Unidad 4. Diseño orientado a objetos.

Entornos de desarrollo
1º Desarrollo de Aplicaciones Multiplataforma

Introducción a la orientación a objetos

La **construcción de software** es un proceso cuyo objetivo es dar **solución a problemas utilizando una herramienta informática** y tiene como **resultado la construcción de un programa informático**.

Como en cualquier otra disciplina en la que se obtenga un **producto final de cierta complejidad**, si queremos obtener un producto de **calidad**, es preciso realizar un **proceso previo de análisis** y **especificación del proceso que vamos a seguir,** y de los **resultados que pretendemos conseguir**.

Introducción a la orientación a objetos

Enfoque estructurado

En un principio se tomaba el **problema de partida** y se iba sometiendo a un proceso de **división en subproblemas** más pequeños reiteradas veces, hasta que se **llegaba a problemas elementales** que se podía resolver utilizando una **función**.

Luego las **funciones se hilaban y entretejían** hasta formar una **solución global al problema de partida**

Introducción a la orientación a objetos

Enfoque estructurado

Era un proceso **centrado en los procedimientos**, se codificaban mediante funciones que actuaban sobre **estructuras de datos**, por eso a este tipo de programación se le llama **programación estructurada**

Sigue una filosofía en la que se intenta aproximar **qué hay que hacer, para así resolver un problema**

Introducción a la orientación a objetos

Enfoque orientado a objetos

Nuevo paradigma el proceso se centra en **simular los elementos de la realidad** asociada al problema de la forma más cercana posible

La abstracción que permite representar estos elementos se denomina **objeto**

Introducción a la orientación a objetos

Enfoque orientado a objetos

Los objetos tienen las siguientes características

- Conjunto de **atributos**, que son los datos que le caracterizan
- Conjunto de **operaciones** que definen su comportamiento
    - Actúan **sobre sus atributos para modificar su estado**

Introducción a la orientación a objetos

Enfoque orientado a objetos

Las aplicaciones orientadas a objetos están formadas por un conjunto de objetos que **interaccionan enviándose mensajes para producir resultados**

Los **objetos similares se abstraen en clases**, se dice que un **objeto es una instancia de una clase**

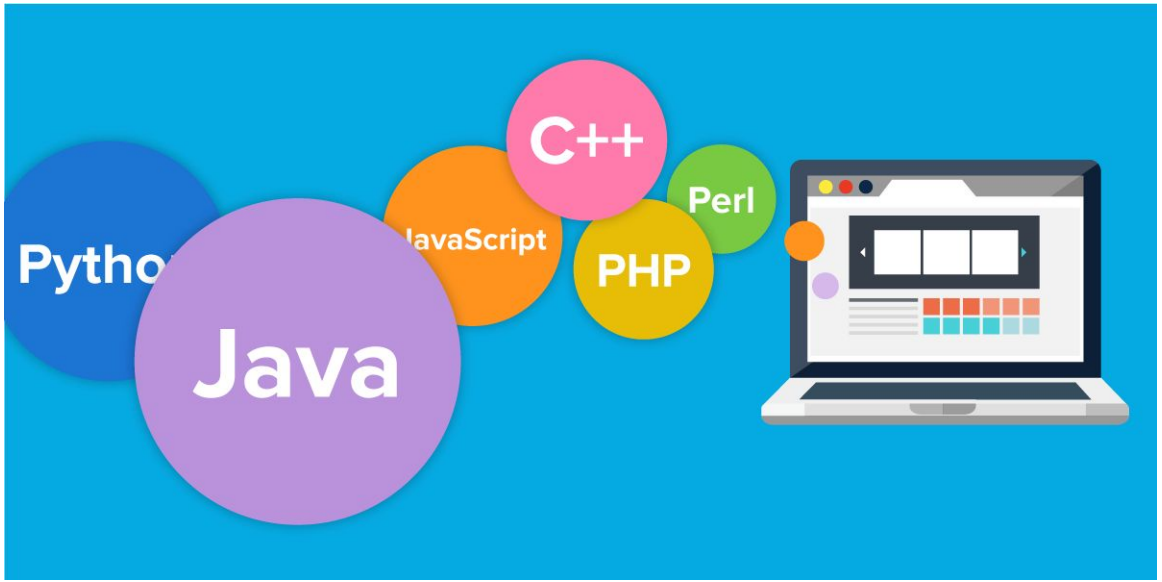
Introducción a la orientación a objetos

Enfoque orientado a objetos

Cuando se ejecuta un programa orientado a objetos ocurren tres sucesos:

