

El simulador bancario

Caso de estudio #2

El simulador bancario (Problema)

- ▶ Se quiere una aplicación que haga la simulación en el tiempo del **portafolio** bancario de un **cliente**
- ▶ Un **cliente** tiene:
 - ▶ Nombre
 - ▶ Número de cédula (identifica la cuenta)
- ▶ Un **portafolio** tiene:
 - ▶ Una cuenta de ahorro
 - ▶ Una cuenta corriente
 - ▶ Certificado de depósito a término (CDT)
- ▶ Se quiere que el programa permita a una persona simular el manejo de sus productos bancarios:
 - ▶ Hacer las operaciones necesarias sobre los productos que conforman la cuenta
 - ▶ Avanzar mes por mes en el tiempo, para que el cliente pueda ver el resultado de sus movimientos bancarios y el rendimiento de sus inversiones



El simulador bancario (Solución)

The image shows a screenshot of a web-based application titled "Simulador Bancario". The interface is divided into three main sections: "Datos Personales", "Saldo", and "Cálculos".

Datos Personales

Nombre: Sergio López Cédula: 50.152.468 Mes: 1 >>

Saldo

Saldo Corriente: \$ 0,00

Saldo Ahorros: \$ 0,00 [0.6%]

Saldo CDT: \$ 0,00 [0.0%] Total: \$ 0,00

Cálculos

Abrir inversion CDT	Consignar cuenta corriente	Consignar cuenta ahorro	Opcion1
Cerrar inversion CDT	Retirar cuenta corriente	Retirar cuenta ahorro	Opcion2

Ejercicio

- Identificar tres requerimientos funcionales para el problema del simulador bancario

Nombre	
Resumen	
Entradas	
Resultados	



Nombre	R3 Visualizar el saldo del CDT del cliente.
Resumen	Visualiza el saldo del CDT por el cliente.
Entradas	
Ninguna.	
Resultados	
Se muestra el saldo del CDT del cliente y el interés del mismo.	

Nombre	R4 – Visualizar el saldo total que tiene el cliente en los productos del banco.
Resumen	Visualiza el saldo total que tiene el cliente en los productos del banco.
Entradas	
Ninguna.	
Resultados	
Se muestra el saldo del total del cliente.	

Nombre	R5 – Invertir un monto de dinero en un CDT.
Resumen	Abre un CDT para el cliente dado el monto y un interés.
Entradas	
Monto de dinero que se quiere invertir.	
Interés del CDT.	
Resultados	
Se actualiza la interfaz con el nuevo saldo invertido en CDT y se actualiza el saldo total.	
Si se ingresa un monto de dinero no válido, se muestra un mensaje de error.	

Reto 1 de Java...

Nombre	RN1 - Acelerar simulación
Resumen	Avanza la simulación el número de meses indicado por el usuario.
Entradas	
Número de meses a avanzar en la simulación	
Resultados	
Se actualiza la información con el nuevo saldo de la cuenta de ahorros, el saldo en CDT y el nuevo saldo total.	

Nombre	RN2 -Reiniciar simulación
Resumen	Informa al usuario del interés total generado e inicia una nueva simulación.
Entradas	
Ninguna	
Resultados	
Se muestra el interés total generado por la simulación.	
Se actualiza la información a sus valores por defecto para iniciar una nueva simulación. (Mes: 1, Saldos: \$0)	



Documento de especificación

Modelo del mundo del problema

Modelo del mundo

- ▶ Proceso de observación del problema
- ▶ El objetivo de esta etapa es identificar los elementos que allí aparecen y describirlos de la mejor manera posible
- ▶ Cuatro actividades:
 - ▶ **Identificar** las entidades
 - ▶ **Modelar** las características de las entidades
 - ▶ **Buscar** las relaciones entre las entidades
 - ▶ **Documentar** (reglas, restricciones, etc)



Modelo del mundo:

1. Identificar los elementos que hacen parte del mundo del problema en el simulador bancario
2. Modelar las características de las entidades
3. Buscar las relaciones entre las entidades del problema
4. Documentar las reglas y restricciones del problema



El simulador bancario (Entidades)

- ▶ Se quiere una aplicación que haga la simulación en el tiempo del portafolio bancario de un cliente
- ▶ Un cliente tiene:
 - ▶ Nombre
 - ▶ Número de cédula (identifica la cuenta)
- ▶ Un portafolio tiene:
 - ▶ Una cuenta de ahorro
 - ▶ Una cuenta corriente
 - ▶ Certificado de depósito a término (CDT)
- ▶ Se quiere que el programa permita a una persona simular el manejo de sus productos bancarios:
 - ▶ Hacer las operaciones necesarias sobre los productos que conforman la cuenta
 - ▶ Avanzar mes por mes en el tiempo, para que el cliente pueda ver el resultado de sus movimientos bancarios y el rendimiento de sus inversiones



El simulador bancario (Entidades)

- ▶ Se quiere una aplicación que haga la simulación en el **tiempo** del **portafolio** bancario de un **cliente**
- ▶ Un **cliente** tiene:
 - ▶ **Nombre**
 - ▶ **Número de cédula** (identifica la cuenta)
- ▶ Un **portafolio** tiene:
 - ▶ Una **cuenta de ahorro**
 - ▶ Una **cuenta corriente**
 - ▶ Certificado de depósito a término (**CDT**)
- ▶ Se quiere que el programa permita al **cliente** simular el manejo de sus productos bancarios:
 - ▶ Hacer las operaciones necesarias sobre los productos que conforman la cuenta
 - ▶ Avanzar mes por **mes** en el tiempo, para que el cliente pueda ver el resultado de sus movimientos bancarios y el **rendimiento** de sus inversiones



