

1. Human Resources (Módulo 6)

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 4 (Employee):

1. Se agrega el método `getDetails` a la clase `Employee` como se muestra en la *Imagen 1* localizada en el paquete `images` del proyecto "HumanResources" con el nombre de "HR v9.png".

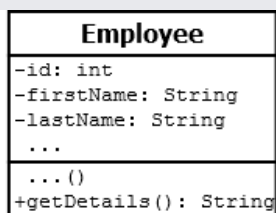


Imagen 1

2. Se desarrolla el *Código 1* en la clase `Employee`:

```

Alumno\Resources\Modulo 2\HumanResources\src\employees\Employee.java
136 public String getDetails() {
137     return "Full name: " + firstName + " " + lastName
138         + ", Department: " + department.getName() + ", Salary: $" + salary + ", PhoneNumber: " +
phoneNumber;
139 }
  
```

Código 1

Diapositiva 6 (Manager hereda de Employee):

1. Se crea la clase `Manager` y la clase `Director` en el paquete `employees`, correspondiente al diagrama UML de la *Imagen 2* localizado en el paquete `images` del proyecto "HumanResources" con el nombre de "HR v10.png".

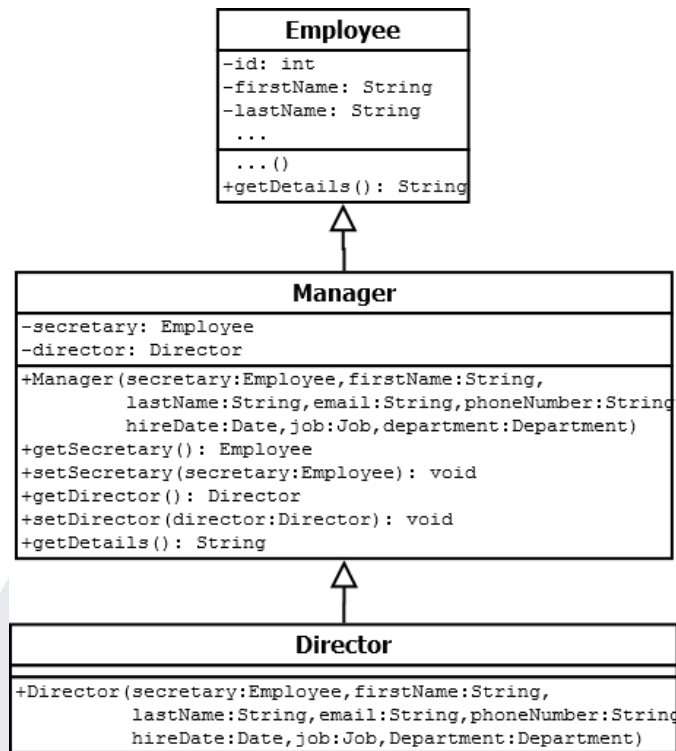


Imagen 2

2. Se desarrolla el Código 2:

Alumno\Resources\Modulo 2\HumanResources\src\employees\Manager.java

```

13 public class Manager extends Employee {
14
15     private Employee secretary;
16     private Director director;
17
18     public Manager(Employee secretary, String firstName, String lastName, String email, String phoneNumber,
19 Date hireDate, Job job, Department department) {
20         super(firstName, lastName, email, phoneNumber, hireDate, job);
21         this.secretary = secretary;
22         super.setDepartment(department);
23     }
24
25     public Employee getSecretary() {
26         return secretary;
27     }
28
29     public void setSecretary(Employee secretary) {
30         this.secretary = secretary;
31     }
32
33     public Director getDirector() {
34         return director;
35     }
36
37     public void setDirector(Director director) {
38         this.director = director;
39     }
  
```

Manager

Alumno\Resources\Modulo 2\HumanResources\src\employees\Director.java

```

13 public class Director extends Manager {
  
```

```

14
15     public Director(Employee secretary, String firstName, String lastName, String email, String phoneNumber,
Date hireDate, Job job, Department department) {
16         super(secretary, firstName, lastName, email, phoneNumber, hireDate, job, department);
17     }
18 }

```

Director

Diapositiva 19 (Overriding Methods):

