Programación Avanzada

Introducción al Paradigma de Orientación a Objetos

Orientación a Objetos

- Enfoque diferente al tradicional
- Puede ser entendida como:
  - Una forma de pensar basada en abstracciones de conceptos existentes en el mundo real
  - Organizar el software como una colaboración de objetos que interactúan entre sí por medio de mensajes

Enfoque Tradicional

- Una aplicación implementada con un enfoque tradicional presenta la siguiente estructura general:

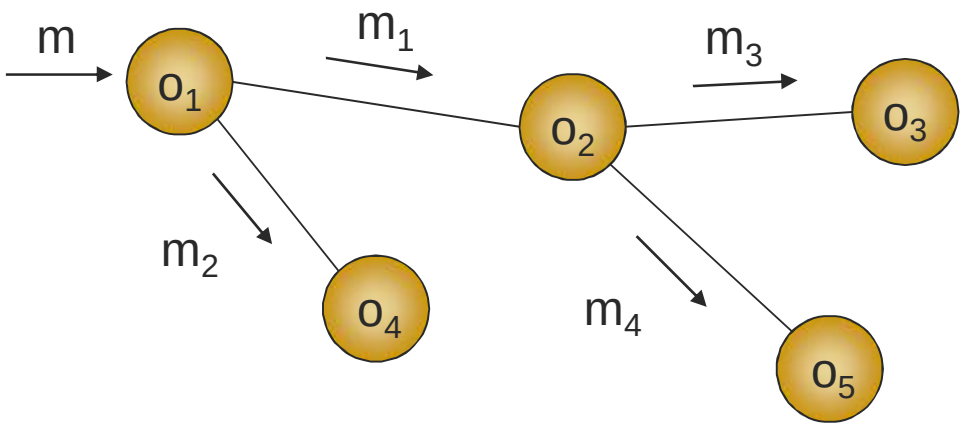
```
type T = …

f1(T t) {…}
…
fn() {…}

main() {
      //invocaciones a fi
}
```

Enfoque Orientado a Objetos

- Una aplicación orientada a objetos es el resultado de la codificación en un lenguaje de programación orientado a objetos del siguiente esquema:



Desarrollo Orientado a Objetos

- Los pasos generales de desarrollo se mantienen en el enfoque orientado a objetos
- Pero las actividades que constituyen algunos de ellos son particulares:
  - Análisis ð Análisis Orientado a Objetos
  - Diseño ð Diseño Orientado a Objetos
  - Implem. ð Implem. Orientada a Objetos

