

1. Human Resources

Humam Resources es una aplicación destinada a la administración de Empleados en una Empresa, administrando sus empleos, departamentos, gerentes, directores, salarios de cada empleo el cual se le asigna a los empleados, en si la Lógica de Negocio para la administración de Recursos Humanos.

Desarrollo de la aplicación en clase

Diapositiva 6 (Class Attributes):

1. Se abre el proyecto “HumanResources” desarrollado en el Modulo anterior.
2. Se elimina la clase `TestEmployee` ubicada en el paquete test.
3. Se copia al paquete test la clase `TestEmployee` ubicada en “Resources/Modulo 7/TestEmployee.java”
4. Se agrega el atributo `counter` a la clase `Employee` como muestra el diagrama UML de la imagen 1 y se inicializa a 0 como se muestra en el *Código 1*:

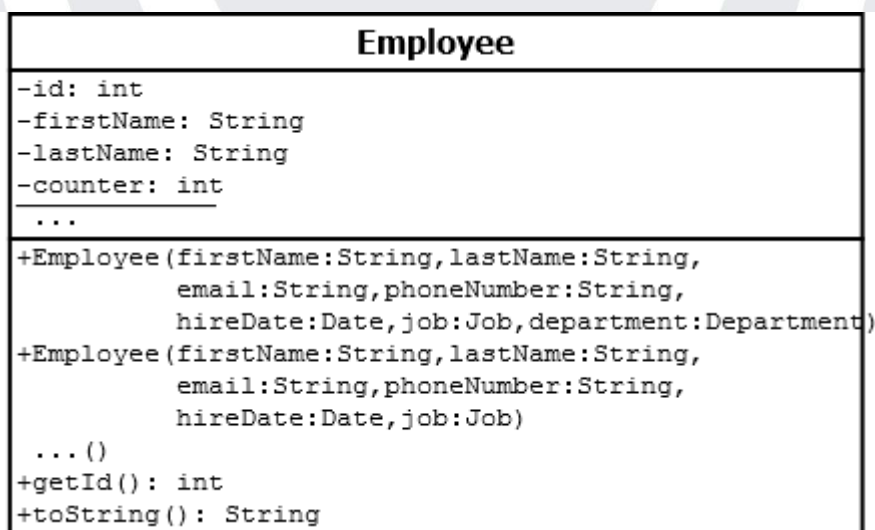


Imagen 1

```
15 private static int counter = 0;
```

Código 1

5. Se modifican los constructores de la clase `Employee` como se muestra en el *Código 2*:

```

30 public Employee(String firstName, String lastName, String email,
String
phoneNumber, Date hireDate, Job job, Department department) {
31     this(firstName, lastName, email, phoneNumber, hireDate, job);
32     this.department = department;
34 }
35
36 public Employee(String firstName, String lastName, String email,
String
phoneNumber, Date hireDate, Job job) {
37     this.firstName = firstName;
38     this.lastName = lastName;
39     this.email = email;
40     this.phoneNumber = phoneNumber;
41     this.hireDate = hireDate;
42     this.job = job;
43     this.id = counter++;
44 }

```

Código 2

Diapositiva 11 (Static Initializers):

1. Se modifica el atributo `counter` de la clase `Employee` y se agrega un bloque `static` como muestra el *Código 5*:

```

25 private static int counter;
26 static {
27     counter = 0;
28 }

```

Código 5

Diapositiva 13 (The final Keyword):

1. Se modifica el atributo `id` como `final` de la clase `Employee` como muestra el *Código 6*, se realiza una reestructuración de la clase para cambiar el atributo `id` a `ID`, dado por el diagrama UML de la *Imagen 1*:

```

15 private final int ID;

```

Código 6

2. Se modifica el método `toString()` de la clase `Employee` como se muestra el *Código 7*:

```

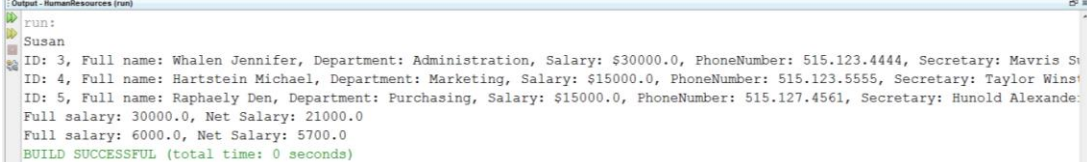
143 @Override
144 public String toString() {
145     return "ID: " + ID + ", Full name: " + lastName + " " +
firstName

```

```
146     + ", Department: " + department.getName() + ", Salary: $"
147     + salary + ", PhoneNumber: " + phoneNumber;
148 }
```

Código 7

3. Se obtiene la salida:



```
run:
Susan
ID: 3, Full name: Whalen Jennifer, Department: Administration, Salary: $30000.0, PhoneNumber: 515.123.4444, Secretary: Mavis St
ID: 4, Full name: Hartstein Michael, Department: Marketing, Salary: $15000.0, PhoneNumber: 515.123.5555, Secretary: Taylor Wins
ID: 5, Full name: Raphaely Den, Department: Purchasing, Salary: $15000.0, PhoneNumber: 515.127.4561, Secretary: Hunold Alexande
Full salary: 30000.0, Net Salary: 21000.0
Full salary: 6000.0, Net Salary: 5700.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Diapositiva 25 (Advanced Enumerated Types):

1. Se modifica la clase country como se muestra en el *Código 8*:

