

Fundamentos de Programación

Aprendizaje activo basado en casos

Jorge A. Villalobos / Rubby Casallas G.

Problemas, soluciones y programas

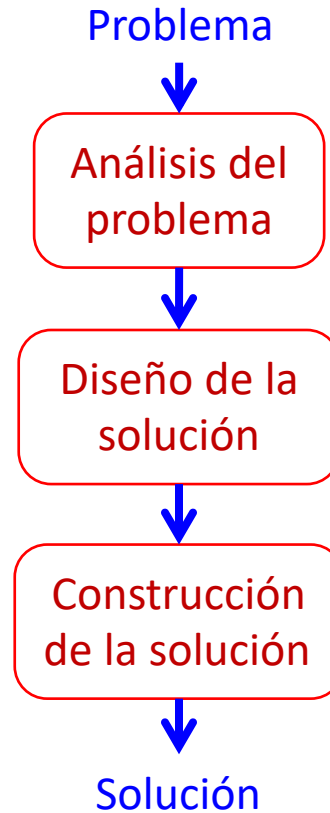
Nivel 1

Algorítmica y programación orientada a objetos I

Solucionar un problema (Construir un programa)



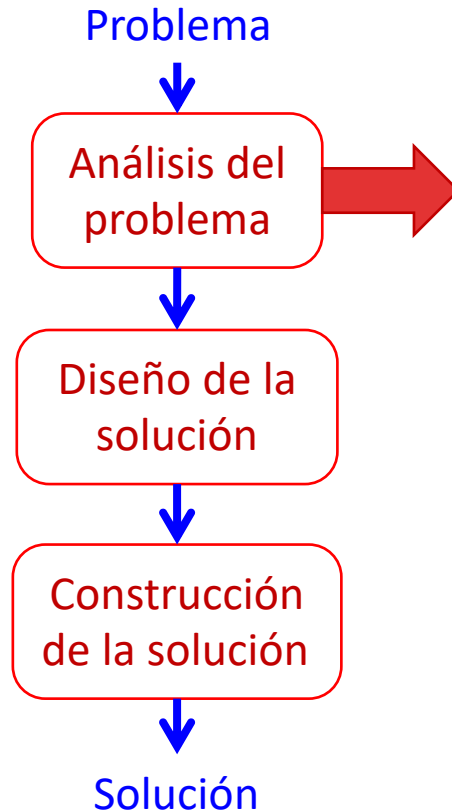
Herramientas y lenguajes



Solucionar un problema (Construir un programa)



Herramientas y lenguajes

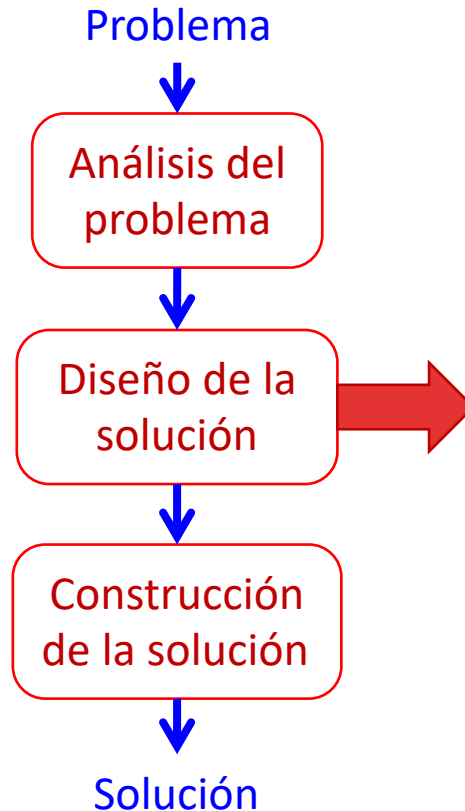


- **Entender** el problema que tiene el cliente
- **Especificar** toda la información que suministre el cliente

Solucionar un problema (Construir un programa)



Herramientas y lenguajes

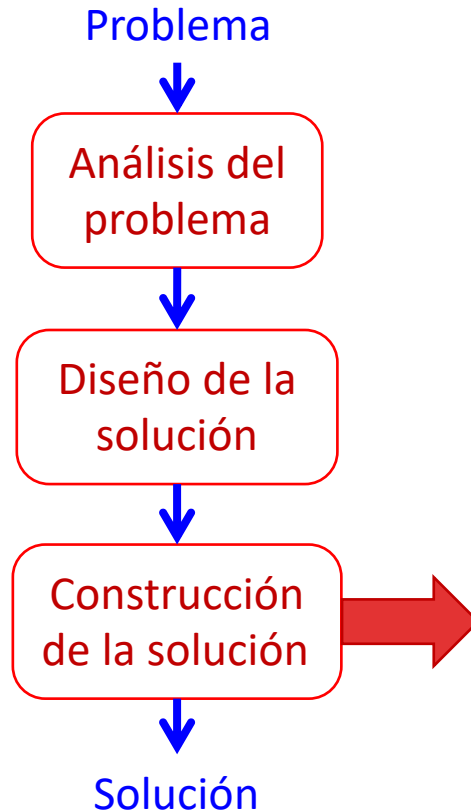


- **Detallar** las características que tendrá la solución
- **Utilizar** herramientas para especificar y transmitir la información

Solucionar un problema (Construir un programa)