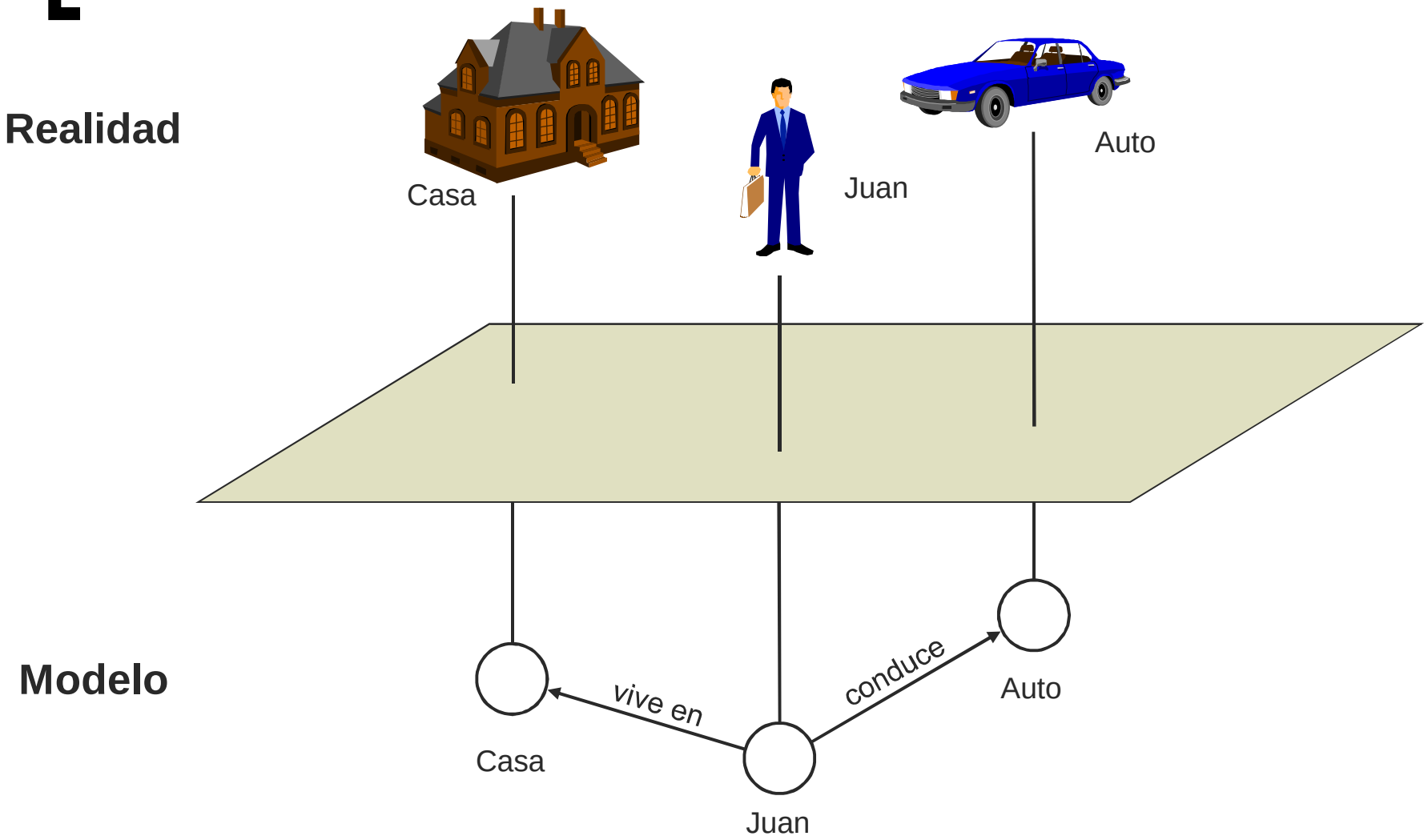
Análisis Orientado a Objetos

- Considerar el dominio de la aplicación y su solución lógica en términos de conceptos (cosas, entidades)
- Concepto clave: *abstracción*
- Objetivo: encontrar y describir los conceptos en el dominio de la aplicación:
  - Esto permite comprender mejor la realidad y el problema

Análisis Orientado a Objetos (2)

- Estos conceptos pueden entenderse como una primera aproximación a la solución al problema
- En un sistema de software orientado a objetos (bien modelado) existe un *isomorfismo* entre estos conceptos y los elementos que participan en el problema en la vida real

Análisis Orientado a Objetos (3)



Diseño Orientado a Objetos

- Objetivo: definir objetos lógicos (de software) y la forma de comunicación entre ellos para una posterior programación
- En base a los “conceptos candidatos” encontrados durante el análisis y por medio de ciertos principios y técnicas, se debe decidir:
  - Cuáles de éstos serán los objetos que participarán en la solución
  - Cómo se comunican entre ellos para obtener el resultado deseado

Diseño Orientado a Objetos (2)

- Concepto clave: *responsabilidades*
- En esta transición:
  - No todos los conceptos necesariamente participarán de la solución
  - Puede ser necesario “reflotar” conceptos inicialmente dejados de lado
  - Será necesario fabricar “ayudantes” (también objetos) para que los objetos puedan llevar a cabo su tarea

