

3

Construcción de la Solución

Algoritmos e Instrucciones



Algoritmos e Instrucciones

- Algoritmo =
 - Secuencia de instrucciones para resolver un problema
 - Secuencia ordenada de pasos para realizar una actividad
- Ejemplos:
 - Algoritmo para preparar unos huevos pericos
 - Algoritmo para amarrarse los zapatos
 - Algoritmo para cambiar una llanta
 - Algoritmo para llegar a una dirección dada



En el computador ...

- Las instrucciones de los algoritmos deben estar escritos en un lenguaje que entienda el computador



Lenguaje de Programación



Nuestro lenguaje: JAVA

Un programa en java está formado por un conjunto de CLASES

Cada Clase se guarda en un archivo distinto



Declaración de clases en Java

Empleado
nombre apellido sexo salario

- Archivo: Empleado.java
`public class Empleado`
{
 // Atributos
 private String nombre;
 private String apellido;
 private int salario;
 private int sexo;
}

Fecha
dia mes año

- Archivo: Fecha.java
`public class Fecha`
{
 private int dia;
 private int mes;
 private int año;
}



Declaración de clases en Java

Empleado
nombre apellido sexo salario

- Archivo: Empleado.java

```
public class Empleado
{
    private String nombre;
    private String apellido;
    private int salario;
    private int sexo;
}
```

Fecha
dia mes año

- Archivo: Fecha.java

```
public class Fecha
{
    private int dia;
    private int mes;
    private int año;
}
```



Declaración de clases en Java

- Archivo: Empleado.java

```
public class Empleado
{
    private String nombre;
    private String apellido;
    private int salario;
    private int sexo;
}
```

- Enteros

- **int**

- Reales

- **double**

- Cadenas de caracteres

- **string**

- Archivo: Fecha.java

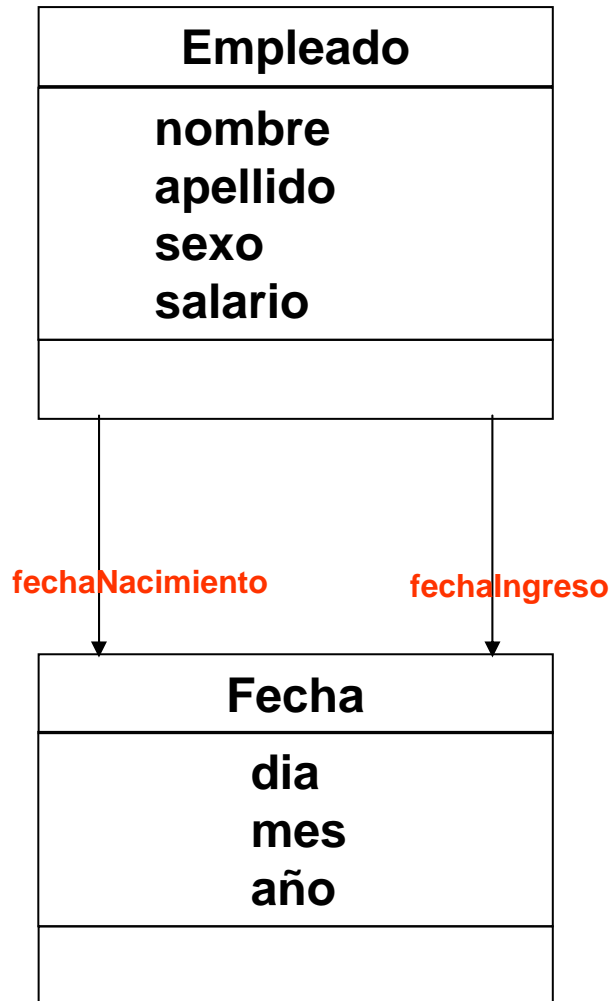
```
public class Fecha
{
    private int dia;
    private int mes;
    private int año;
}
```

Es de tipo entero (int).