Herramientas y lenguajes



- **Implementar** el programa a partir del diseño
- **Probar** su correcto funcionamiento

El empleado

Caso de estudio #1

El empleado (Problema)

- Se quiere una aplicación que permita manejar la información de un empleado
- El empleado tiene:
 - Nombre
 - Apellido
 - Sexo
 - Fecha de nacimiento
 - Imagen asociada
 - Fecha de ingreso a la misma
 - Salario básico asignado
- La aplicación debe permitir:
 - Modificar el salario del empleado
 - Realizar algunos cálculos con la información disponible
 - Edad actual
 - Antigüedad en la empresa
 - Prestaciones a las que tiene derecho. Para el cálculo de las prestaciones se utiliza la fórmula $p = (a * s) / 12$ (p: prestaciones, a: antigüedad, s: salario)

El empleado (Solución)

Sistema de Empleados

Datos Personales

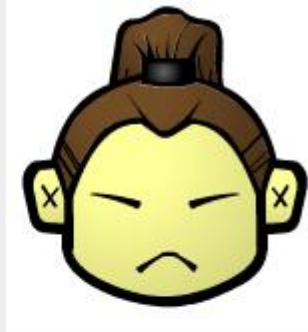
Nombre:

Apellido:

Sexo:

Fecha de Nacimiento:

Fecha de Ingreso:



Salario

Salario:

Cálculos

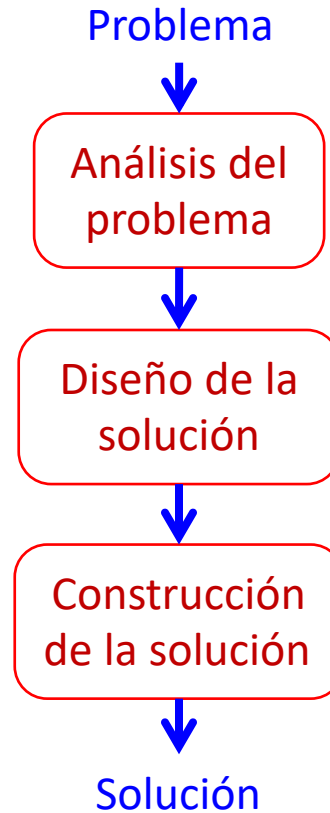
Puntos de Extensión

Etapa 1: Análisis del problema

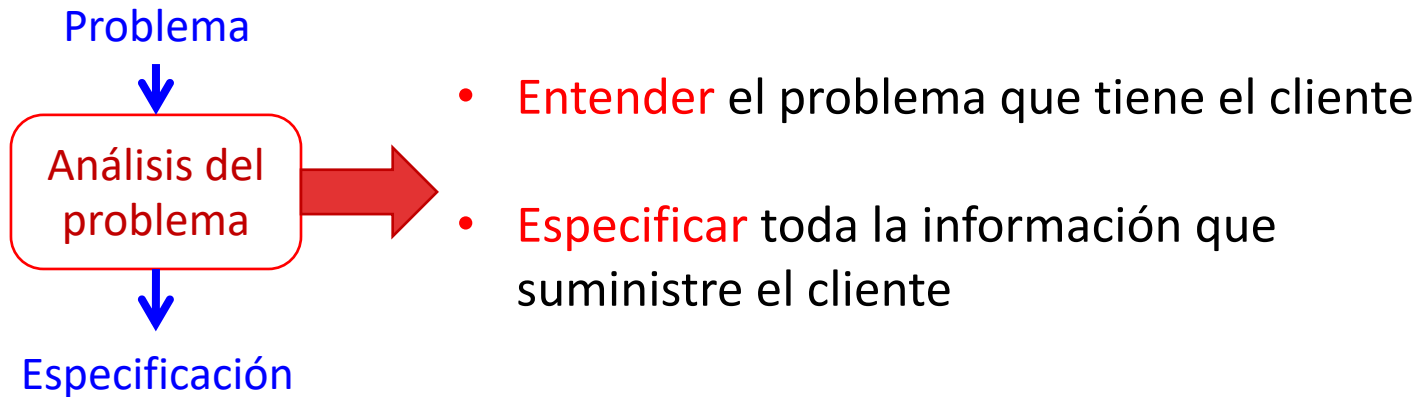
Solucionar un problema (Construir un programa)



Herramientas y lenguajes



1. Análisis del problema



¿Especificación?

Especificación del problema

- Es un **documento** que tiene, entre otros, los siguientes tres elementos
 - Requerimientos funcionales
 - Mundo del problema
 - Requerimientos no funcionales

Especificación – Requerimientos funcionales

- Indican las **necesidades** del cliente
- Establecen las **operaciones** o **servicios** que el programa debe proveer al usuario

¿Qué hace el programa?