```
11 public enum Country {
12     ARGENTINA("Argentina"),
13     AUSTRALIA("Australia");
14
15     private String name;
16
17     private Country(String name) {
18         this.name = name;
19     }
20
21     public String getName() {
22         return name;
23     }
24
25     public void setName(String name) {
26         this.name = name;
27     }
28
29 }
```

Código 8

2. Se modifica la clase TestEmployee como se muestra el *Código 9*:

```
21 public class TestEmployee {
22
23     public static void main(String[] args) {
24         Location lArgentina = new Location("12-98 Victoria street",
25         2901,
26         "Sidney", "New South Wales", Country.ARGENTINA);
27         Location lAustralia = new Location("20 Rue des Corps-Saints",
28         1730,
29         "Geneva", "Geneve", Country.AUSTRALIA);
30
31     }
32 }
```

Código 9

Diapositiva 25 (Static Imports):

1. Se agrega la sentencia `import static` a la clase `TestEmployee` como se muestra en el *Código 10*:

```
16 import static employees.Country.*;
```

Código 10

2. Se organiza las importaciones con ayuda del IDE Netbeans como se muestra la imagen:

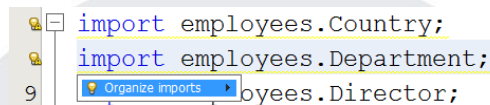


Imagen 2

3. Se modifica la clase `TestEmployee` como se muestra el *Código 11*:

```
23 public static void main(String[] args) {
24     Location lArgentina = new Location("12-98 Victoria street", 2901,
    "Sidney", "New South Wales", ARGENTINA);
25     Location lAustralia = new Location("20 Rue des Corps-Saints",
    1730,
    "Geneva", "Geneve", AUSTRALIA);
```

Código 11

Diapositiva 45 (Interfaces):

1. Se crea una nueva interfaz con el nombre de `ITaxService.java` en el paquete `taxrate`.
2. Se modifica la interfaz `ITaxService.java` como muestra el *Código 12*:

```
10 */
11 public interface ITaxService {
12
13     public double getTaxeRate(double amount);
14 }
```

Código 12

3. Se modifica la clase `TaxService` como muestra el *Código 13*:

```
11 public class TaxService implements taxerate.ITaxService {
12
13     @Override
14     public double getTaxeRate(double amount) {
15         if (amount * 12 > 400000.00) {
16             return 0.40;
17         } else if (amount * 12 > 300000.00) {
18             return 0.30;
19         } else if (amount * 12 > 200000.00) {
20             return 0.20;
```

```

21     } else if (amount * 12 > 100000.00) {
22         return 0.10;
23     } else {
24         return 0.05;
25     }
26 }
27
28     public double findnetpay(Employee e) {
29         return e.getSalary() - e.getSalary() *
getTaxRate(e.getSalary());
30     }
31 }

```

Código 13

4. Se crea un nuevo paquete con el nombre de `invoice` y se copia la clase `Invoice` ubicada en "Resources/Modulo 7/Invoice.java" y se modifica la clase como muestra el Código 14:

```

11 public class Invoice implements taxerate.ITaxService {
12
13     ...
14     public double getTotal() {
15         return getSubTotal() + (getSubTotal() *
getTaxRate(getSubTotal()));
16     }
17
18     @Override
19     public double getTaxRate(double amount) {
20         if (amount > 150000) {
21             return .18;
22         } else {
23             return .16;
24         }
25     }
26
27     @Override
28     public String toString() {
29         return "Item Number: " + itemNumber + ", Description: "
+ description + ", " + "UnitPrice: " + unitPrice
+ ", SubTotal: " + getSubTotal() + ", Total: "
+ getTotal();
30     }
31 }

```

Código 14

5. Se agrega el Código 15 a la clase `TestEmployee`:

```

82     Invoice invoice = new Invoice(1, "monthly salary",
83     e3.getSalary(), 1);
84     System.out.println(invoice);

```

Código 15

6. Se obtiene la salida:

```
Output - HumanResources (run)
run:
Susan
ID: 3, Full name: Whalen Jennifer, Department: Administration, Salary: $30000.0, PhoneNumber: 515.123.4444, Secretary: Mavris Susan
ID: 4, Full name: Hartstein Michael, Department: Marketing, Salary: $15000.0, PhoneNumber: 515.123.5555, Secretary: Taylor Winston
ID: 5, Full name: Raphaely Den, Department: Purchasing, Salary: $15000.0, PhoneNumber: 515.127.4561, Secretary: Hunold Alexander
Full salary: 30000.0, Net Salary: 21000.0
Full salary: 6000.0, Net Salary: 5700.0
Item Number: 1, Description: monthly salary, UnitPrice: 5500.0, SubTotal: 5500.0, Total: 6380.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

