ENTERPRISE JAVA DEVELOPER

JAVA ORIENTADO A OBJETOS

INTERFACES

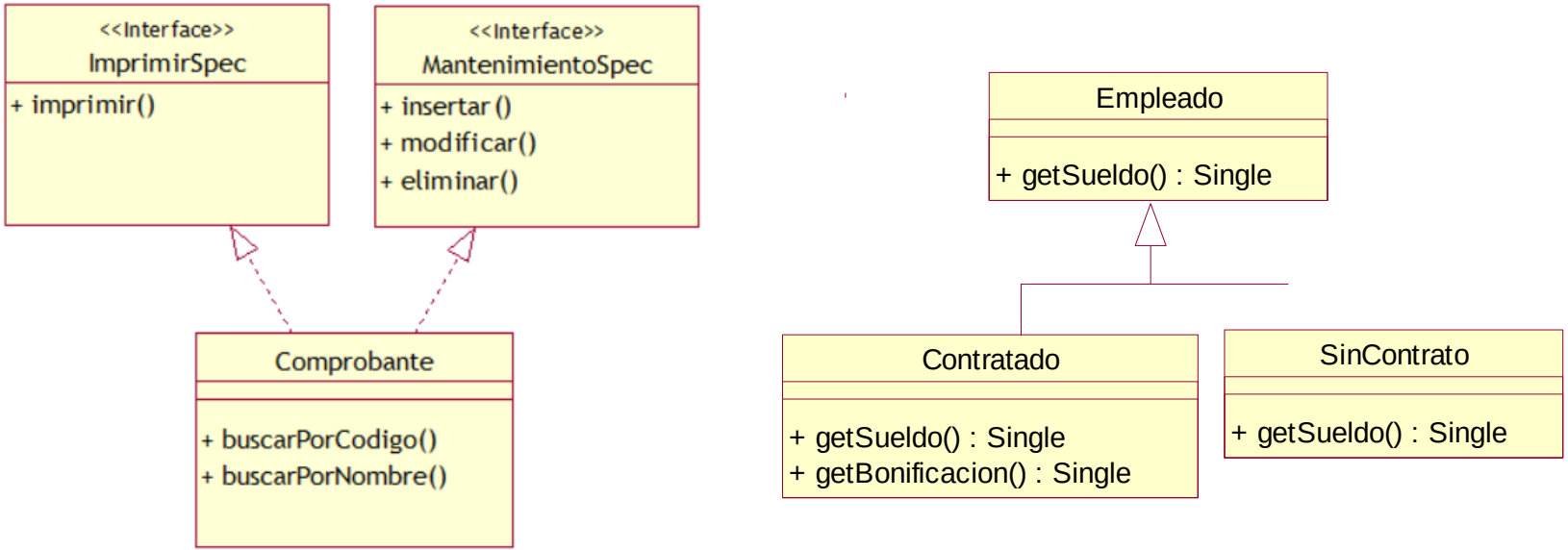
Eric Gustavo Coronel Castillo
gcoronelc.blogspot.com

Temas

- Objetivo
- Interface
- Clase Concreta, Abstracta e Interface
- Polimorfismo
- Operador instanceof
- Casting
- Proyecto Ejemplo