El empleado (Problema)

- Se quiere una aplicación que permita manejar la información de un empleado
- El empleado tiene:
 - Nombre
 - Apellido
 - Sexo
 - Fecha de nacimiento
 - Imagen asociada
 - Fecha de ingreso a la misma
 - Salario básico asignado
- La aplicación debe permitir:
 - Modificar el salario del empleado
 - Realizar algunos cálculos con la información disponible
 - Edad actual
 - Antigüedad en la empresa
 - Prestaciones a las que tiene derecho. Para el cálculo de las prestaciones se utiliza la fórmula $p = (a * s) / 12$ (p: prestaciones, a: antigüedad, s: salario)

El empleado (Requerimientos funcionales)

- Se quiere una aplicación que permita manejar la información de un empleado
- El empleado tiene:
 - Nombre
 - Apellido
 - Sexo
 - Fecha de nacimiento
 - Imagen asociada
 - Fecha de ingreso a la misma
 - Salario básico asignado
- La aplicación debe permitir:
 - Modificar el salario del empleado
 - Realizar algunos cálculos con la información disponible
 - Edad actual
 - Antigüedad en la empresa
 - Prestaciones a las que tiene derecho. Para el cálculo de las prestaciones se utiliza la fórmula $p = (a * s) / 12$ (p: prestaciones, a: antigüedad, s: salario)

El empleado (Requerimientos funcionales)

- R1: Modificar el salario del empleado
- R2: Calcular la edad actual del empleado
- R3: Calcular la antigüedad en la empresa
- R4: Calcular las prestaciones de empleado

Especificación – Mundo del problema

- **Contexto** en el que ocurre el problema
- Elementos (**datos**, **información**) que intervienen en el problema

¿Cuál es el ambiente del programa?

El empleado (Problema)

- Se quiere una aplicación que permita manejar la información de un empleado
- El empleado tiene:
 - Nombre
 - Apellido
 - Sexo
 - Fecha de nacimiento
 - Imagen asociada
 - Fecha de ingreso a la misma
 - Salario básico asignado
- La aplicación debe permitir:
 - Modificar el salario del empleado
 - Realizar algunos cálculos con la información disponible
 - Edad actual
 - Antigüedad en la empresa
 - Prestaciones a las que tiene derecho. Para el cálculo de las prestaciones se utiliza la fórmula $p = (a * s) / 12$ (p: prestaciones, a: antigüedad, s: salario)

El empleado (Mundo del problema)

- Se quiere una aplicación que permita manejar la información de un empleado
- El empleado tiene:
 - Nombre
 - Apellido
 - Sexo
 - Fecha de nacimiento
 - Imagen asociada
 - Fecha de ingreso a la misma
 - Salario básico asignado
- La aplicación debe permitir:
 - Modificar el salario del empleado
 - Realizar algunos cálculos con la información disponible
 - Edad actual
 - Antigüedad en la empresa
 - Prestaciones a las que tiene derecho. Para el cálculo de las prestaciones se utiliza la fórmula $p = (a * s) / 12$ (p: prestaciones, a: antigüedad, s: salario)

Especificación – Requerimientos no funcionales

- **Restricciones** o condiciones que impone el cliente al programa
- Ejemplos:
 - Tiempo de entrega del programa
 - Número de usuarios simultáneos
 - Tiempo de ejecución del programa

¿Cómo funciona el programa?

Documento de especificación

Requerimientos funcionales

Especificación de requerimientos funcionales

- Se describen a través de 4 elementos:
 - **Identificador** y **nombre**
 - **Resumen** de la operación
 - **Entradas** que debe dar el **usuario** para que el programa pueda realizar la operación
 - **Resultado** de la operación
 - **Modificación** de un valor en el mundo del problema
 - **Cálculo** de un valor
 - **Mezcla** de los dos anteriores

Ejemplo requerimiento funcional

Nombre	R1 – Actualizar el salario básico del empleado
Resumen	Permite la modificación del salario básico del empleado
Entradas	
Nuevo salario	
Resultados	
Se modificó el salario básico del empleado	

Artefacto: requerimiento funcional

Nombre	R2: Cambiar el empleado
Resumen	Permite al usuario cambiar la información del empleado: datos personales y datos de vinculación a la empresa.
Entradas	
1) Nombre del empleado. 2) Apellido del empleado. 3) Género del empleado. 4) Fecha de nacimiento. 5) Fecha de ingreso a la compañía. 6) Salario básico. 6) Imagen del empleado	
Resultados	
La información del empleado ha sido actualizada.	

Artefacto: requerimiento funcional

Nombre	R3
Resumen	
Entradas	
Resultados	

Ejemplo requerimiento funcional

Nombre	R4 – Calcular las prestaciones del empleado
Resumen	Calcula las prestaciones del empleado
Entradas	
Resultados	

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 software window titled "Simulador Bancario" with standard Windows window controls (minimize, maximize, close). The window 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	

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

El empleado

- Características de un **empleado**

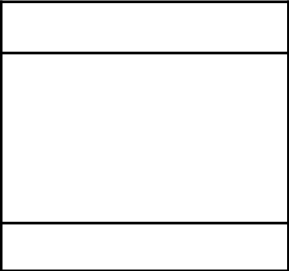
Atributo	Valores posibles
nombre	Cadena de letras
apellido	Cadena de letras
sexo	Masculino o femenino
salario	Valores enteros positivos

¡Usamos UML!
(Unified Modeling Language)