Convención: 1= masculino, 2=femenino



Diagrama de asociaciones en JAVA



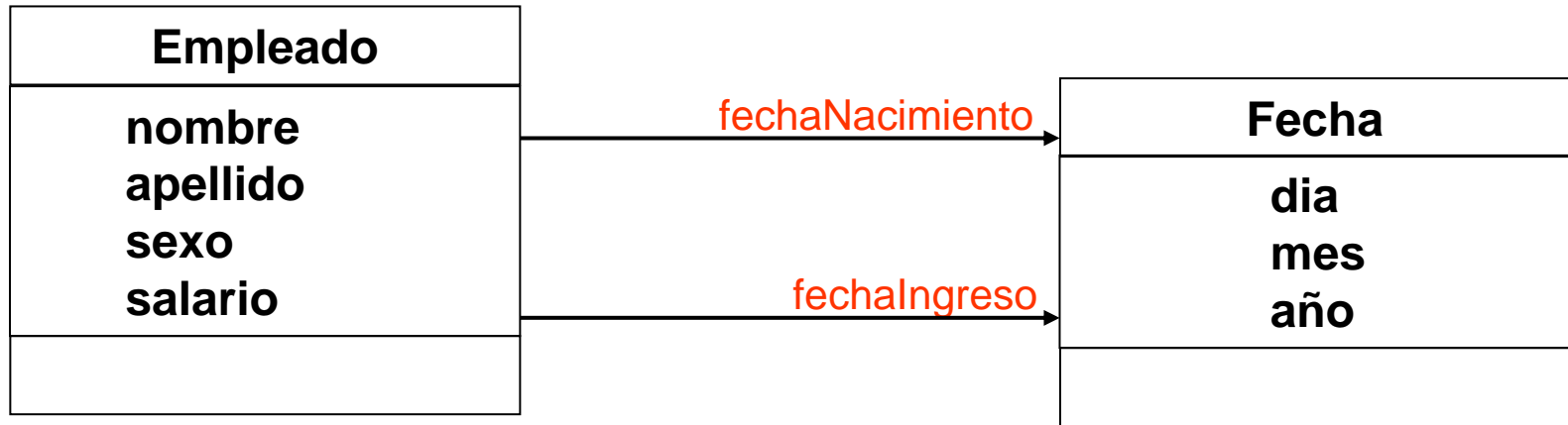
- Archivo: Empleado.java
public class Empleado
{
// Atributos
private String nombre;
private String apellido;
private int salario;
private int sexo;

```
private Fecha fechaNacimiento;  
private Fecha fechaIngreso;  
}
```

El orden de declaración de los atributos NO es importante



Diagrama de clases completo del caso del empleado



Caso de Estudio: Las líneas telefónicas




Las líneas telefónicas

- Se quiere crear una aplicación para controlar los gastos telefónicos de una empresa. La empresa cuenta con tres líneas telefónicas a través de las cuales se pueden realizar llamadas locales, de larga distancia y a celulares.
- La aplicación debe permitir:
 1. Registrar una llamada en alguna de las líneas
 2. Mostrar la información detallada de cada línea
 - Número de llamadas realizadas
 - Duración total de las llamadas en minutos
 - Costo total de las llamadas en pesos
 3. Mostrar un consolidado total de la información de todas las líneas (costo total en pesos de las tres líneas, número total de llamadas realizadas, duración total de llamadas en minutos y el cálculo del costo promedio por minuto según el costo total y el total de minutos).
 4. Reiniciar el uso las líneas telefónicas, dejando todos sus valores en cero.



Interfaz usuario



MiEmpresa
Manejo Líneas Telefónicas

Totales

\$ 0,00

Total Llamadas: 0

Total Minutos: 0

Costo promedio por minuto: N/A

Línea #1



\$ 0,00

Número Llamadas: 0

Número de Minutos: 0

Agregar Llamada

Línea #2



\$ 0,00

Número Llamadas: 0

Número de Minutos: 0

Agregar Llamada

Línea #3



\$ 0,00

Número Llamadas: 0

Número de Minutos: 0

Agregar Llamada

Opciones

Reiniciar

Opción 1

Opción 2



Requerimientos Funcionales

- **R1:** Registrar (agregar) una llamada en alguna de las líneas
- **R2:** Mostrar la información detallada de cada línea.
- **R3:** ...
- **R4:** ...