1. Se sobrescribe el método `getDetails` en `Manager` tal como se muestra en la *Imagen 6*:

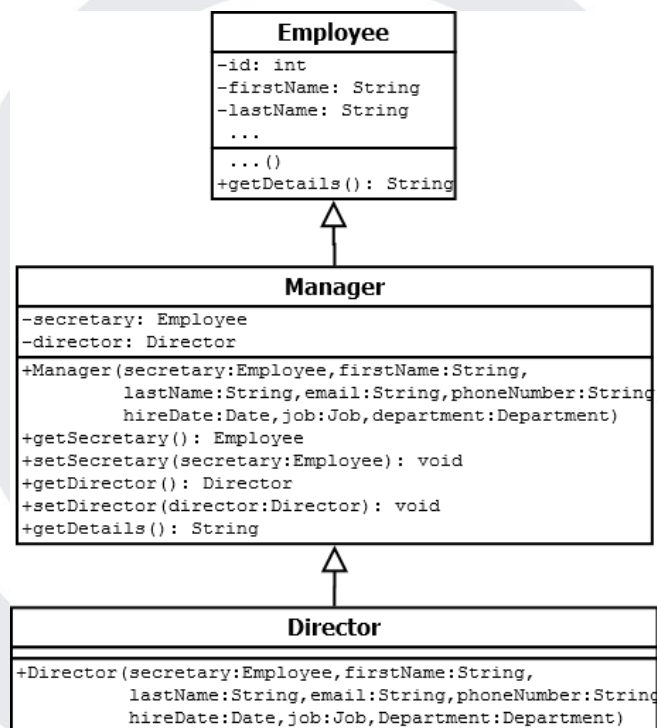


Imagen 6

3. Se desarrolla el Código 5 con las herramientas NetBeans a continuación los pasos:

Alumno\Resources\Modulo 2\HumanResources\src\employees\Manager.java

```

40     @Override
41     public String getDetails() {
42         return super.getDetails();
43     }

```

Código 5

4. Clic derecho al final del código de la clase `Manager`, se selecciona "Insertar Código" como muestra la *Imagen 7*:

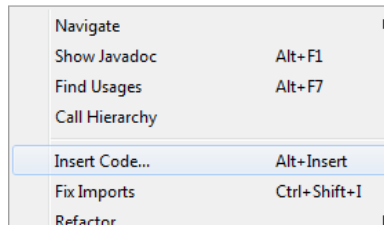


Imagen 7

5. Se selecciona "Override Method..." como muestra la *Imagen 8*:

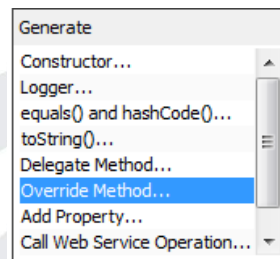


Imagen 8

6. Se selecciona "getDetails" como muestra la *Imagen 9* y presionamos "Generate":

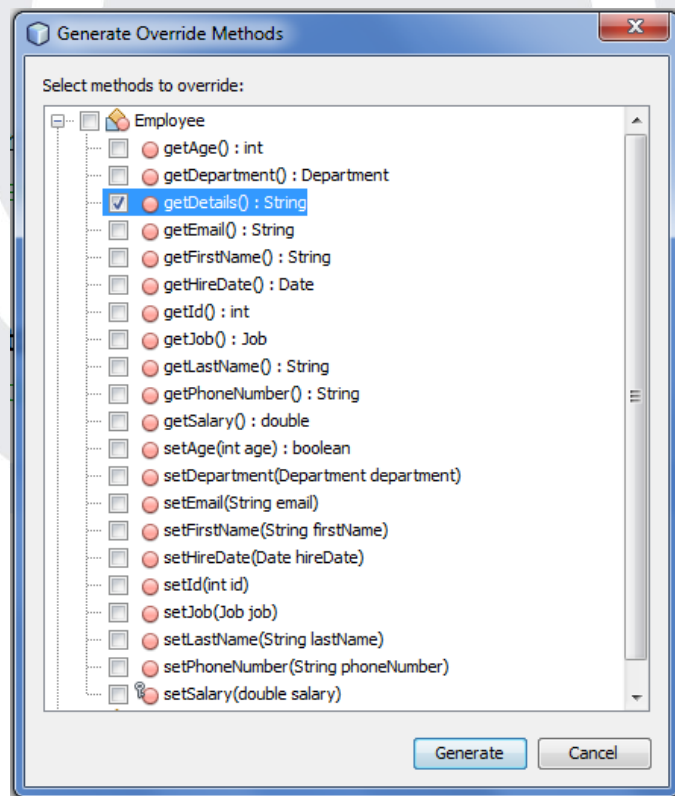


Imagen 9

Diapositiva 20 (Overridden Methods Cannot Be Less Accessible):

1. Se cambia el modificador del método `getDetails()` en `Manager` como se muestra en el *Código 6* para probar la regla de los métodos sobrescritos no pueden ser menos accesibles.