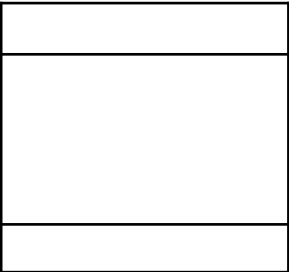
Ejercicio

- Documentar las características de las entidades del simulador bancario

Clase: Portafolio

Atributo	Valores posibles	Diagrama UML
		

Clase: Cuenta corriente

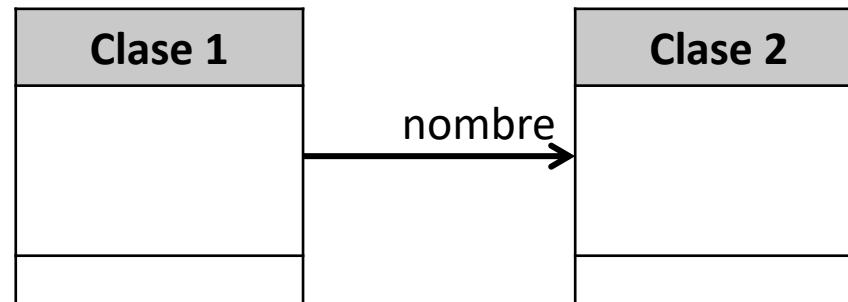
Atributo	Valores posibles	Diagrama UML
		

Modelo del mundo:

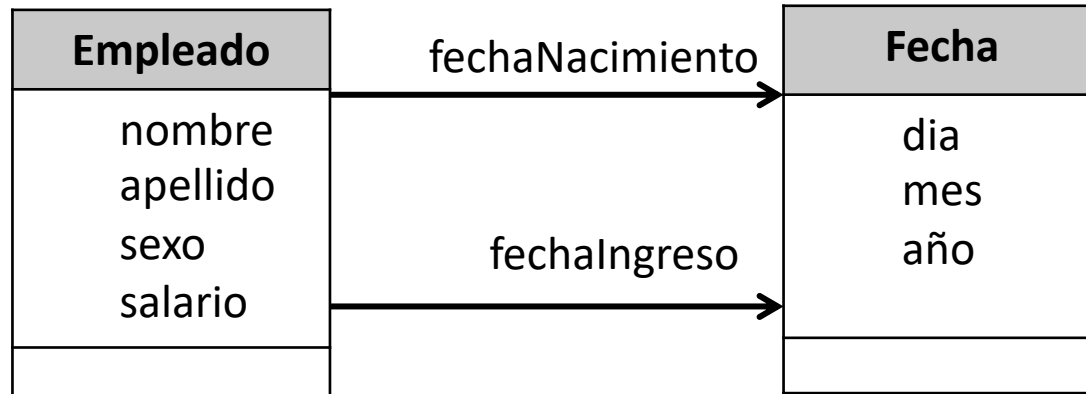
1. Identificar los elementos que hacen parte del mundo del problema en el simulador bancario
2. Modelar las características de las entidades
3. **Buscar las relaciones entre las entidades del problema**
4. Documentar las reglas y restricciones del problema

Modelo del mundo: relaciones entre entidades

- Identificar las relaciones que existen entre las distintas entidades del mundo (clases)
- Dar un nombre a cada relación
- En POO las llamamos asociaciones

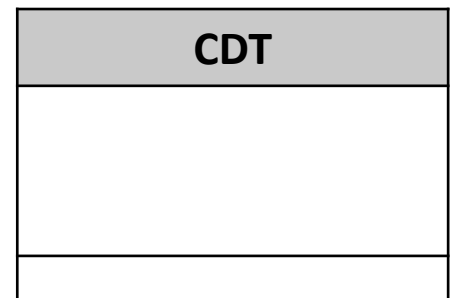
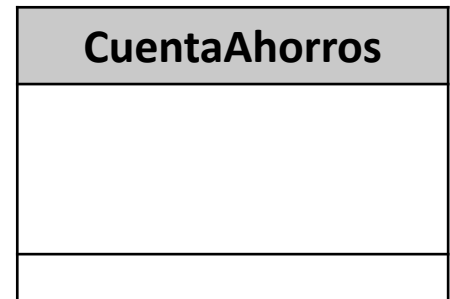
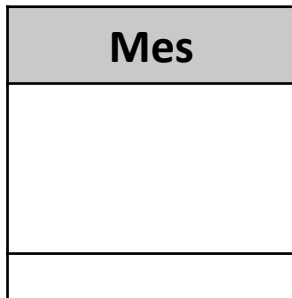
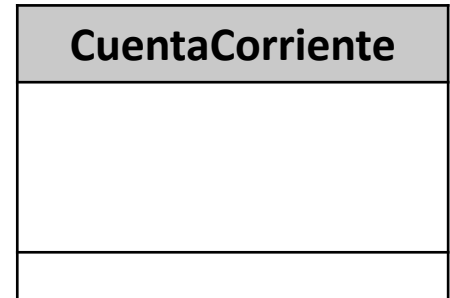
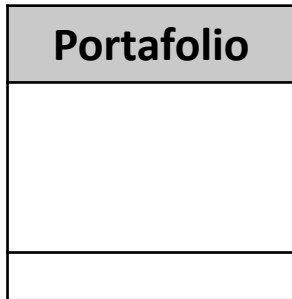


Ejemplo sobre El empleado



- El empleado **tiene** una fecha de nacimiento
- El empleado **tiene** una fecha de ingreso a la empresa
- La fecha es una entidad del mundo representada por la clase **Fecha**