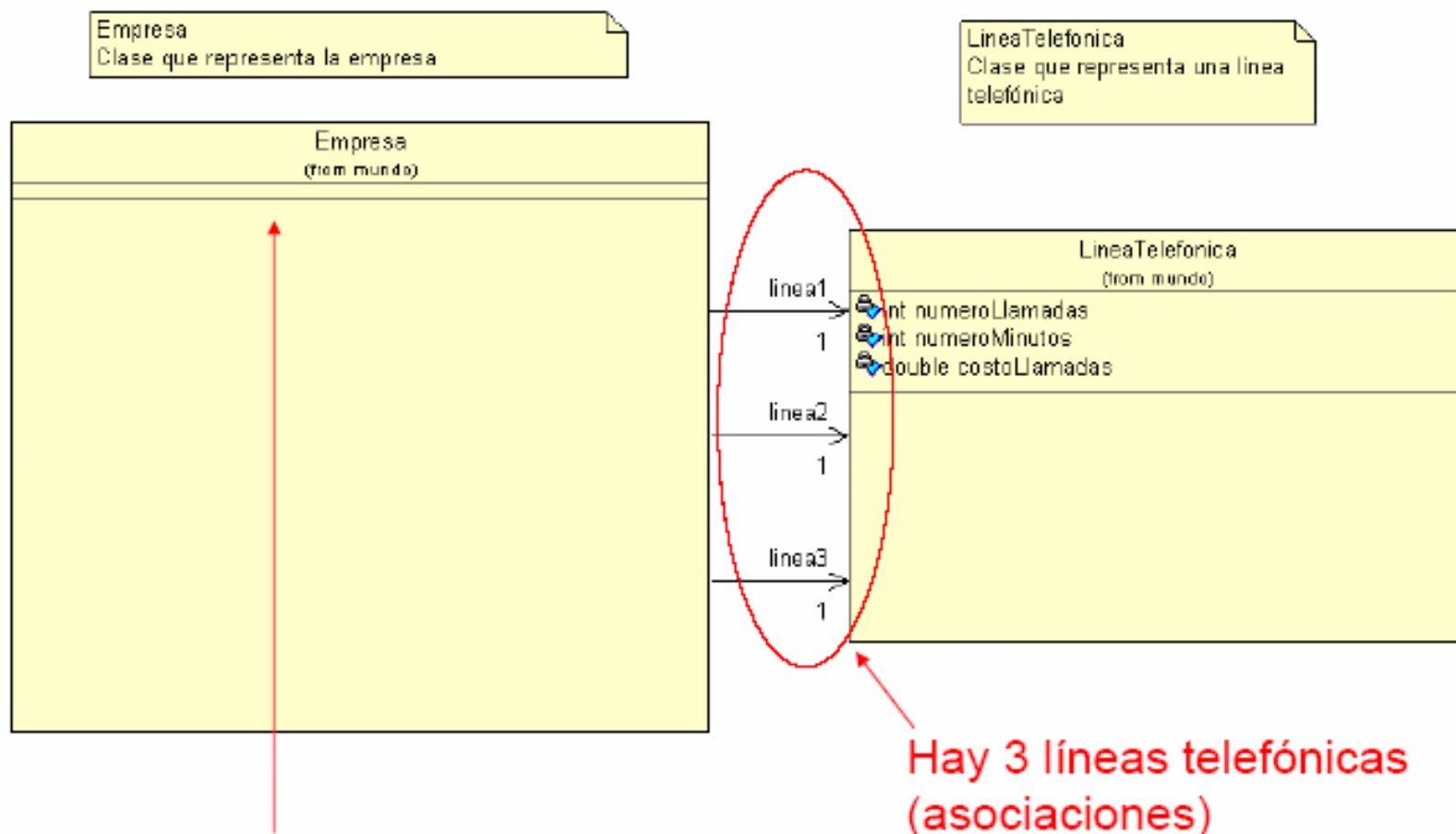


R1

Nombre	R1: Agregar una llamada a una línea telefónica
Resumen	Se agrega una llamada a una línea telefónica. Se debe especificar la cantidad de minutos consumidos, así como el tipo de llamada realizada.
Entradas	
Número de línea, siendo opciones validas la línea 1, 2 o 3.	
Número de minutos consumidos, sabiendo que el número de minutos es un valor positivo.	
Tipo de llamada realizada. Puede ser local, larga distancia o celular.	
Resultados	
La línea telefónica tiene una llamada más.	
Los minutos consumidos por la línea especificada aumentaron según el número de minutos de la llamada.	
El costo total de llamadas realizadas por la línea especificada se incrementó en el costo de la llamada. El valor por minuto de una llamada local es de \$35, de una llamada de larga distancia es de \$380, y de una llamada a celular es de \$999	
Los totales de toda la empresa se actualizan.	



