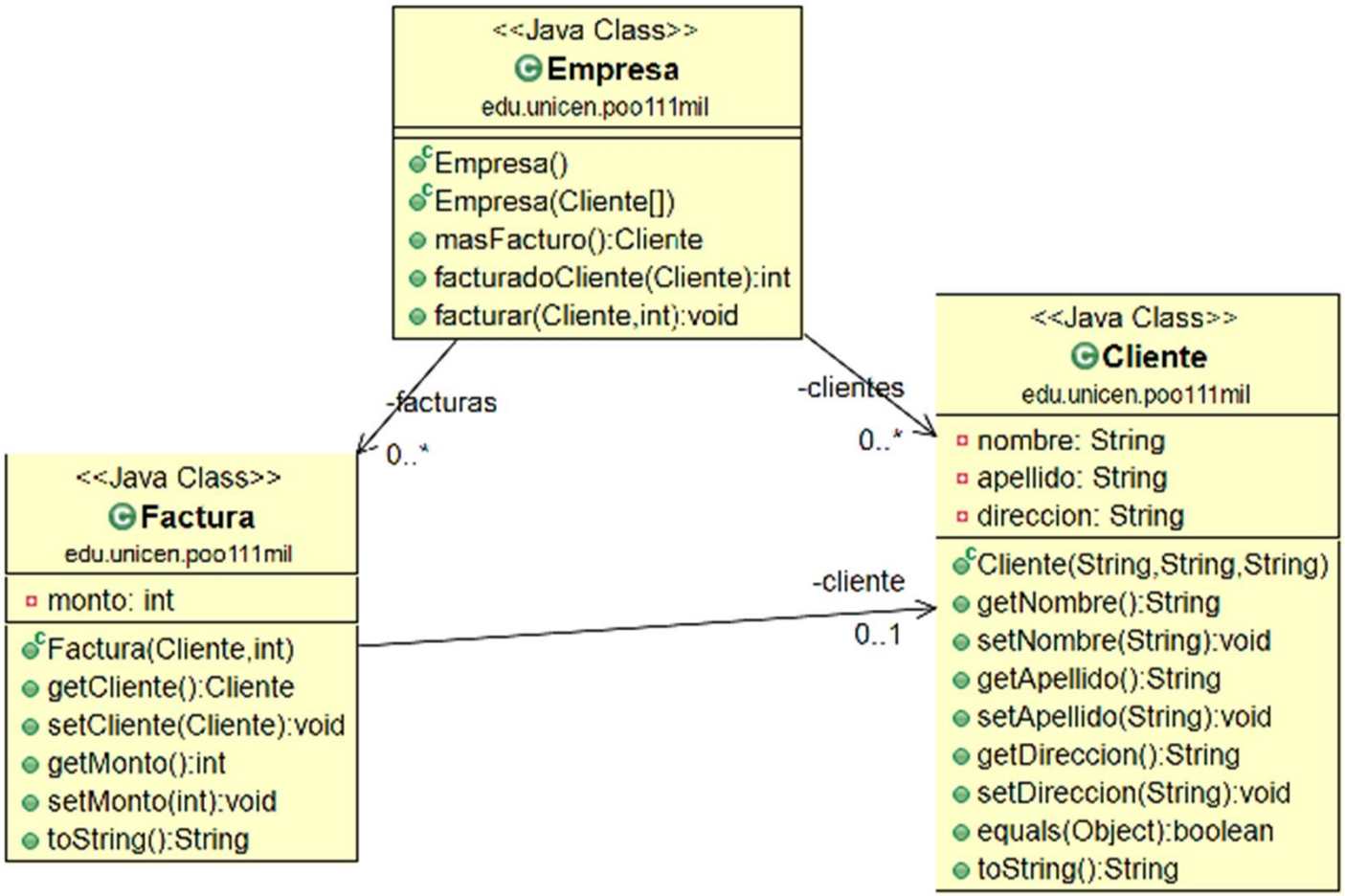
Factura:

- Getters y Setters.