Modelo del mundo: identificación de entidades

- ▶ Elementos relevantes del mundo que intervienen en el problema
 - ▶ Concretos (persona, vehículo)
 - ▶ Abstractos (cuenta bancaria)
- ▶ Se les da un nombre significativo
- ▶ Suelen ser los sustantivos del problema
- ▶ En POO se les llama CLASES
- ▶ Convención: los nombres de las clases empiezan por mayúscula



El simulador bancario (Entidades)

- ▶ Se quiere una aplicación que haga la simulación en el **tiempo** del **portafolio** bancario de un **cliente**
- ▶ Un **cliente** tiene:
 - ▶ **Nombre**
 - ▶ **Número de cédula** (identifica la cuenta)
- ▶ Un **portafolio** tiene:
 - ▶ Una **cuenta de ahorro**
 - ▶ Una **cuenta corriente**
 - ▶ Certificado de depósito a término (**CDT**)
- ▶ Se quiere que el programa permita al **cliente** simular el manejo de sus productos bancarios:
 - ▶ Hacer las operaciones necesarias sobre los productos que conforman la cuenta
 - ▶ Avanzar mes por **mes** en el tiempo, para que el cliente pueda ver el resultado de sus movimientos bancarios y el **rendimiento** de sus inversiones



Modelo del mundo:

1. Identificar los elementos que hacen parte del mundo del problema en el simulador bancario
2. **Modelar las características de las entidades**
3. Buscar las relaciones entre las entidades del problema
4. Documentar las reglas y restricciones del problema



Modelo del mundo: características de las entidades

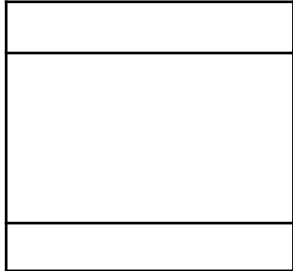
- ▶ A cada característica le debemos asociar
 - ▶ Un nombre significativo
 - ▶ Una descripción del conjunto de valores que dicha característica puede tomar
- ▶ En POO los llamamos atributos
- ▶ Convención: los nombres de los atributos empiezan por minúscula, sin espacios o separadores en blanco



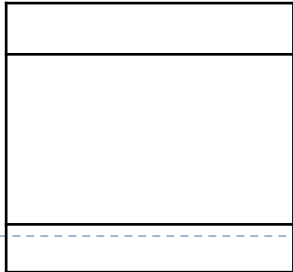
Ejercicio

- Documentar las características de las entidades del simulador bancario

Clase: Portafolio

Atributo	Valores posibles	Diagrama UML
cedula	str	
nombre	str	
mes_actual	int	

Clase: Cuenta corriente

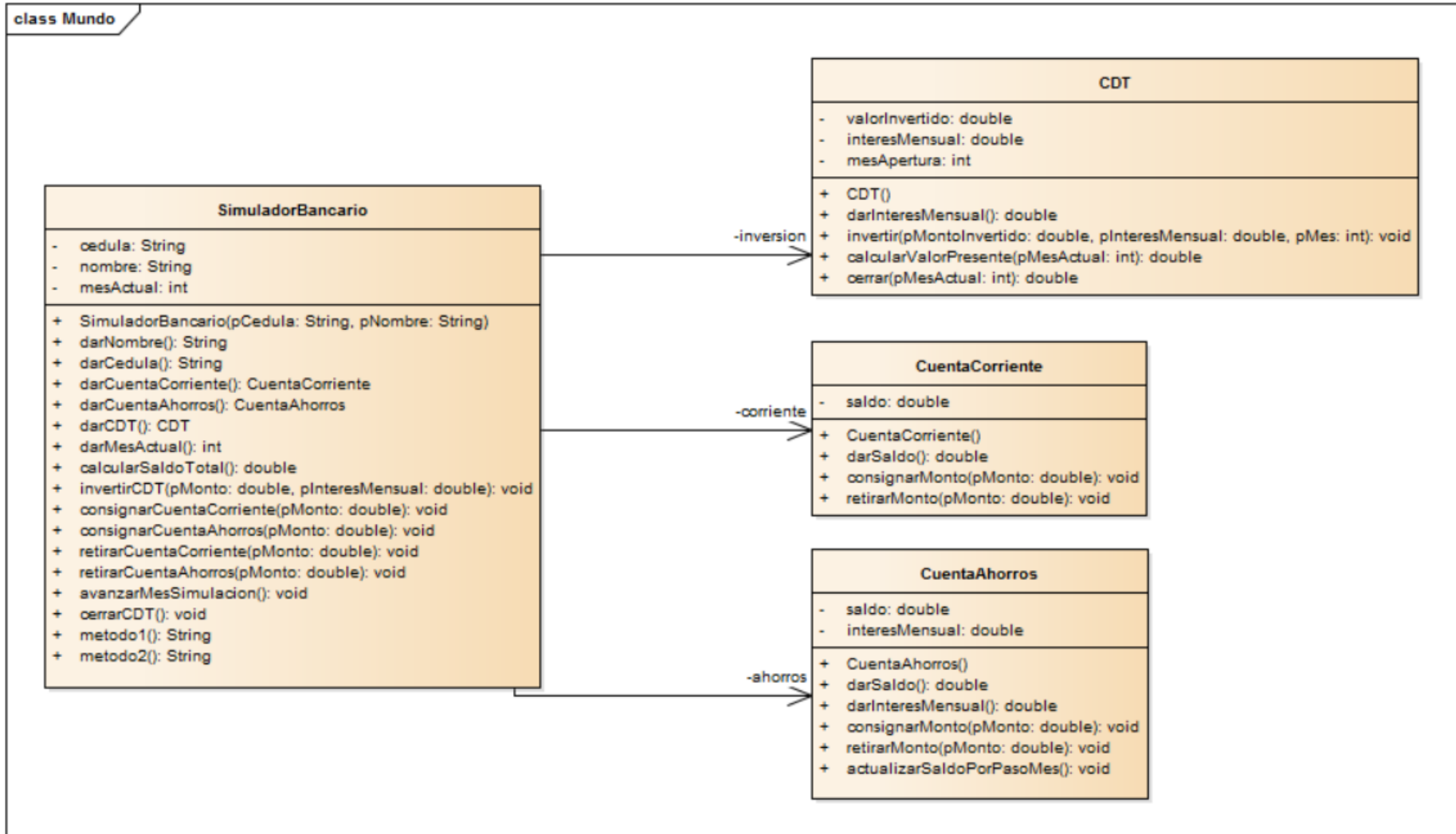
Atributo	Valores posibles	Diagrama UML
saldo	double	