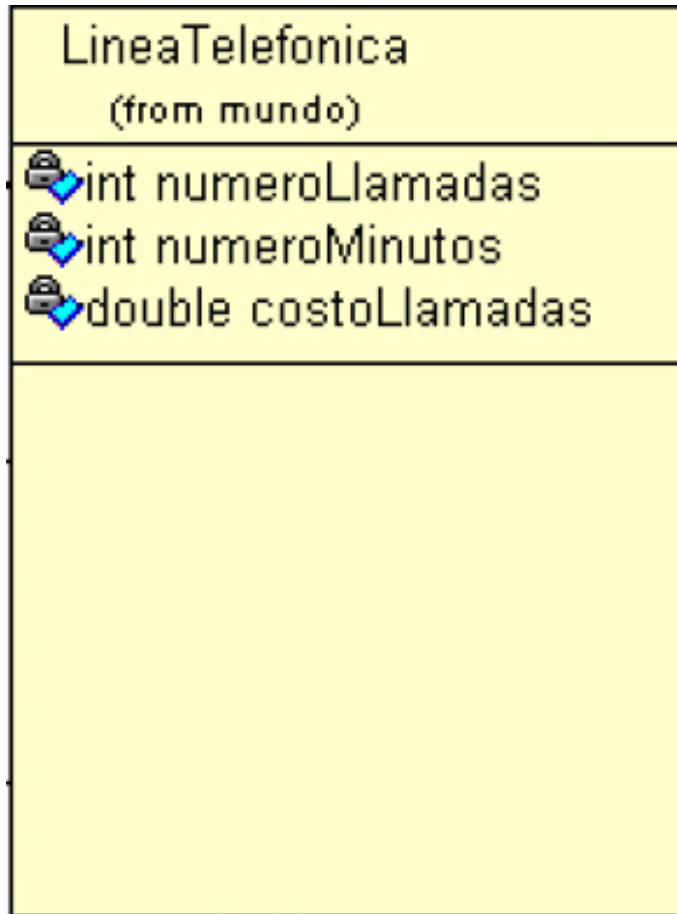
Modelo del Mundo



No tiene atributos particulares



Clase Línea telefónica



Archivo: LineaTelefonica.java

```
public class LineaTelefonica
{
    private int numeroLlamadas;
    private int numeroMinutos;
    private double costoLlamadas;
}
```



Clase Empresa

Archivo: Empresa.java

```
Public class Empresa
```

```
{
```

```
    private LineaTelefonicalinea1;
```

```
    private LineaTelefonicalinea2;
```

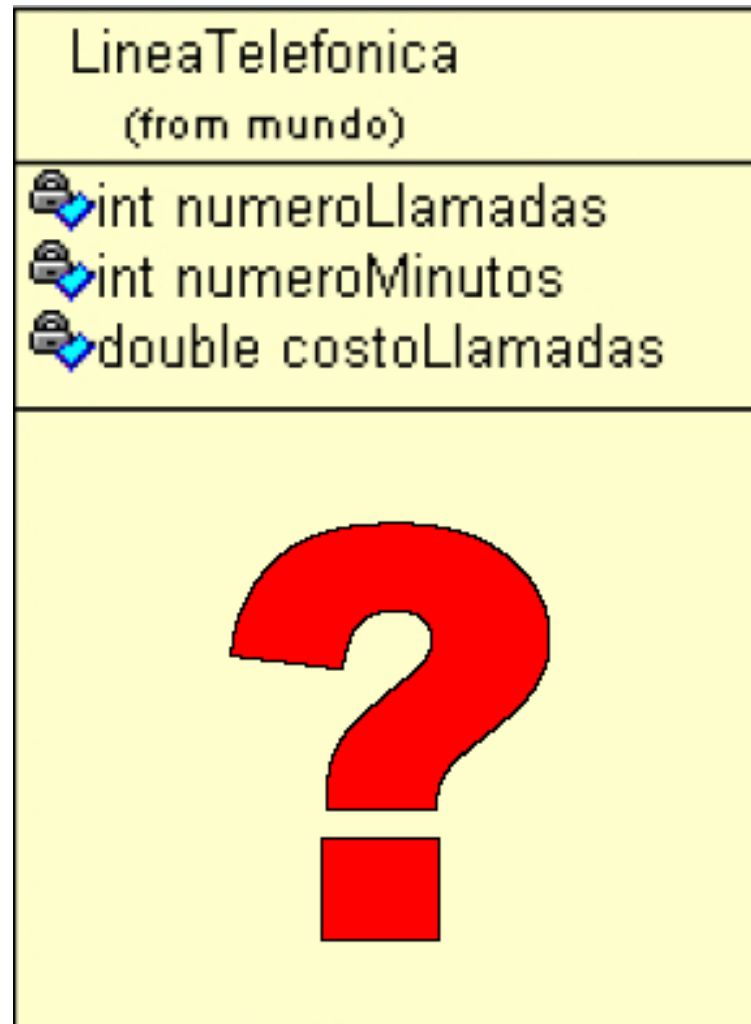
```
    private LineaTelefonicalinea3;
```

```
}
```

**Son las 3
asociaciones**



Qué son los métodos ?