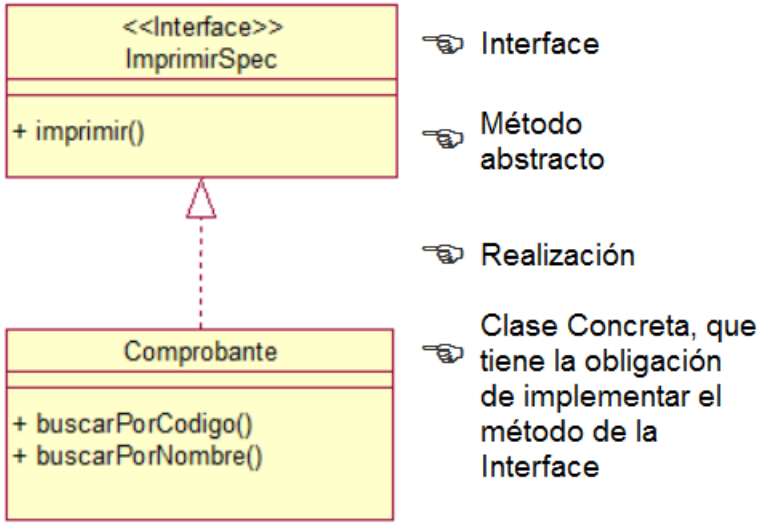
OBJETIVOS

- Aplicar interfaces en el diseño de componentes software.
- Aplicar el polimorfismo en el diseño de componentes software



INTERFACE

- Solo contienen operaciones (métodos) sin implementación, es decir solo la firma de los métodos.
- Las clases son las encargadas de implementar las operaciones (métodos) de una o varias interfaces.
- A nivel de interfaces si existe la herencia múltiple, una interface puede heredar de varias interfaces.




```java
public interface ImprimirSpec{

   void imprimir();

}
```

```java
public class Comprobante implements ImprimirSpec{

   public void buscarPorCodigo() {
      // Implementa el método de la clase
   }

   public void buscarPorNombre() {
      // Implementa el método de la clase
   }

   @Override
   public void imprimir() {
     // Implementa el método de la interface
   }

}
```

INTERFACE

Ejemplo de Herencia múltiple de Interface.

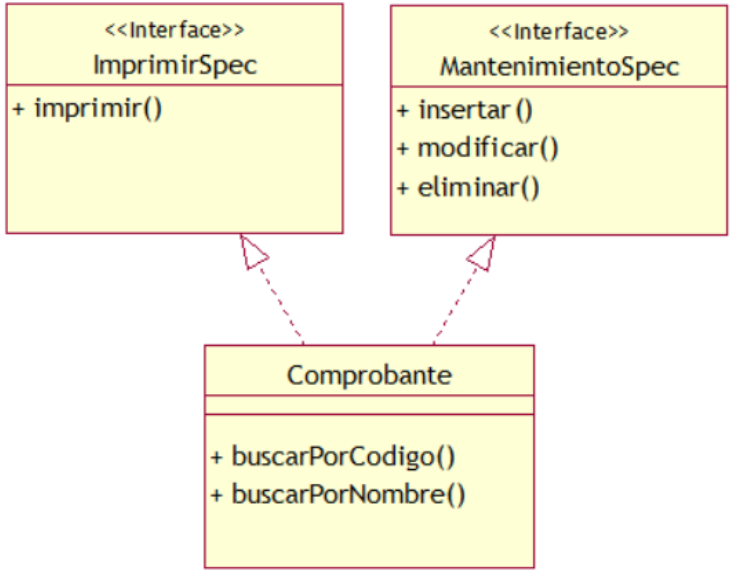
```
public interface ImprimirSpec {
   void imprimir();
}
```

```
public interface MantenimientoSpec {
   void insertar();
   void modificar();
   void eliminar();
}
```

```
public class Comprobante
implements ImprimirSpec, MantenimientoSpec {

   // Implementa sus propios métodos y
   // los métodos de las interfaces

}
```



CLASE CONCRETA, ABSTRACTA E INTERFACE

CARACTERISTICA	CLASE CONCRETA	CLASE ABSTRACTA	INTERFACE
HERENCIA	extends (simple)	extends (simple)	implements, * extends ** (múltiple)
INSTANCIABLE	Si	No	No
IMPLEMENTA	Métodos	Algunos métodos	Nada
DATOS	Se permite	Se permite	No se permite ***