2. Alumno\Resources\Modulo 2\HumanResources\src\employees\Manager.java

```
40  @Override
41  protected String getDetails() {
42      return super.getDetails();
43  }
```

Código 6

3. Se cambia el modificador del método `getDetails()` en `Manager` a `public`.

Diapositiva 22 (Invoking Overridden Methods):

1. Se explica la palabra reservada `super` utilizada en el método `getDetails()` y se modifica el comportamiento del método como se muestra el *Código 7*:

Alumno\Resources\Modulo 2\HumanResources\src\employees\Manager.java

```
40  @Override
41  public String getDetails() {
42      return super.getDetails() + ", Secretary: " + secretary.getLastName()
43          + " " + secretary.getFirstName();
44  }
```

Código 7

Diapositiva 30 (Polimorfismo):

1. Se explica el error de compilación de la línea 61 de la clase *TestEmployee*:

Alumno\Resources\Modulo 2\HumanResources\src\test\TestEmployee.java

```
66  System.out.println(m1.getSecretary().getFirstName());
```

Código 8

2. Se comenta la línea 61 de la clase *TestEmployee*.

Diapositiva 32 (Invocación de métodos virtuales):

1. Se agrega el Código 9, a la clase *TestEmployee*:

Alumno\Resources\Modulo 2\HumanResources\src\test\TestEmployee.java

```
76  System.out.println(m1.getDetails());
77  System.out.println(m2.getDetails());
78  System.out.println(m3.getDetails());
```

Código 9

2. Se obtiene la salida:

```

run:
Full name: Jennifer Whalen, Department: Administration, Salary: $15000.0, PhoneNumber: 515.123.4444, Secretary: Mavis Susan
Full name: Michael Hartstein, Department: Marketing, Salary: $9000.0, PhoneNumber: 515.123.5555, Secretary: Taylor Winston
Full name: Den Raphaely, Department: Purchasing, Salary: $8000.0, PhoneNumber: 515.127.4561, Secretary: Hunold Alexander
BUILD SUCCESSFUL (total time: 0 seconds)

```

Imagen 10

Diapositiva 36 (Polymorphic Arguments):

1. Se copia al paquete employees la clase TaxService ubicada en "Resources/Modulo 6/ TaxService.java"
2. Se agrega el Código 10, a la clase TaxService:

```

Alumno\Resources\Modulo 2\HumanResources\src\employees\TaxService.java
27 public double findnetpay(Employee e) {
28     return e.getSalary() - e.getSalary() * getTaxRate(e);
29 }

```

Código 10

3. Se agrega el Código 11, a la clase TestEmployee:

```

Alumno\Resources\Modulo 2\HumanResources\src\test\TestEmployee.java
83     TaxService taxSvc = new TaxService();
84     System.out.println("Full salary: " + m1.getSalary() + ", Net Salary: " + taxSvc.findnetpay(m1));
85     System.out.println("Full salary: " + e1.getSalary() + ", Net Salary: " + taxSvc.findnetpay(e1));

```

Código 11

5. Se obtiene la salida:

```

Output - HumanResources (run) Search Results
run:
Full name: Jennifer Whalen, Department: Administration, Salary: $15000.0, PhoneNumber: 515.123.4444, Secretary: Ma
vris Susan
Full name: Michael Hartstein, Department: Marketing, Salary: $9000.0, PhoneNumber: 515.123.5555, Secretary: Taylor
Winston
Full name: Den Raphaely, Department: Purchasing, Salary: $8000.0, PhoneNumber: 515.127.4561, Secretary: Hunold Ale
xander
Full salary: 15000.0, Net Salary: 13500.0
Full salary: 3000.0, Net Salary: 2850.0
BUILD SUCCESSFUL (total time: 0 seconds)

```

Imagen 11

Diapositiva 40 (Casting Objects):

1. Se agrega el Código 12 a la clase TestEmployee:

```

Alumno\Resources\Modulo 2\HumanResources\src\test\TestEmployee.java
85     Manager m = (Manager) m1;
86     System.out.println("Manager Full name: " + m.getFirstName() + m.getLastName() + ", Secretary: " +
m.getSecretary().getFirstName());