Empresa:

- Getters y Setters.
- masFacturo(): Cliente
- facturadoCliente(Cliente): int

Diagrama de clases



Código -- ¡Al fin!

Antes de comenzar a implementar vamos a definir un main para probar nuestra implementación.

- Crear 10 clientes.
- Crear una empresa con esos clientes.
- Generar 100 facturas aleatorias.
- Obtener lo facturado por cliente, e imprimirlo por pantalla.
- Obtener el cliente al que más se le facturó, imprimir los datos del cliente por pantalla e imprimir el monto por pantalla.

Código -- ¡Al fin! (esta vez de verdad)

- Crear 10 clientes.

```
public static void main(String[] args) {
   Cliente[] clientes = new Cliente[10];
   for(int i = 0; i<10; i++)
      clientes[i] = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
   ...
}
```

Código -- ¡Al fin! (esta vez de verdad)

- Crear una empresa con esos clientes.

```
public static void main(String[] args) {
   Cliente[] clientes = new Cliente[10];
   for(int i = 0; i<10; i++)
      clientes[i] = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
   Empresa e = new Empresa(clientes);
   ...
}
```

Código -- ¡Al fin! (esta vez de verdad)

- Generar 100 facturas aleatorias.

```
public static void main(String[] args) {
   Cliente[] clientes = new Cliente[10];
   for(int i = 0; i<10; i++)
      clientes[i] = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
   Empresa e = new Empresa(clientes);
   for(int i = 0; i < 100; i++) {
      Cliente c = clientes[(int)(clientes.length * Math.random())];
      int monto = (int)(1000 * Math.random());
      e.facturar(c, monto);
   }
   ...
}
```

Código -- ¡Al fin! (esta vez de verdad)

- Obtener facturado por cliente...

```
public static void main(String[] args) {
   Cliente[] clientes = new Cliente[10];
   for(int i = 0; i<10; i++)
      clientes[i] = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
   Empresa e = new Empresa(clientes);
   for(int i = 0; i < 100; i++) {
      Cliente c = clientes[(int)(clientes.length * Math.random())];
      int monto = (int)(1000 * Math.random());
      e.facturar(c, monto);
   }
   for(int i = 0; i<10; i++) {
      Cliente c = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
      System.out.println(c + " - " + e.facturadoCliente(c));
   }
   ...
}
```

Código -- ¡Al fin! (esta vez de verdad)

- Obtener el cliente al que más se le facturó...

```
public static void main(String[] args) {
   Cliente[] clientes = new Cliente[10];
   for(int i = 0; i<10; i++)
      clientes[i] = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
   Empresa e = new Empresa(clientes);
   for(int i = 0; i < 100; i++) {
      Cliente c = clientes[(int)(clientes.length * Math.random())];
      int monto = (int)(1000 * Math.random());
      e.facturar(c, monto);
   }
   for(int i = 0; i<10; i++) {
      Cliente c = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
      System.out.println(c + " - " + e.facturadoCliente(c));
   }
   Cliente mejor = e.masFacturo();
   System.out.println("El mejor cliente es "+ mejor);
   System.out.println("Facturo: " + e.facturadoCliente(mejor));
}
```

Código -- Cliente

```
public class Cliente {
    private String nombre;
    private String apellido;
    private String direccion;
    public Cliente(String nombre, String apellido, String direccion) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.direccion = direccion;
    }

    public String getNombre() {
        return nombre;
    }

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

Código -- Factura

```
public class Factura {
    private Cliente cliente;
    private int monto;
    public Factura(Cliente cliente, int monto) {
        this.cliente = cliente;
        this.monto = monto;
    }

    public Cliente getCliente() {
        return cliente;
    }

    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
    }

    public int getMonto() {
        return monto;
    }

    public void setMonto(int monto) {
        this.monto = monto;
    }
}
```

Código – Empresa (1)