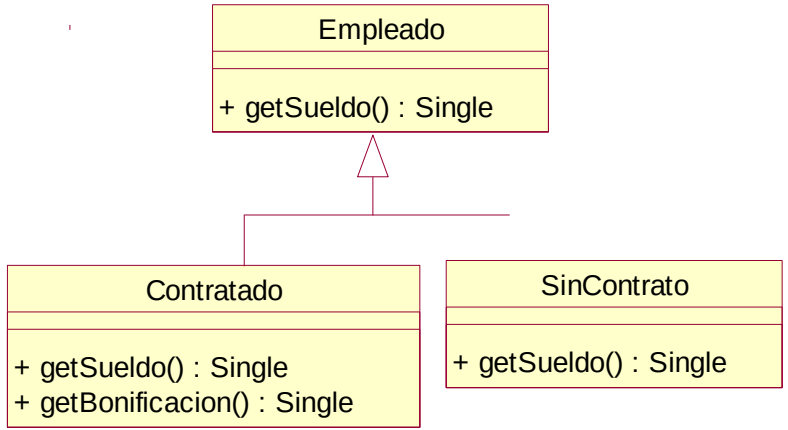
* Una clase puede implementar varias interfaces
** Una interface puede heredar de varias interfaces
*** Las variables que se declaran en una interface son implícitamente estáticas, finales y públicas.

POLIMORFISMO

- Existe polimorfismo cuando un método definido en una clase o interface es implementado de varias formas en otras clases.
- Algunos ejemplos de polimorfismos de herencia son: *sobre-escritura*, *implementación* de métodos abstractos (clase abstracta e interface).
- Es posible apuntar a un objeto con una variable de tipo de *clase padre* (supercalse), esta sólo podrá acceder a los miembros (campos y métodos) que le pertenece.

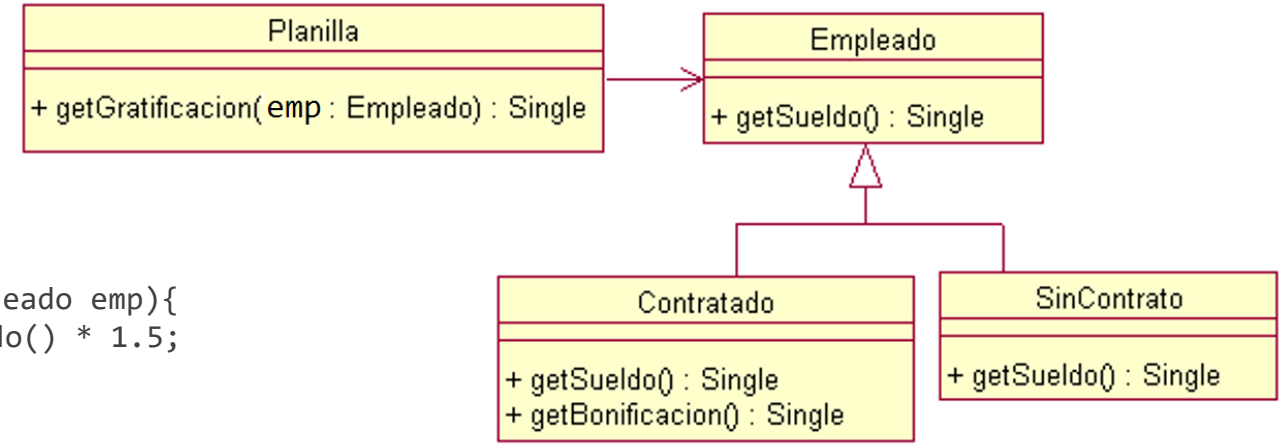
```
// Variable de tipo Empleado y apunta a un
// objeto de tipo Contratado.
Empleado objEmp = new Contratado();

// Invocando sus métodos
double s = objEmp.getSueldo();        // OK
double b = objEmp.getBonificacion(); // Error
```



POLIMORFISMO

- El método **getGratificacion** puede recibir objetos de tipo **Empleado** o subtipos a este.
- Cuando invoque el método **getSueldo** se ejecutará la versión correspondiente al objeto referenciado.