Métodos

- Son los “algoritmos” de la clase.
- Lo que la clase sabe hacer:
 - Resolver un problema puntual
 - Servicio que la clase debe prestar a las demás clases del modelo
- Piense que ...
 - Una clase es la responsable de manejar la información contenida en sus atributos
 - Los métodos son el medio para hacerlo



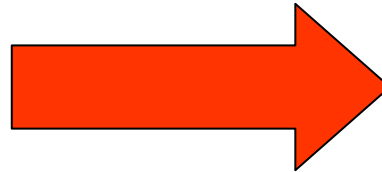
Ejemplo: Qué debe saber hacer una línea telefónica

- Informar:
 - El número total de sus llamadas
 - El costo total de sus llamadas
 - La cantidad de minutos consumidos
- Agregar
 - Una llamada local
 - Una llamada de larga distancia
 - Una llamada de celular



Entonces ...

- Cada una de las acciones que sabe hacer una clase



METODO



Métodos de la LineaTelefonica

- Informar:
 - El número total de sus llamadas
 - El costo total de sus llamadas
 - La cantidad de minutos consumidos
 - Agregar
 - Una llamada local
 - Una llamada de larga distancia
 - Una llamada de celular
- 
- darCostoLlamadas
 - darNumeroLlamadas
 - darNumeroMinutos
 - agregarLlamadaLocal
 - agregarLlamadaLargaDist
 - agregarLlamadaCelular



Un método está compuesto por:

```
public void agregarLlamadaLocal( int minutos )
```

↑
nombre



Un método está compuesto por:

```
public void agregarLlamadaLocal( int minutos )
```

Lista de Parámetros

Conjunto de valores (cada uno con su tipo) necesarios para resolver el problema



Un método está compuesto por:

```
public void agregarLlamadaLocal( int minutos )
```

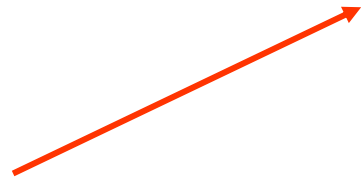
Tipo de Respuesta

Tipo de dato al que pertenece el resultado que va a retornar el método. Si no hay respuesta, se indica el tipo void



Un método está compuesto por:

```
public void agregarLlamadaLocal( int minutos )  
{  
    numeroLlamadas = numeroLlamadas + 1;  
    numeroMinutos = numeroMinutos + minutos;  
    costoLlamadas = costoLlamadas + ( minutos * 35 );  
}
```



Cuerpo del método

- Lista de instrucciones que representa el algoritmo que resuelve el problema puntual
- En el cuerpo se explica la forma de utilizar los valores de los atributos para calcular alguna información o la forma de modificarlos
- Cuerpo del método



Tipos de instrucciones

- Instrucción de asignación
 - Para definir el nuevo valor de un atributo
 - Se construye con un “=”

```
public void agregarLlamadaLocal( int minutos )  
{  
    numeroLlamadas = numeroLlamadas + 1;  
    numeroMinutos = numeroMinutos + minutos;  
    costoLlamadas = costoLlamadas + ( minutos * 35 );  
}
```



Tipos de instrucciones

- `numeroLlamadas = numeroLlamadas + 1;`

Atributo que
va a ser
modificado

Expresión que indica el
nuevo valor que debe
guardarse en el atributo

Pueden hacer parte de
una expresión: los
atributos, los
parámetros y los valores
constantes.

Con operadores
aritméticos (+, -, *, /)



Tipos de instrucciones

- Instrucción de asignación
 - Para definir el nuevo valor de un atributo
 - Se construye con un “=”

```
public void agregarLlamadaLocal( int minutos )  
{  
    numeroLlamadas = numeroLlamadas + 1;  
    numeroMinutos = numeroMinutos + minutos;  
    costoLlamadas = costoLlamadas + ( minutos * 35 );  
}
```