Diagrama de clases del modelo del mundo para el simulador bancario

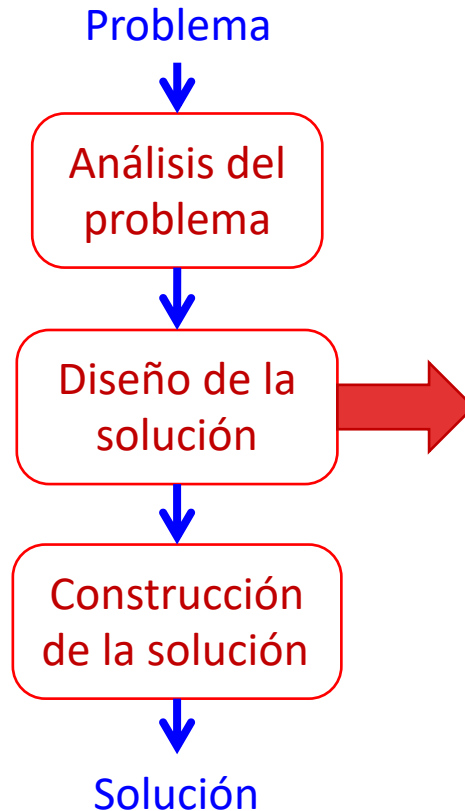


Etapa 2: Diseño de la solución

Solucionar un problema (Construir un programa)



Herramientas y lenguajes



- **Detallar** las características que tendrá la solución
- **Utilizar** herramientas para especificar y transmitir la información

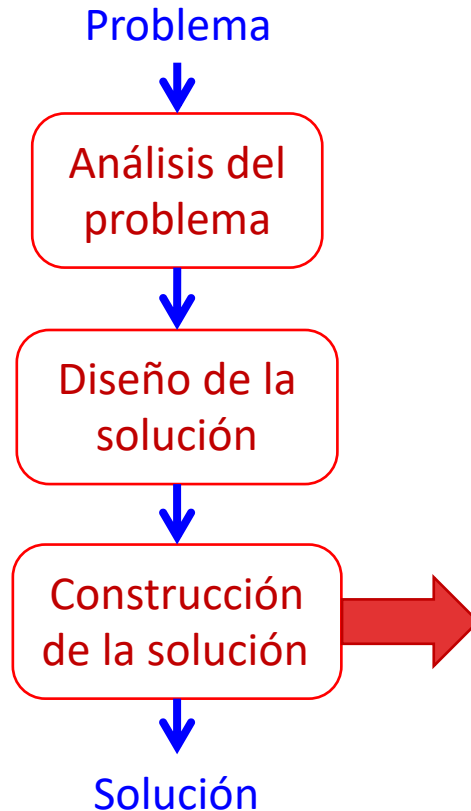
Etapa 3: Construcción de la solución

Implementación

Solucionar un problema (Construir un programa)



Herramientas y lenguajes



- **Implementar** el programa a partir del diseño
- **Probar** su correcto funcionamiento

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

Algoritmos en el computador

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

Lenguaje de programación

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

Declaración de clases en Java

- Una Clase por archivo y un archivo por Clase

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{  
    //Atributos  
    private int dia;  
    private int mes;  
    private int año;  
}
```

Tipos de datos en Java

- Un atributo tiene un tipo
- En Java el tipo de un atributo antecede al nombre del atributo
- Antes del tipo tiene que estar la palabra reservada «**private**»

`private tipo atributo;`

Tipos de datos en Java

- Existen varios tipos de datos:
- Enteros
 - `int`
- Reales
 - `double`
- Cadenas de letras
 - `String`

Asociaciones entre clases en Java

- Toda asociación tiene un nombre
- En Java el nombre de la relación es antecedido con el nombre de la clase relacionada
- Antes del nombre de la clase relacionada debe estar la palabra reservada «**private**»