```

Código 12

2. Se obtiene la salida:

```
Output - HumanResources (run) Search Results
Full name: Jennifer Whalen, Department: Administration, Salary: $15000.0, PhoneNumber: 515.123.4444, Secretary: Ma
vris Susan
Full name: Michael Hartstein, Department: Marketing, Salary: $9000.0, PhoneNumber: 515.123.5555, Secretary: Taylor
Winston
Full name: Den Raphaely, Department: Purchasing, Salary: $8000.0, PhoneNumber: 515.127.4561, Secretary: Hunold Ale
xander
Full salary: 15000.0, Net Salary: 13500.0
Full salary: 3000.0, Net Salary: 2850.0
Manager Full name: JenniferWhalen, Secretary: Susan
BUILD SUCCESSFUL (total time: 0 seconds)
```

Imagen 11

Diapositiva 48 (Overloading Methods):

1. Se agrega el Código 13, a la clase Director:

```
Alumno\Resources\Modulo 2\HumanResources\src\employees\Director.java
19 public void setSalary(Employee employee, double salary) {
20     employee.setSalary(salary);
21 }
```

Código 13

2. Se agrega la siguiente regla de Negocio a la aplicación:

“Solo los Directores podrán asignar y modificar el salario de todos los empleados”.

3. Es tarea del Estudiante resolver lo anterior, basta con poner el método setSalary protected en la clase Employee para cumplir con la regla anterior tal como muestra el Código 14:

```
Alumno\Resources\Modulo 2\HumanResources\src\employees\Employee.java
108 protected void setSalary(double salary) {
109     if (salary > this.getJob().getMaxSalary()) {
110         salary = this.getJob().getMaxSalary();
111     } else if (salary < this.getJob().getMinSalary()) {
112         salary = this.getJob().getMinSalary();
113     }
114     this.salary = salary;
115 }
```

Código 14

3. Se obtendrán errores de compilación en la clase TestEmployee, se reemplazan estas líneas con el Código 15:

```
Alumno\Resources\Modulo 2\HumanResources\src\test\TestEmployee.java
70 Director director = new Director(s3, "William", "Gietz", "WGIEZ", "515.123.8181", new Date(2002,
6, 7), jDirector, dAdministration);
71 director.setSalary(m1, 32000);
72 director.setSalary(m2, 45000);
73 director.setSalary(m3, 58000);
74
75 director.setSalary(e1, 12000);
76 director.setSalary(e2, 16000);
77 director.setSalary(e3, 22000);
```

Diapositiva 50 (Overloading Constructors):

1. Se agrega un nuevo constructor a la clase Employee con los atributos que muestra la Imagen 10 y Código 16:

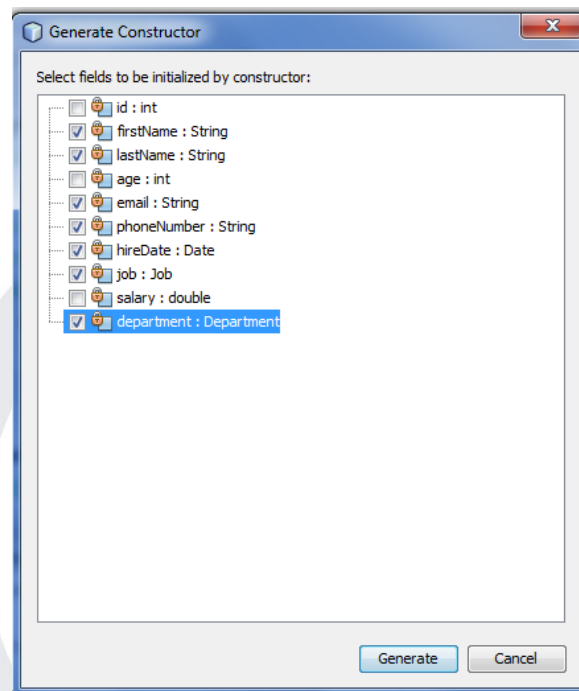


Imagen 10

```
Alumno\Resources\Modulo 2\HumanResources\src\employees\Employee.java
35 public Employee(String firstName, String lastName, String email, String phoneNumber, Date hireDate,
36 Job job, Department department) {
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     setDepartment(department);
44 }
```