Modelo del mundo:

1. Identificar los elementos que hacen parte del mundo del problema en el simulador bancario
2. Modelar las características de las entidades
3. **Buscar las relaciones entre las entidades del problema**
4. Documentar las reglas y restricciones del problema

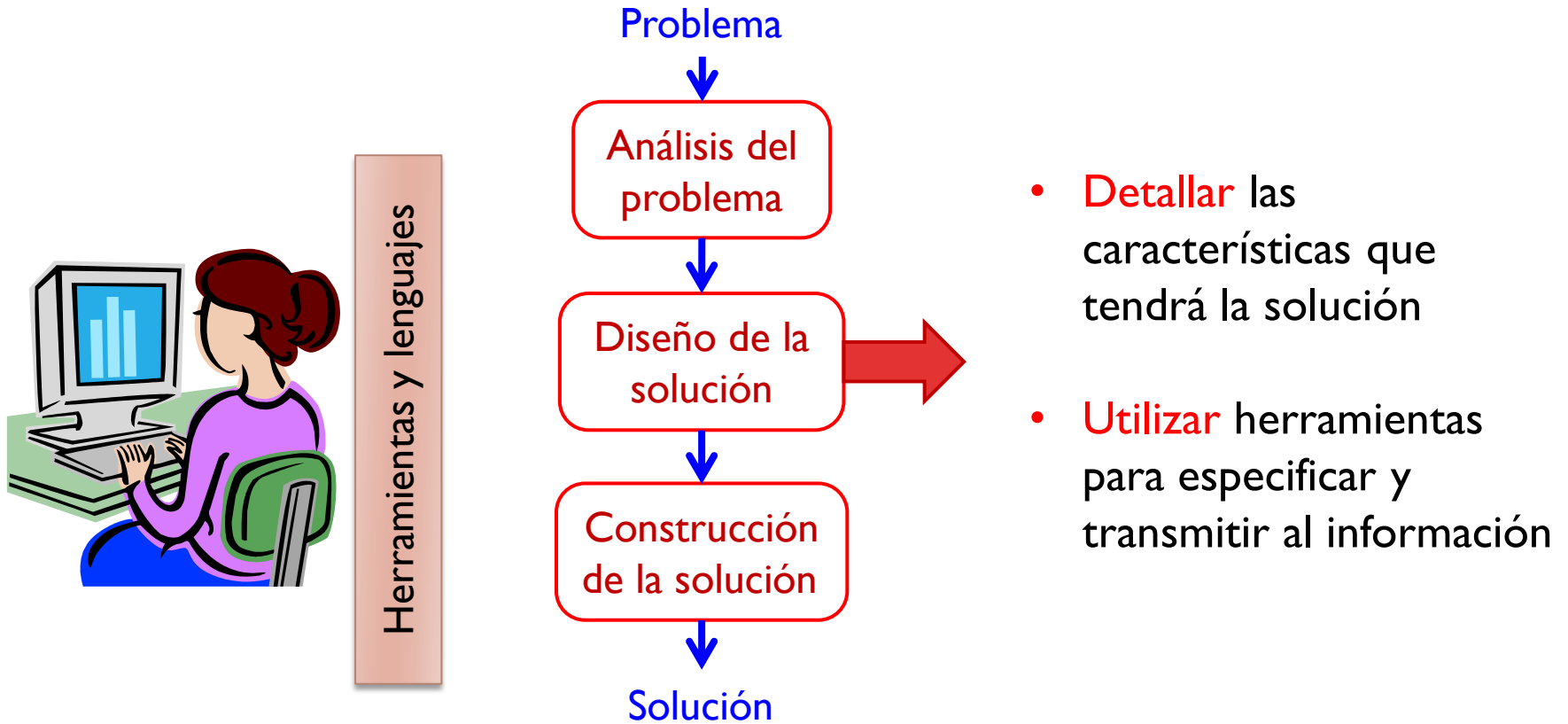


Diagrama de clases del modelo del mundo para el simulador bancario



Etapa 2: Diseño de la solución

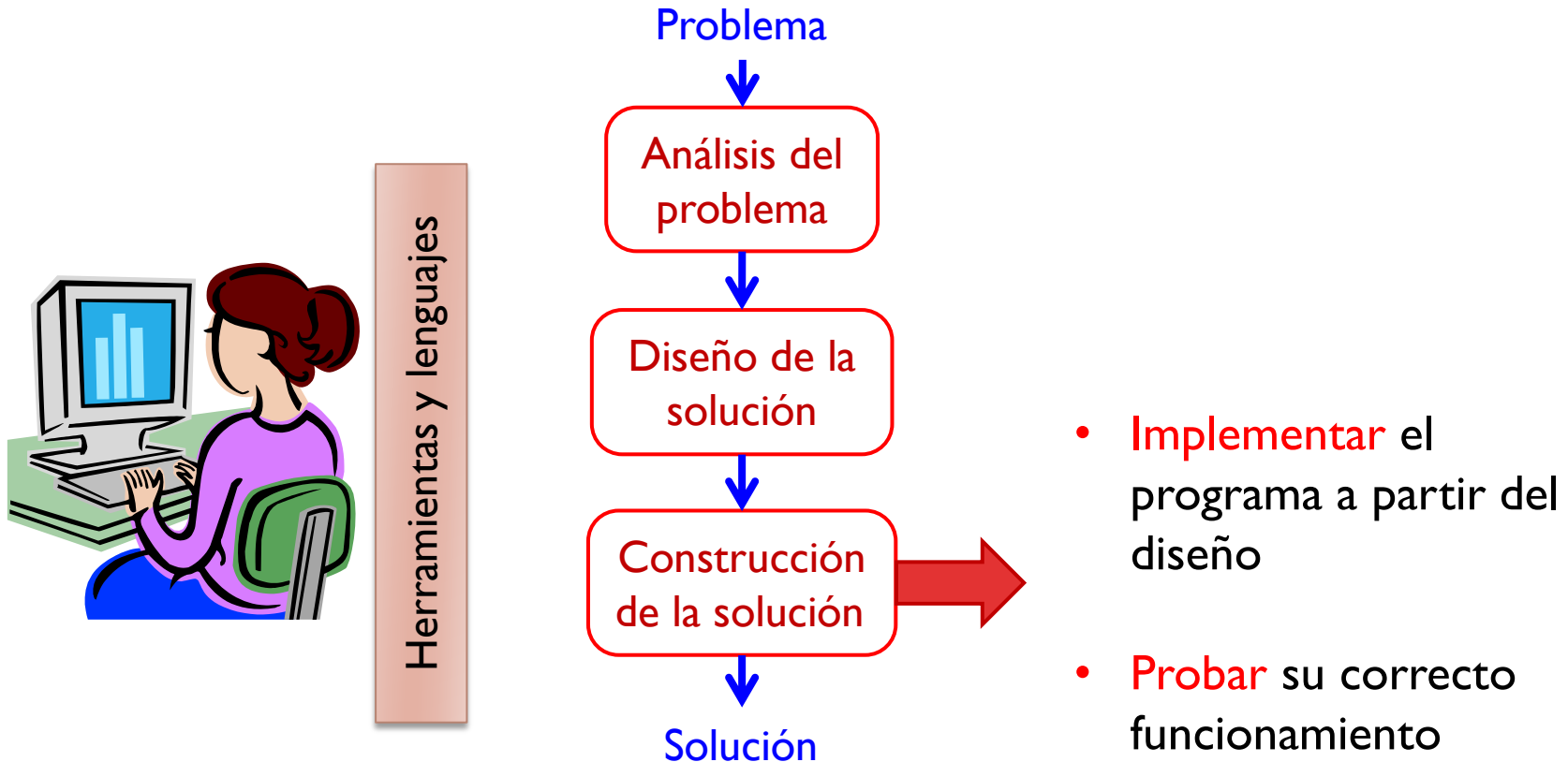
Solucionar un problema (Construir un programa)



Etapa 3: Construcción de la solución

Implementación

Solucionar un problema (Construir un programa)



Algoritmos e instrucciones

Construcción de la solución

Algoritmos e instrucciones

▶ Algoritmo:

- ▶ Secuencia de instrucciones para resolver un problema
- ▶ Secuencia ordenada de pasos para realizar una actividad

▶ Ejemplos

- ▶ Algoritmo para preparar huevos pericos
- ▶ Algoritmo para amarrarse los zapatos
- ▶ Algoritmo para cambiar una llanta
- ▶ Algoritmo para llegar a una dirección dada



Java

- ▶ Un lenguaje de programación de propósito general
- ▶ Un programa Java está conformado por un conjunto de CLASES
- ▶ Cada Clase se guarda en un archivo distinto



Las líneas telefónicas

Caso de estudio #3

Las líneas telefónicas (problema)

- ▶ 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:
 - ▶ Registrar una llamada en alguna de las líneas
 - ▶ 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
 - ▶ 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).
 - ▶ Reiniciar el uso las líneas telefónicas, dejando todos sus valores en cero



Las líneas telefónicas (**solución**)