```
private ClaseRelacionada nombreRelacion;
```

Construcción de la solución

- Ya vimos cómo establecer nuestro modelo del mundo en un programa Java

¿Y los algoritmos?

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

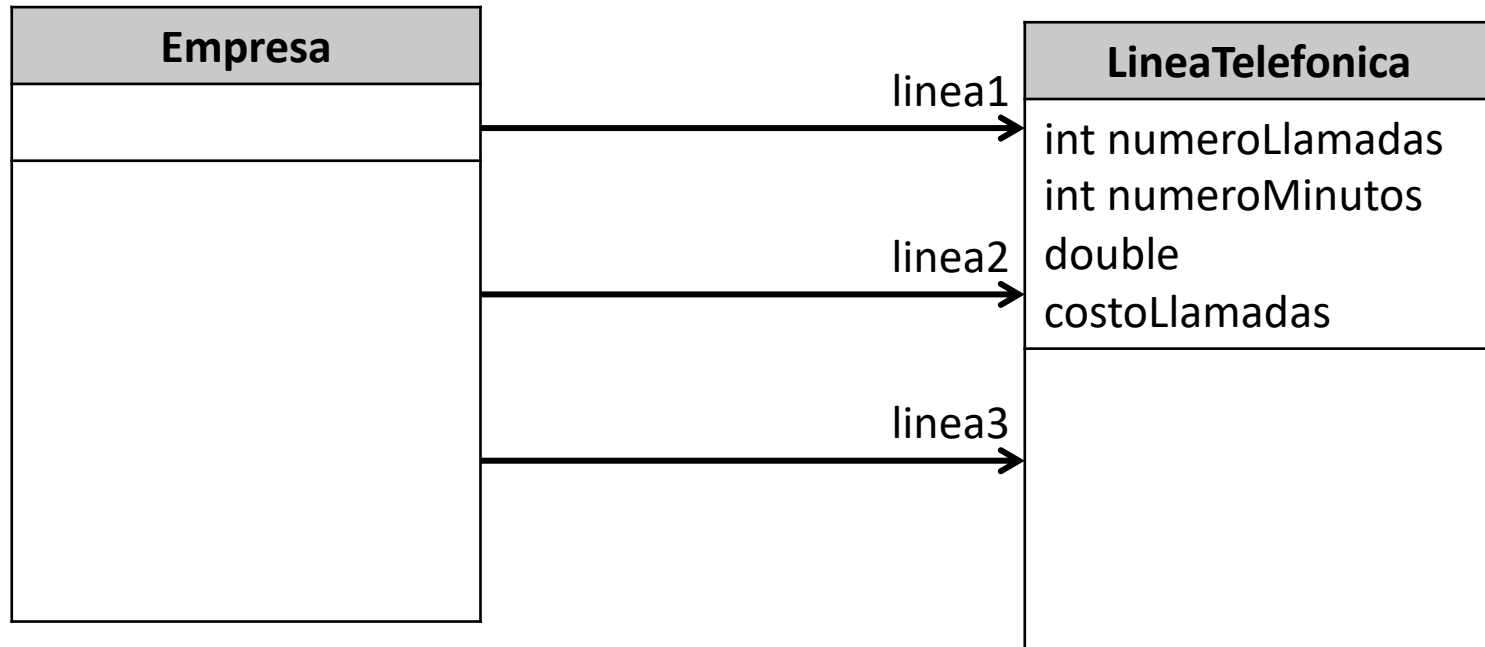
Las líneas telefónicas: Requerimientos funcionales

- 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
- El costo total de sus llamadas
- La cantidad de minutos consumidos

darCostoLlamadas
darNumeroLlamadas
darNumeroMinutos

- Agregar

- Una llamada local
- Una llamada de larga distancia
- Una llamada celular

agregarLlamadaLocal
agregarLlamadaLargaDistancia
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 «=»

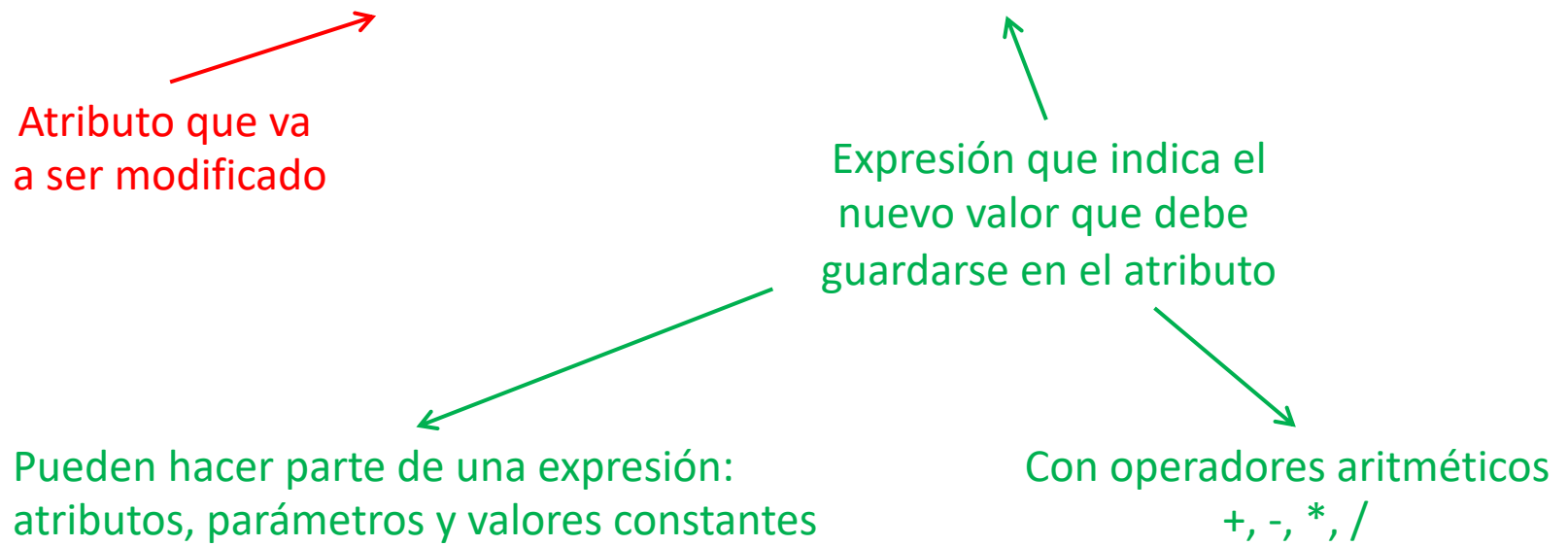
```
public void agregarLlamadaLocal(int minutos)
{
    numeroLlamadas = numeroLlamadas + 1;
    numeroMinutos = numeroMinutos + minutos;
    costoLlamadas = costoLlamadas + ( minutos * 35 );
}
```

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 más 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 String apellido int sexo int salario
int calcularSalarioAnual() int calcularImpuesto()