Código 16

Diapositiva 51 (Constructors Are Not Inherited)

1. Se agrega un nuevo constructor a la clase Manager con los atributos que muestra la Imagen 11, denotados en el diagrama UML de la Imagen 13 ubicado en el paquete images del proyecto "HumanResources" con el nombre de "HR v11.png" :

```
Alumno\Resources\Modulo 2\HumanResources\src\employees\Manager.java
24 public Manager(String firstName, String lastName, String email, String phoneNumber, Date
25 hireDate, Job job) {
26     super(firstName, lastName, email, phoneNumber, hireDate, job);
27 }
```


Código 17

2. Se agrega un nuevo constructor a la clase Director con los atributos que muestra en el siguiente Código 18:

```
Alumno\Resources\Modulo 2\HumanResources\src\employees\Director.java
19 public Director(String firstName, String lastName, String email, String phoneNumber, Date
hireDate, Job job) {
20     super(firstName, lastName, email, phoneNumber, hireDate, job);
21 }
```

Código 18

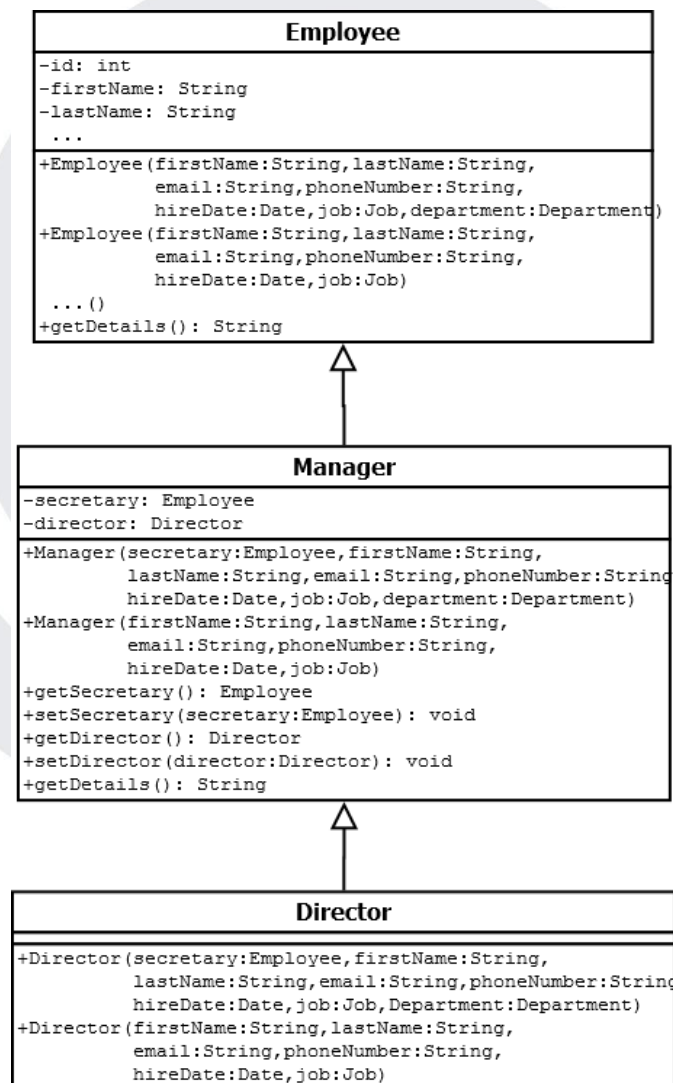


Imagen 13

Diapositiva 53 (Invoking Parent Class Constructors)

1. Se modifica los constructores de la clase `Employee` como muestra el *Código 19*:

Alumno\Resources\Modulo 2\HumanResources\src\employees\Employee.java

```
26 public Employee(String firstName, String lastName, String email, String phoneNumber, Date hireDate, Job job) {  
27     this.firstName = firstName;  
28     this.lastName = lastName;  
29     this.email = email;  
30     this.phoneNumber = phoneNumber;  
31     this.hireDate = hireDate;  
32     this.job = job;  
33 }  
34  
35 public Employee(String firstName, String lastName, String email, String phoneNumber, Date hireDate, Job job, Department department) {  
36     this(firstName, lastName, email, phoneNumber, hireDate, job);  
37     setDepartment(department);  
38 }
```