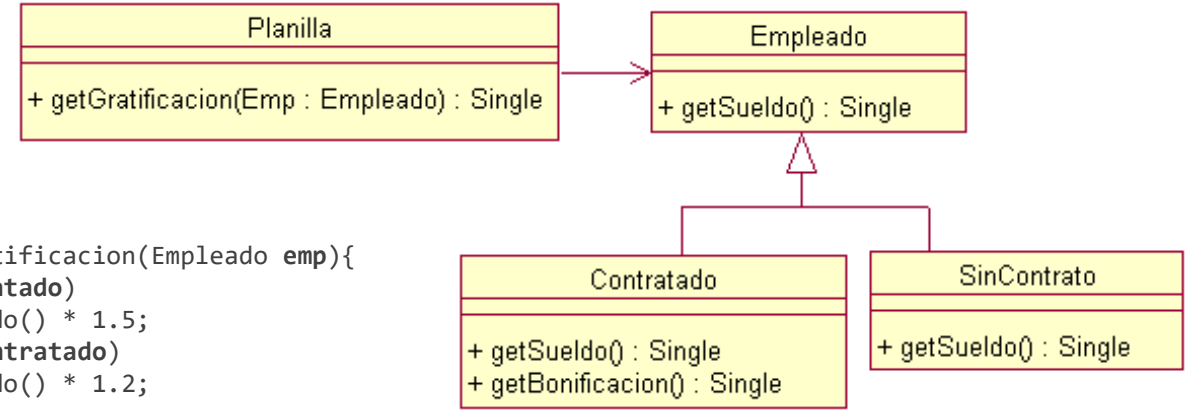



```
public class Planilla {
  public static double
  getGratificacion(Empleado emp){
    return emp.getSueldo() * 1.5;
  }
}
```

```
// Usando la clase Planilla
double g1 = Planilla.getGratificacion(new Contratado());
double g2 = Planilla.getGratificacion(new SinContratado());
```

OPERADOR instanceof

- Este operador permite verificar si el objeto es de un tipo determinado, es decir, el objeto debe pasar por la verificación ES-UN para una determinada clase o interface.




```
public class Planilla {
   public static double getGratificacion(Empleado emp){
      if (emp instanceof Contratado)
            return emp.getSueldo() * 1.5;
      if (emp instanceof SinContratado)
            return emp.getSueldo() * 1.2;
   }
}
```

```
// Usando la clase Planilla
double g1 = Planilla.getGratificacion(new Contratado());
double g2 = Planilla.getGratificacion(new SinContratado());
```

CASTING

- Para restablecer la funcionalidad completa de un objeto, que es de un tipo y hace referencia a otro tipo, debe realizar una conversión (Cast).
- **UpCasting:** Conversión a clases superiores de la jerarquía de clases (Herencia), es automático (conversión implícita), basta realizar la asignación.
- **DownCasting:** Conversión hacia abajo, es decir hacia las subclases de la jerarquía (Herencia), se debe realizar Cast (conversión explicita), si no es compatible genera un error (Excepción).

```
// UpCasting (Conversión implícita)
Contratado a = new Contratado();
Empleado b = a;

// DownCasting (Conversión explícita)
Empleado a = new Contrtado();
Contratado b = (Contratado) a;

// Error de compilación
SinContrato a = new SinContrato();
Contratado b = (Contratado) a;
```

PROYECTO EJEMPLO

- La empresa “Secure Money Exchange” es una casa de cambio, y necesita de un software que le permita a sus empleados realizar una atención ágil y segura.

- El software debe permitirles obtener el tipo de cambio según la moneda.

- Para la solución debe aplicar los conceptos desarrollados en este tema.

JAVA ORIENTADO A OBJETOS

Gracias

Eric Gustavo Coronel Castillo
gcoronelc.blogspot.com