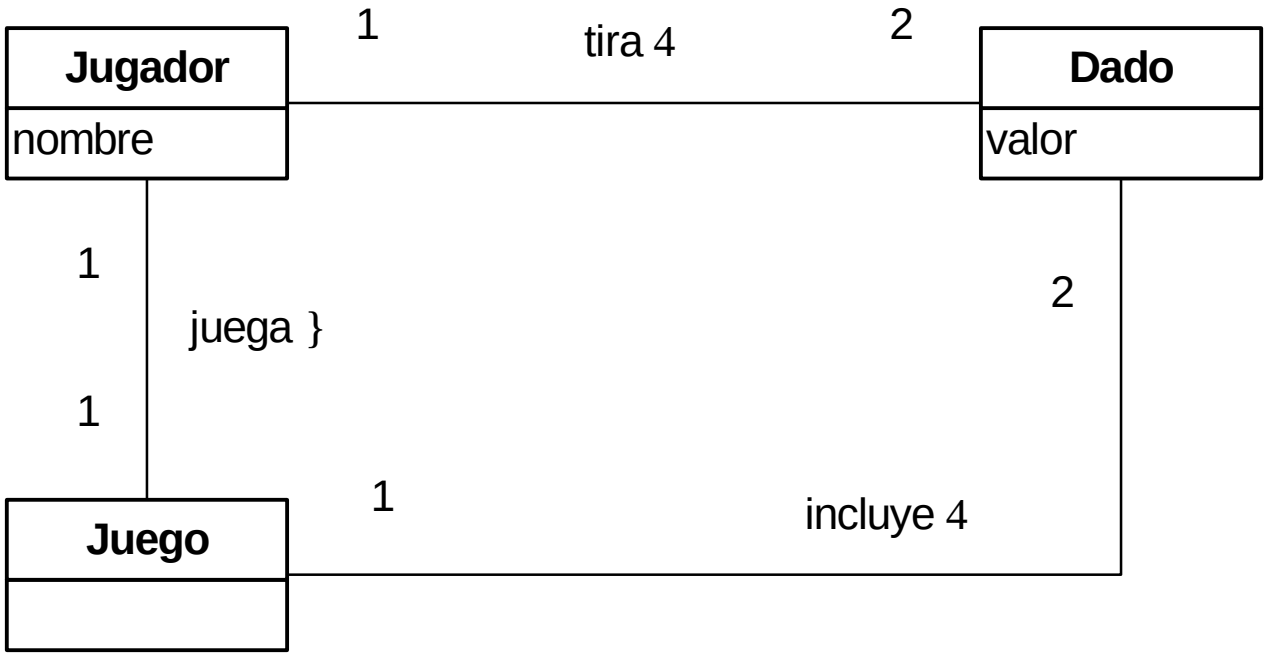
Implementación OO

- Objetivo: codificar en un lenguaje de programación orientado a objetos las construcciones definidas en el diseño
- La definición de los objetos y el intercambio de mensajes requieren construcciones particulares en el lenguaje a utilizar