MiEmpresa - Manejo Líneas Telefónicas


Manejo Líneas Telefónicas


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

Línea #2


\$ 0,00
Número Llamadas: 0
Número de Minutos: 0

Línea #3


\$ 0,00
Número Llamadas: 0
Número de Minutos: 0

Opciones

Las líneas telefónicas: Requerimientos funcionales

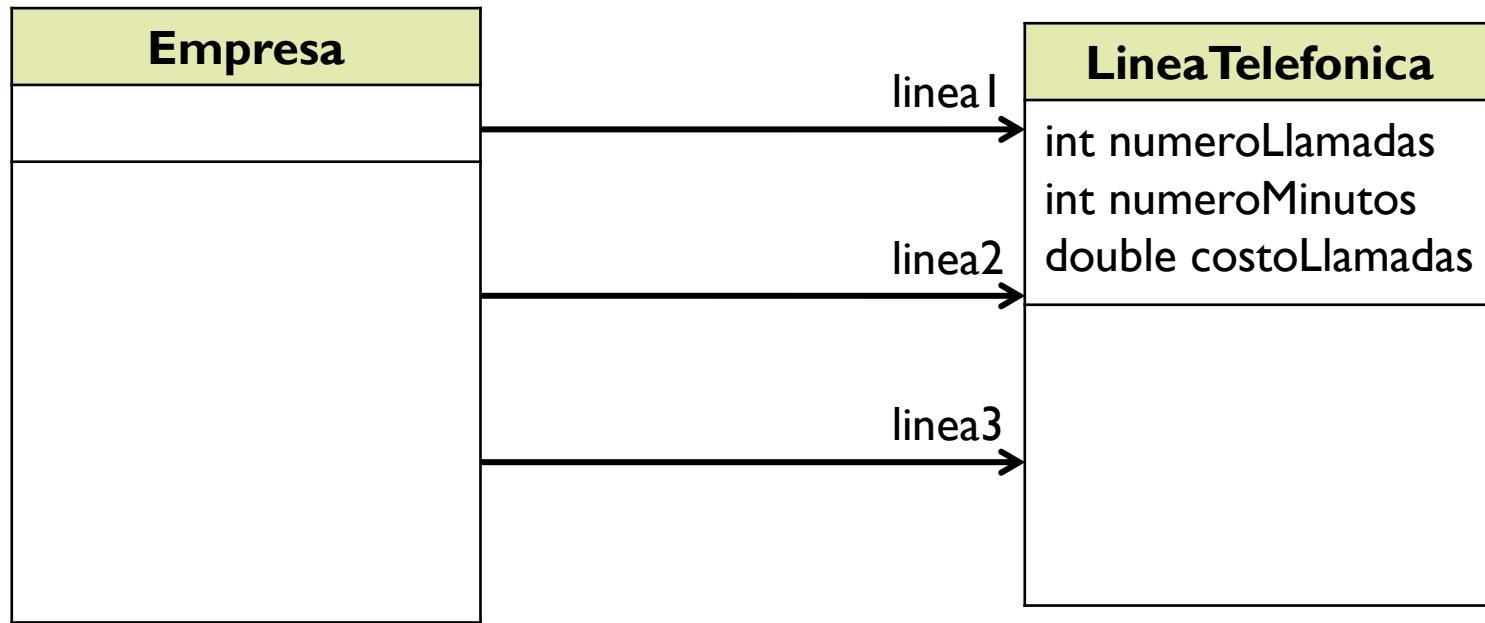
- ▶ R1: Registrar (agregar) una llamada en alguna de las líneas
- ▶ R2: Mostrar la información detallada de cada línea
- ▶ R3: ...
- ▶ R4: ...



R1: Registrar (agregar) una llamada en alguna de las líneas

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 válidas 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.	

Las líneas telefónicas: **Modelo del mundo**



Métodos

- ▶ Son los «algoritmos» de la clase
- ▶ Lo que la clase sabe hacer:
 - ▶ Resolver un problema puntual
 - ▶ Servicio que la clase presta a otras 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 celular



Métodos

- ▶ Cada una de las acciones que sabe hacer una clase es un método



Ejemplo: ¿Qué debe saber hacer una línea telefónica?

► Informar

- El número total de sus llamadas `darCostoLlamadas`
- El costo total de sus llamadas `darNumeroLlamadas`
- La cantidad de minutos consumidos `darNumeroMinutos`

► Agregar

- Una llamada local `agregarLlamadaLocal`
- Una llamada de larga distancia `agregarLlamadaLargaDistancia`
- Una llamada celular `agregarLlamadaCelular`



Métodos en Java

- ▶ Un método tiene dos partes importantes
 - ▶ Signatura
 - ▶ Visibilidad (scope) – siempre public
 - ▶ Tipo de respuesta – Tipo de dato al que pertenece el resultado que va a calcular el método. Si no hay respuesta se indica vacío (void)
 - ▶ Nombre
 - ▶ Parámetros – Conjunto de valores necesarios para resolver el problema
 - ▶ Cuerpo
 - ...
- ```
public void agregarLlamadaLocal(int minutos)
```



# Métodos en Java

---

- ▶ Un método tiene dos partes importantes
  - ▶ Signatura
  - ▶ **Cuerpo**
    - ▶ 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

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



# Instrucciones en Java

Tipos de instrucciones

# Instrucción de **asignación**

---

- ▶ Cambiar/asignar el valor asignado a un atributo
- ▶ Se construye con un igual «**=**»

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



# Instrucción de **asignación**

---

atributo = expresión;

**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:  
atributos, parámetros y valores constantes

Con operadores aritméticos  
+, -, \*, /





# Instrucción de **retorno**

---

- ▶ Para devolver un resultado como solución del problema puntual
- ▶ Utiliza la palabra reservada «**return**»

```
public int darNumeroLlamadas()
{
 return numeroLlamadas;
}
```