```
public class Empleado{
```

```
...
```

```
    public int calcularSalarioAnual(){
```

```
        return (salario * 12);
```

```
    }
```

```
    public int calcularImpuesto(){
```

```
        return (calcularSalarioAnual()*19.5)/100
```

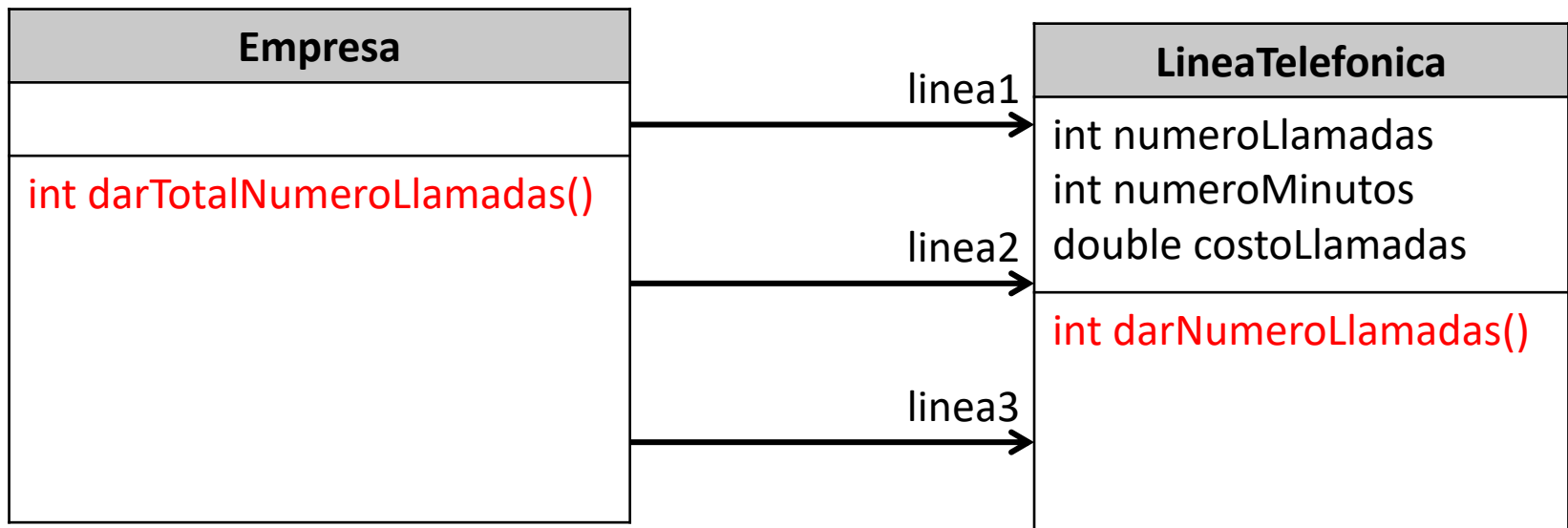
```
    }
```

```
...
```

```
}
```

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
int numeroLlamadas int numeroMinutos double costoLlamadas
int darNumeroLlamadas()