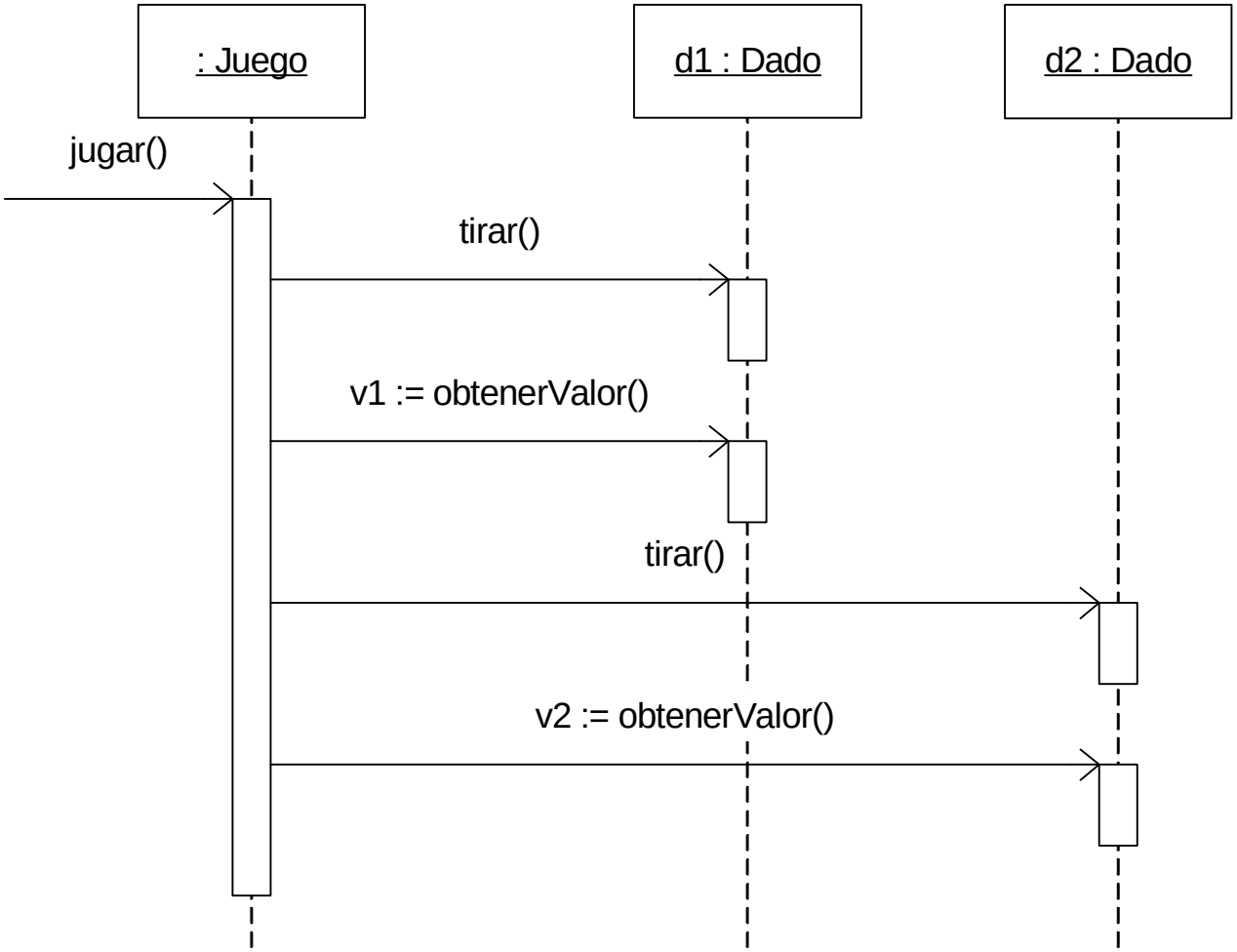
Ejemplo – Juego de Dados

- Problema sencillo para ilustrar conceptos claves
- Actividades a realizar:
  - Modelado del dominio
  - Definición de interacciones
  - Definición de estructura
  - Codificación

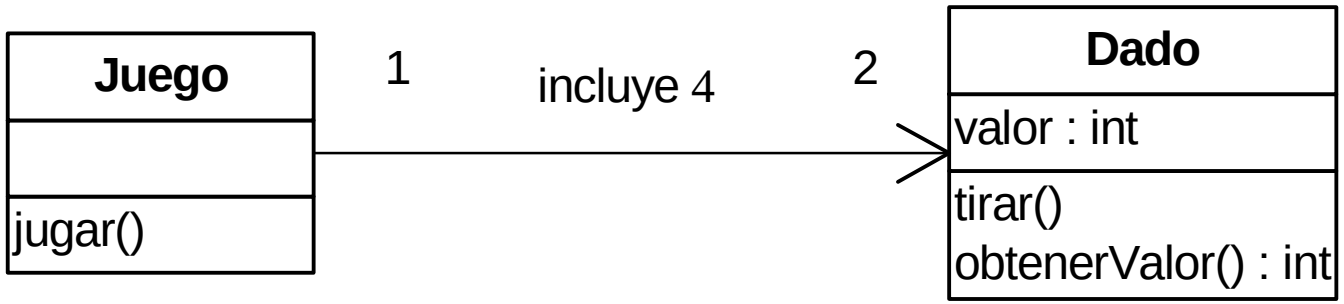
Ejemplo – Dominio



Ejemplo – Interacciones



Ejemplo – Estructura



Ejemplo – Codificación

```
// Dado.h
class Dado {
   public:
       int obtenerValor();
       void tirar();
   private:
       int valor;
};
```

```
// Dado.cpp
#include “Dado.h”
int Dado::obtenerValor() {
   return valor;
}
void Dado::tirar() {
   valor = ... //random
}
```

Ejemplo – Codificación

```
// Juego.h
class Juego {
   public:
        void jugar();
   private:
        Dado* d1, d2;
};
```

```
// Juego.cpp
#include “Juego.h”
void Juego::jugar() {
      int v1,v2;
      ...
      d1->tirar();
      v1 = d1->obtenerValor();
      d2->tirar();
      v2 = d2->obtenerValor();
      ... // algo con v1 y v2
}
```