# Instrucción de **llamado a otro método**

---

- ▶ Se hace para construir métodos complejos a partir de métodos mas simples que ya están escritos.
  - ▶ Usar métodos de la misma clase
  - ▶ Usar métodos de un objeto de otra clase con la cual existe una asociación
- ▶ Ejemplo: calcular el monto de los impuestos que debe pagar el empleado en un año. Los impuestos se calculan como el 19,5% del total de salarios recibidos en un año
  - ▶ Calcular el valor total del salario anual
  - ▶ Calcular el monto del impuesto usando el método anterior



# Invocación de un método de la **misma** clase

---

| Empleado                                                    |
|-------------------------------------------------------------|
| String nombre<br>String apellido<br>int sexo<br>int salario |
| int calcularSalarioAnual()<br>int calcularImpuesto()        |

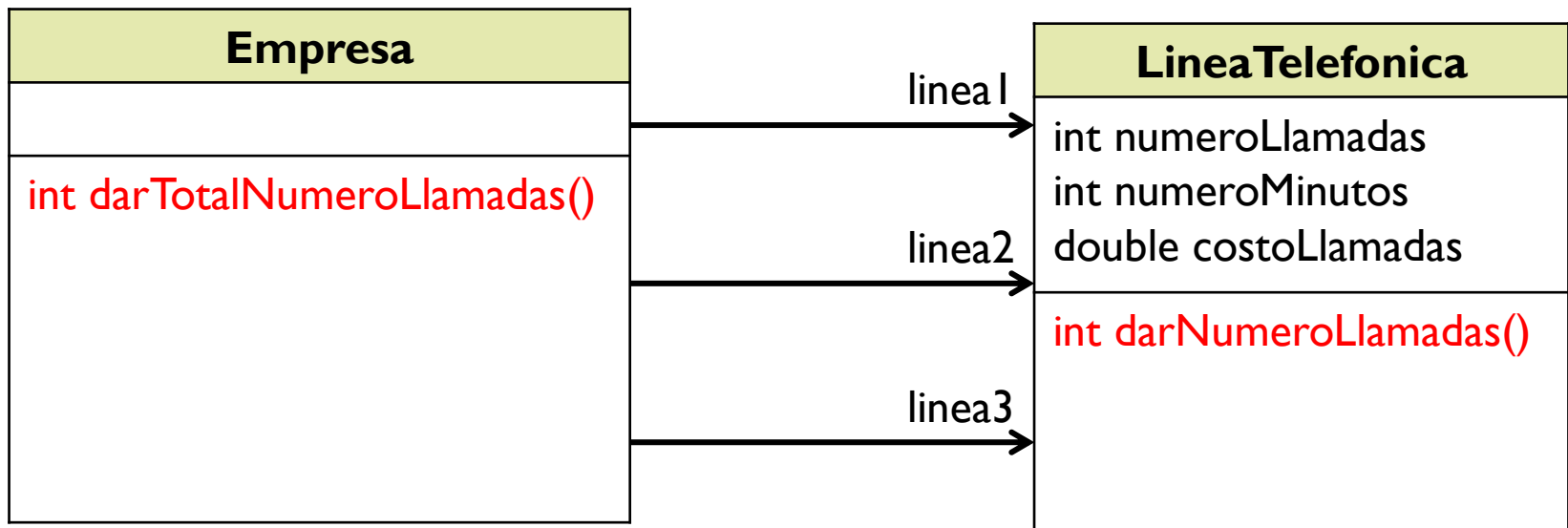
```
public class Empleado{
 ...
 public int calcularSalarioAnual(){
 return (salario * 12);
 }
 public int calcularImpuesto(){
 return (calcularSalarioAnual()*19.5)/100
 }
 ...
}
```



# Invocación de un método de **una asociación**

---

- ▶ Se hace cuando se necesita obtener o modificar alguna información de un objeto de otra clase con el cual existe una asociación



## Invocación de un método de una asociación

```
public class LineaTelefonica{
```

• • •

```
public int darNumeroLlamadas(){
```

```
return _____;
```

}

• • •

}

| LineaTelefonica                                                      |
|----------------------------------------------------------------------|
| <pre>int numeroLlamadas int numeroMinutos double costoLlamadas</pre> |
| <pre>int darNumeroLlamadas()</pre>                                   |



# Invocación de un método de **una asociación**

---

```
public class LineaTelefonica{
```

```
...
```

```
public int darTotalNumeroLlamadas(){
 return linea1.darNumeroLlamadas()+
 linea2.darNumeroLlamadas()+
 linea3.darNumeroLlamadas();
```

```
}
```

```
...
```

```
}
```

| LineaTelefonica                                                 |
|-----------------------------------------------------------------|
| int numeroLlamadas<br>int numeroMinutos<br>double costoLlamadas |
| int darNumeroLlamadas()                                         |



# Llamado de métodos con parámetros

---

- ▶ **¿Cuándo necesita parámetros un método?**
  - ▶ Cuando la información que tiene el objeto en sus atributos no es suficiente para resolver el problema
- ▶ **¿Cómo se declara un parámetro?**
  - ▶ En la signature del método se define el tipo del dato del parámetro y se le asocia un nombre
- ▶ **¿Cómo se utiliza el valor de un parámetro?**
  - ▶ Basta con utilizar el nombre del parámetro en el cuerpo del método de la misma manera que se utilizan los atributos



# Ejemplo llamado de métodos con parámetros

---

```
public class LineaTelefonica{
```

```
...
```

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

```
...
```

```
}
```





# Ejemplo llamado de métodos con parámetros

---

```
public class Empresa{
 ...
 public void agregarLlamadaLocalLinea l (int minutos)
 {
 linea l .agregarLlamadaLocal(minutos);
 }
 ...
}
```