```
import java.util.ArrayList;

public class Empresa {
   private ArrayList<Cliente> clientes;
   private ArrayList<Factura> facturas;
   public Empresa() {
      this.clientes = new ArrayList<>();
      this.facturas = new ArrayList<>();
   }
   public Empresa(Cliente[] clientes){
      this();
      for(Cliente c: clientes)
         this.clientes.add(c);
   }

   public void facturar(Cliente c, int monto) {
      Factura f = new Factura(c, monto);
      this.facturas.add(f);
   }
   ...
}
```

Código – Empresa (2)

```
import java.util.ArrayList;

public class Empresa {
   private ArrayList<Cliente> clientes;
   private ArrayList<Factura> facturas;
   public Empresa() {
      this.clientes = new ArrayList<>();
      this.facturas = new ArrayList<>();
   }
   public Empresa(Cliente[] clientes){
      this();
      for(Cliente c: clientes)
         this.clientes.add(c);
   }

   public void facturar(Cliente c, int monto) {
      Factura f = new Factura(c, monto);
      this.facturas.add(f);
   }
   ...
}
```

Este constructor crea los ArrayList vacios

Este constructor llama al primer constructor a través de this y luego rellena el arreglo con los clientes.

Se recorre el arreglo con un for each

Código – Empresa (3)

```
public int facturadoCliente(Cliente c) {
    int total = 0;
    for(Factura f: facturas)
        if (f.getCliente().equals(c))
            total += f.getMonto();
    return total;
}
```

Código – Empresa (4)

```
public Cliente masFacturo() {
    int[] facturado = new int[clientes.size()];
    for(Factura f: facturas) {
        int index = clientes.indexOf(f.getCliente());
        facturado[index] += f.getMonto();
    }
    int max = facturado[0];
    Cliente mejor = clientes.get(0);
    for (int i = 1; i < clientes.size(); i++)
        if(max < facturado[i]) {
            max = facturado[i];
            mejor = clientes.get(i);
        }
    return mejor;
}
```

Código – Recordando el main

```
public static void main(String[] args) {
   Cliente[] clientes = new Cliente[10];
   for(int i = 0; i<10; i++)
      clientes[i] = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
   Empresa e = new Empresa(clientes);
   for(int i = 0; i < 100; i++) {
      Cliente c = clientes[(int)(clientes.length * Math.random())];
      int monto = (int)(1000 * Math.random());
      e.facturar(c, monto);
   }
   for(int i = 0; i<10; i++) {
      Cliente c = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
      System.out.println(c + " - " + e.facturadoCliente(c));
   }
   Cliente mejor = e.masFacturo();
   System.out.println("El mejor cliente es "+ mejor);
   System.out.println("Facturado: " + e.facturadoCliente(mejor));
}
```

Código – Ejecutar el main

```
edu.unicen.poo111mil.Cliente@6d06d69c - 0
edu.unicen.poo111mil.Cliente@7852e922 - 0
edu.unicen.poo111mil.Cliente@4e25154f - 0
edu.unicen.poo111mil.Cliente@70dea4e - 0
edu.unicen.poo111mil.Cliente@5c647e05 - 0
edu.unicen.poo111mil.Cliente@33909752 - 0
edu.unicen.poo111mil.Cliente@55f96302 - 0
edu.unicen.poo111mil.Cliente@3d4eac69 - 0
edu.unicen.poo111mil.Cliente@42a57993 - 0
edu.unicen.poo111mil.Cliente@75b84c92 - 0
El mejor cliente es
edu.unicen.poo111mil.Cliente@6bc7c054
Facturado: 7803
```

Código -- toString

```
@Override
public String toString() {
   return "Cliente [nombre=" + nombre +
      ", apellido=" + apellido +
      ", direccion=" + direccion + "]";
}
```

Código – Ejecutar el main