Código 19

2. Se modifica los constructores de la clase `Manager` como muestra el *Código 20*:

Alumno\Resources\Modulo 2\HumanResources\src\employees\Manager.java

```
18 public Manager(Employee secretary, String firstName, String lastName, String email, String phoneNumber, Date hireDate, Job job, Department department) {  
19     this(firstName, lastName, email, phoneNumber, hireDate, job);  
20     this.secretary = secretary;  
21     super.setDepartment(department);  
22 }
```

Código 20

3. Se modifica el constructor de la clase `Employee` como muestra el *Código 21* para probar que la primera línea de un constructor debe ser la llamada a `super` o a `this` a un constructor:

Alumno\Resources\Modulo 2\HumanResources\src\employees\Employee.java

```
35 public Employee(String firstName, String lastName, String email, String phoneNumber, Date hireDate, Job job, Department department) {  
36     this.setDepartment(department);  
37     this(firstName, lastName, email, phoneNumber, hireDate, job);  
38 }
```

Código 21

4. Se inserta un constructor sin argumentos en la clase `Manager`, utilizando la herramienta de insertar código, se obtendrá un error de compilación debido a la llamada implícita de `super()`:

Alumno\Resources\Modulo 2\HumanResources\src\employees\Manager.java

```
28 public Manager() {  
29 }
```

Código 22

Diapositiva 62 (The equalsMethod)

1. Se explicará en el Modulo 9.

Diapositiva 67 (The toStringMethod)

1. Se renombra el método `getDetails()` a `toString()` de la clase `Employee` y `Manager` y se agrega la notación `@Override`, como se muestra en el código 22:

```
Alumno\Resources\Modulo 2\HumanResources\src\employees\Employee.java
141     @Override
142     public String toString() {
143         return "Full name: " + firstName + " " + lastName
144             + ", Department: " + department.getName() + ", Salary: $" + salary + ", PhoneNumber: " +
phoneNumber;
145     }

Alumno\Resources\Modulo 2\HumanResources\src\employees\Manager.java
47     @Override
48     public String toString() {
49         return super.toString() + ", Secretary: " + secretary.getLastName()
50             + " " + secretary.getFirstName();
51     }
```

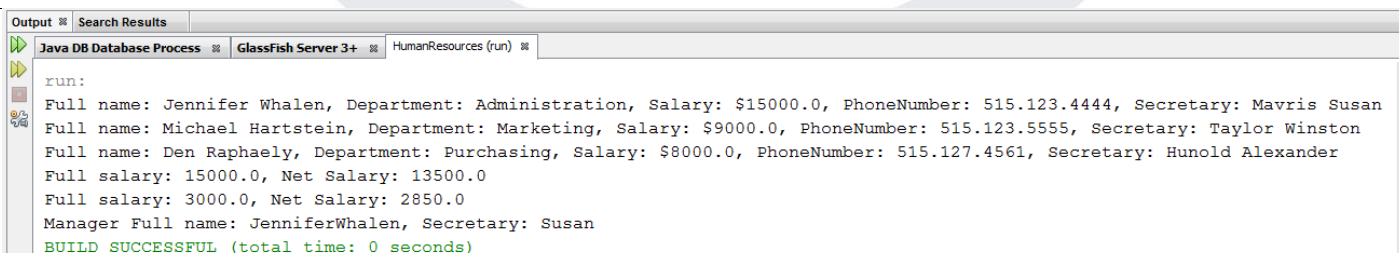
Código 22

1. Se cambia la clase `TestEmployee` para corregir los errores de compilación como se muestra en el Código 21:

```
Alumno\Resources\Modulo 2\HumanResources\src\test\TestEmployee.java
79     System.out.println(m1);
80     System.out.println(m2);
81     System.out.println(m3);
```

Código 23

2. Se obtiene la salida:



```
run:
Full name: Jennifer Whalen, Department: Administration, Salary: $15000.0, PhoneNumber: 515.123.4444, Secretary: Mavis Susan
Full name: Michael Hartstein, Department: Marketing, Salary: $9000.0, PhoneNumber: 515.123.5555, Secretary: Taylor Winston
Full name: Den Raphaely, Department: Purchasing, Salary: $8000.0, PhoneNumber: 515.127.4561, Secretary: Hunold Alexander
Full salary: 15000.0, Net Salary: 13500.0
Full salary: 3000.0, Net Salary: 2850.0
Manager Full name: JenniferWhalen, Secretary: Susan
BUILD SUCCESSFUL (total time: 0 seconds)
```

Imagen 11