# Instrucción de creación de objetos

---

- ▶ Utiliza la palabra reservada **new**
- ▶ Los objetos de las asociaciones los crea la **clase dueña** de las mismas en alguno de sus métodos
  - ▶ inicializar



# Ver método inicializar del simulador

---

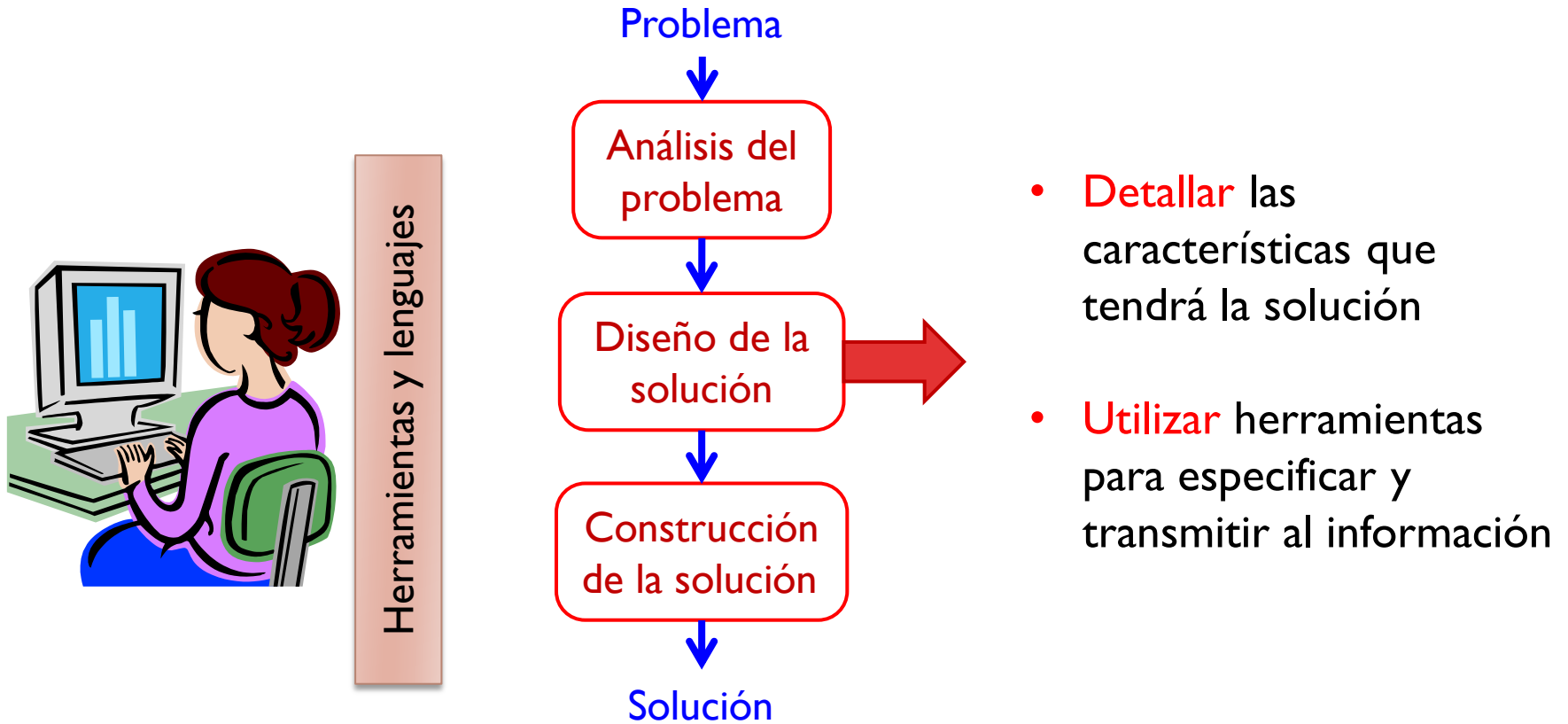
```
public SimuladorBancario(String pCedula, String pNombre)
{
 // Inicializa los atributos personales del cliente
 nombre = pNombre;
 cedula = pCedula;
 // Inicializa el mes en el 1
 mesActual = 1;
 // Inicializa las tres cuentas en vacío
 corriente = new CuentaCorriente();
 ahorros = new CuentaAhorros();
 inversion = new CDT();
}
```