```
Cliente [nombre=Nombre: 0, apellido=Apellido: 0, direccion=Calle 0] - 0
Cliente [nombre=Nombre: 1, apellido=Apellido: 1, direccion=Calle 1] - 0
Cliente [nombre=Nombre: 2, apellido=Apellido: 2, direccion=Calle 2] - 0
Cliente [nombre=Nombre: 3, apellido=Apellido: 3, direccion=Calle 3] - 0
Cliente [nombre=Nombre: 4, apellido=Apellido: 4, direccion=Calle 4] - 0
Cliente [nombre=Nombre: 5, apellido=Apellido: 5, direccion=Calle 5] - 0
Cliente [nombre=Nombre: 6, apellido=Apellido: 6, direccion=Calle 6] - 0
Cliente [nombre=Nombre: 7, apellido=Apellido: 7, direccion=Calle 7] - 0
Cliente [nombre=Nombre: 8, apellido=Apellido: 8, direccion=Calle 8] - 0
Cliente [nombre=Nombre: 9, apellido=Apellido: 9, direccion=Calle 9] - 0
El mejor cliente es Cliente [nombre=Nombre: 4, apellido=Apellido: 4, direccion=Calle 4]
Facturado: 9722
```

Código – Analizando el main

```
for(int i = 0; i<10; i++) {
   Cliente c = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
   System.out.println(c + " - " + e.facturadoCliente(c));
}
```

Creo un nuevo cliente, con los datos de clientes existentes y luego pido cuanto se le facturó ese cliente.

```
Cliente mejor = e.masFacturo();
System.out.println("El mejor cliente es "+ mejor);
System.out.println("Facturado: " + e.facturadoCliente(mejor));
```

Obtengo el cliente al que más se le facturó y luego pregunto cuanto se le facturó.

Código – Analizando el main

```
for(int i = 0; i<10; i++) {
   Cliente c = new Cliente("Nombre: " + i, "Apellido: " + i, "Calle "+i);
   System.out.println(c + " - " + e.facturadoCliente(c));
}
```

Creo un nuevo cliente, con los datos de clientes existentes y luego pido cuanto se le facturó ese cliente.

```
Cliente mejor = e.masFacturo();
System.out.println("El mejor cliente es "+ mejor);
System.out.println("Facturado: " + e.facturadoCliente(mejor));
```

Obtengo el cliente al que más se le facturó y luego pregunto cuanto se le facturó.

¡Hechos!

- En el primer caso, las instancias creadas NO son las mismas que las que tienen la empresa aunque semánticamente sean iguales.
- En el segundo caso, obtengo una instancia que tiene la empresa y luego verifico cuanto se le facturó.

Código – Analizando el main

¡Hechos!

- En el primer caso, las instancias creadas NO son las mismas que las que tienen la empresa aunque semánticamente sean iguales.
- En el segundo caso, obtengo una instancia que tiene la empresa y luego verifico cuanto se le facturó.
- ¿El problema está en facturadoCliente?

```
public int facturadoCliente(Cliente c) {
   int total = 0;
   for(Factura f: facturas)
      if (f.getCliente().equals(c))
         total += f.getMonto();
   return total;
}
```

Código – Analizando el main

¡Hechos!

- En el primer caso, las instancias creadas NO son las mismas que las que tienen la empresa aunque semánticamente sean iguales.
- En el segundo caso, obtengo una instancia que tiene la empresa y luego verifico cuanto se le facturó.
- ¿El problema está en facturadoCliente? NO, pero utiliza el método equals de Cliente
- ¿Definimos el método en cliente?

```
public int facturadoCliente(Cliente c) {
   int total = 0;
   for(Factura f: facturas)
      if (f.getCliente().equals(c))
         total += f.getMonto();
   return total;
}
```

Código – Analizando el main

¡Hechos!

- En el primer caso, las instancias creadas NO son las mismas que las que tienen la empresa aunque semánticamente sean iguales.
- En el segundo caso, obtengo una instancia que tiene la empresa y luego verifico cuanto se le facturó.
- ¿El problema está en facturadoCliente? NO, pero utiliza el método equals de Cliente
- ¿Definimos el método en cliente? NO, por lo que está usando la implementación por defecto equivalente a "=="

```
public int facturadoCliente(Cliente c) {
   int total = 0;
   for(Factura f: facturas)
      if (f.getCliente().equals(c))
         total += f.getMonto();
   return total;
}
```

Código – Analizando el main

¡Hechos!

- ¿Definimos el método en cliente? NO, por lo que está usando la implementación por defecto equivalente a "==".
- Eso explica porqué funciona con la misma instancia y no con instancias distintas. Probemos definir el método equals en cliente.