```
int darNumeroLlamadas()
```

Invocación de un método de una asociación

```
public class LineaTelefonica{
```

```
...
```

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

```
...
```

```
}
```

LineaTelefonica
int numeroLlamadas int numeroMinutos 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 signatura 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 agregarLlamadaLocalLinea1( int minutos )
```

```
    {
```

```
        linea1.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

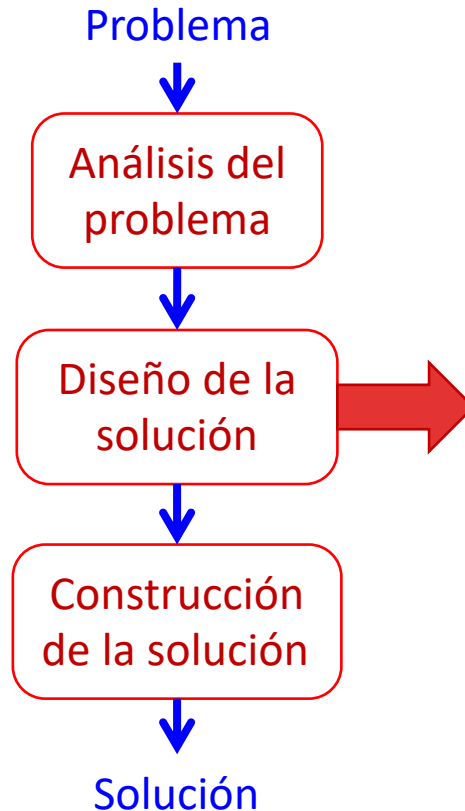
Ver método inicializar de la clase telefónica

Etapa 2: Diseño de la solución

Solucionar un problema (Construir un programa)

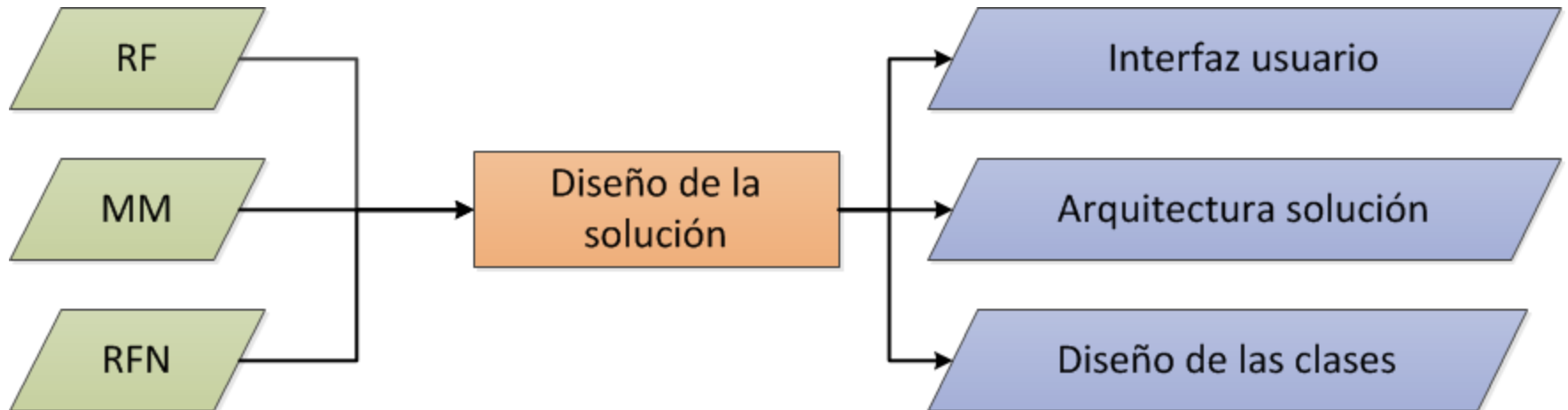


Herramientas y lenguajes



- **Detallar** las características que tendrá la solución
- **Utilizar** herramientas para especificar y transmitir al información

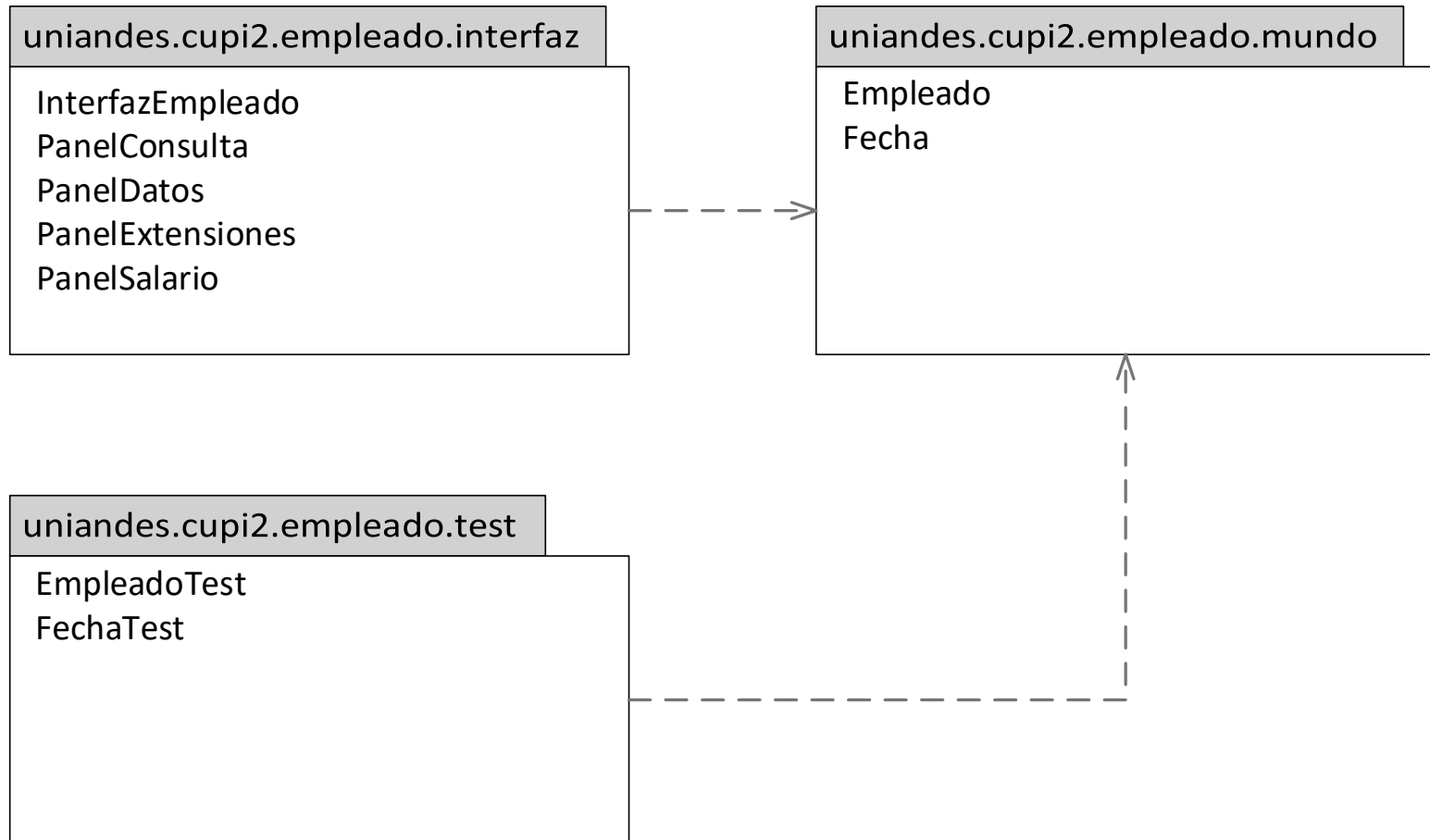
Diseño de la solución



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