## Etapa 2: Diseño de la solución

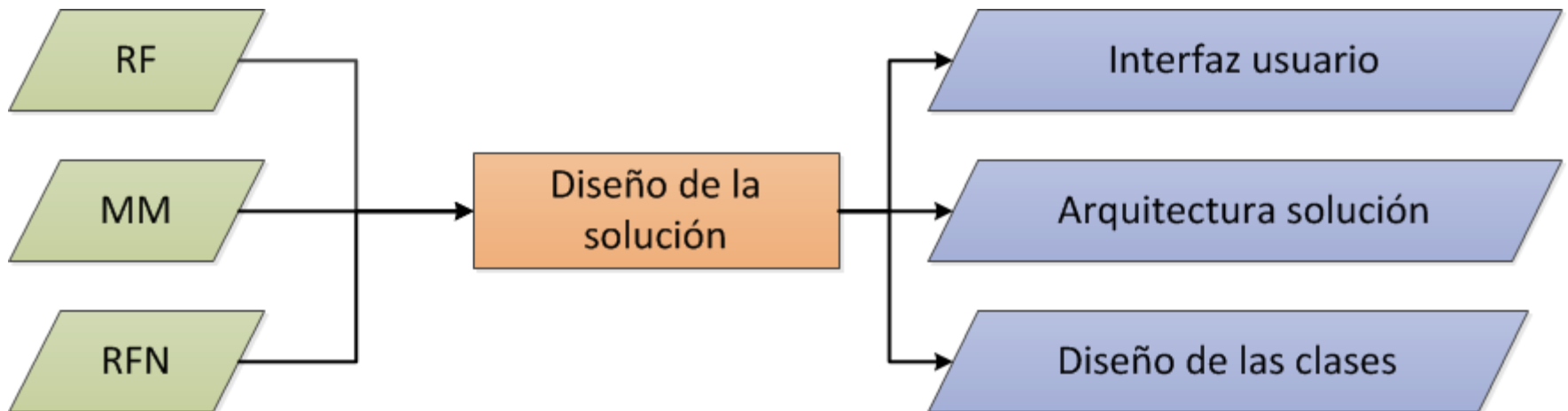
# Solucionar un problema (Construir un programa)

---



# Diseño de la solución

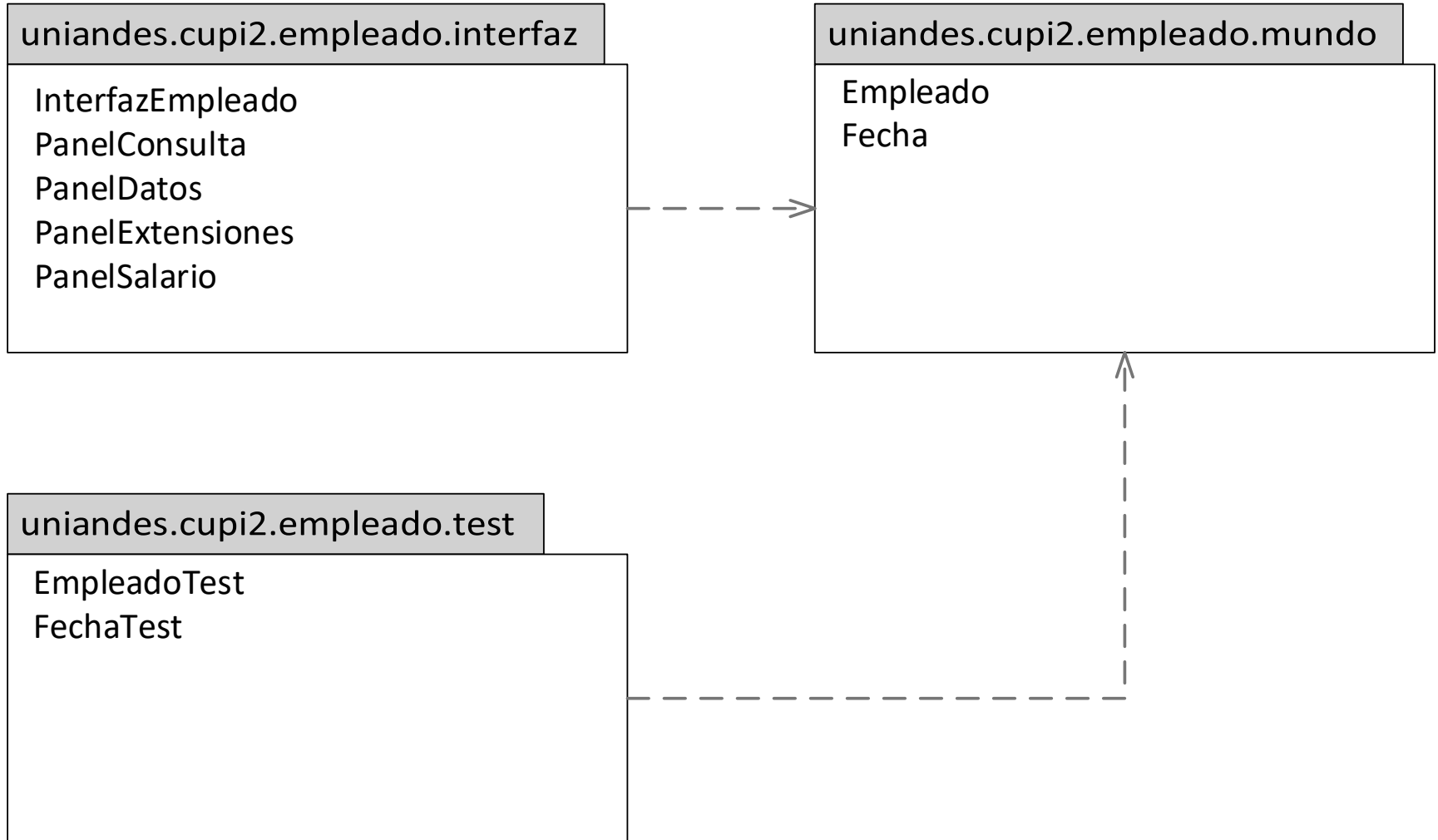
---



# Diseño de la solución: **Interfaz de usuario**



# Diseño de la solución: **Arquitectura de solución**





# Diseño de la solución: Diseño de las clases

---

- ▶ Mostrar los detalles de cada una de las clases que van a hacer parte del programa
  - ▶ Clases
  - ▶ Atributos
  - ▶ Asociaciones
  - ▶ Signaturas de los métodos