```
public int facturadoCliente(Cliente c) {
    int total = 0;
    for(Factura f: facturas)
        if (f.getCliente().equals(c))
            total += f.getMonto();
    return total;
}
```

Código – Definiendo equals en cliente

```
@Override
public boolean equals(Object obj) {
   if (this == obj)
      return true;
   Cliente other = (Cliente) obj;
   if (apellido == null) {
      if (other.apellido != null)
         return false;
   } else if (!apellido.equals(other.apellido))
      return false;
   if (direccion == null) {
      if (other.direccion != null)
         return false;
   } else if (!direccion.equals(other.direccion))
      return false;
   if (nombre == null) {
      if (other.nombre != null)
         return false;
   } else if (!nombre.equals(other.nombre))
      return false;
   return true;
}
```

Código – Ejecutar el main, otra vez

```
Cliente [nombre=Nombre: 0, apellido=Apellido: 0, direccion=Calle 0] - 5274
Cliente [nombre=Nombre: 1, apellido=Apellido: 1, direccion=Calle 1] - 5171
Cliente [nombre=Nombre: 2, apellido=Apellido: 2, direccion=Calle 2] - 4066
Cliente [nombre=Nombre: 3, apellido=Apellido: 3, direccion=Calle 3] - 5040
Cliente [nombre=Nombre: 4, apellido=Apellido: 4, direccion=Calle 4] - 5246
Cliente [nombre=Nombre: 5, apellido=Apellido: 5, direccion=Calle 5] - 5668
Cliente [nombre=Nombre: 6, apellido=Apellido: 6, direccion=Calle 6] - 6153
Cliente [nombre=Nombre: 7, apellido=Apellido: 7, direccion=Calle 7] - 5508
Cliente [nombre=Nombre: 8, apellido=Apellido: 8, direccion=Calle 8] - 7400
Cliente [nombre=Nombre: 9, apellido=Apellido: 9, direccion=Calle 9] - 2809
El mejor cliente es Cliente [nombre=Nombre: 8, apellido=Apellido: 8, direccion=Calle 8]
Facturado: 7400
```

Código – Ejecutar el main, otra vez

Cliente [nombre=Nombre: 0, apellido=Apellido: 0, direccion=Calle 0] - 5274
Cliente [nombre=Nombre: 1, apellido=Apellido: 1, direccion=Calle 1] - 5171
Cliente [nombre=Nombre: 2, apellido=Apellido: 2, direccion=Calle 2] - 4066
Cliente [nombre=Nombre: 3, apellido=Apellido: 3, direccion=Calle 3] - 5040
Cliente [nombre=Nombre: 4, apellido=Apellido: 4, direccion=Calle 4] - 5246
Cliente [nombre=Nombre: 5, apellido=Apellido: 5, direccion=Calle 5] - 5668
Cliente [nombre=Nombre: 6, apellido=Apellido: 6, direccion=Calle 6] - 6153
Cliente [nombre=Nombre: 7, apellido=Apellido: 7, direccion=Calle 7] - 5508
Cliente [nombre=Nombre: 8, apellido=Apellido: 8, direccion=Calle 8] - 7400
Cliente [nombre=Nombre: 9, apellido=Apellido: 9, direccion=Calle 9] - 2809
El mejor cliente es Cliente [nombre=Nombre: 8, apellido=Apellido: 8, direccion=Calle 8]
Facturado: 7400

Recapitualando

- Definimos una solución orientada a objetos para nuestro problema.
- Vimos la ventaja de ArrayList sobre arreglo.
- Definimos y usamos constructores, getters y setters.
- Implementamos la solución.
- Vimos el problema no redefinir toString.
- Vimos el problema de no redefinir el equals cuando trabajamos con multiples instancias.

Más actividades

- Definir el método masfacturo a partir del método facturoCliente.
- Reimplementar la clase Empresa, llevando un ArrayList de Integer que controle en tiempo de facturación cuando se le facturó al cliente.