1. Los **objetos se crean a medida que se necesitan**
2. Los **mensajes se mueven de un objeto a otro** (o del usuario a un objeto) a medida que el programa **procesa información** o responde a la **entrada del usuario**
3. Cuando los objetos ya no se necesitan, se **borran** y se **libera la memoria**

Introducción a la orientación a objetos



Conceptos de Orientación a Objetos

Trata de **acercarse al contexto del problema** lo más posible por medio de la **simulación de los elementos que intervienen** en su resolución y basa su desarrollo en los siguientes conceptos:

- **Abstracción**: representar las **características de la entidad hacia el mundo exterior**, dándonos una serie de **atributos y comportamientos** (propiedades y funciones) que podemos usar sin preocuparnos de qué pasa por dentro cuando lo hagamos

Conceptos de Orientación a Objetos

- **Encapsulación**: permite que todo lo referente a un objeto quede **aislado** dentro de éste. Es decir, que todos los datos referentes a un objeto queden "encerrados" dentro de éste y s**ólo se puede acceder a ellos a través de los miembros que la clase proporcione** (propiedades y métodos).

Conceptos de Orientación a Objetos

- **Modularidad**: Permite **subdividir una aplicación en partes más pequeñas** (llamadas módulos), cada una de las cuales debe ser tan **independiente** como sea posible de la aplicación en sí y de las restantes partes.

Conceptos de Orientación a Objetos

- **Polimorfismo**: Consiste en reunir **bajo el mismo nombre comportamientos diferentes**. La selección de uno u otro depende del objeto que lo ejecute.
    - Por ejemplo de la **clase Persona** todos llamar al **método Hablar()** aunque hablen de distinta forma o en ditinto idioma.

Conceptos de Orientación a Objetos

- **Herencia**: relación que se establece **entre objetos** en los que unos utilizan las **propiedades** y **comportamientos** de otros **formando una jerarquía**. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen.
- **Recolección de basura**: Técnica por la cual el entorno de objetos se encarga de **destruir automáticamente los objetos**, y por tanto desvincular su memoria asociada, que hayan quedado sin ninguna referencia a ellos.

Ventajas de la orientación a objetos

1. Permite **desarrollar software en mucho menos tiempo**, con **menos coste** y de **mayor calidad** gracias a la **reutilización** porque al ser completamente modular facilita la creación de código reusable dando la posibilidad de reutilizar parte del código para el desarrollo de una aplicación similar.

2. Se consigue aumentar la **calidad de los sistemas**, haciéndolos más extensibles ya que es muy sencillo aumentar o modificar la funcionalidad de la aplicación modificando las operaciones.

Ventajas de la orientación a objetos

3. Es más **fácil de modificar** y mantener porque se basa en criterios de **modularidad** en el que el sistema se descompone en objetos con unas **responsabilidades** claramente **especificadas e independientes del resto**.

4. La tecnología de objetos **facilita la adaptación** al entorno y el cambio haciendo aplicaciones escalables. Es sencillo **modificar la estructura y el comportamiento** de los objetos **sin tener que cambiar la aplicación**.

Ventajas de la orientación a objetos

V o F sobre el paradigma de orientación a objetos

Permite crear aplicaciones basadas en módulos de software que representan objetos del entorno del sistema, por lo que no son apropiados para dar solución a otros problemas.

Ventajas de la orientación a objetos

V o F sobre el paradigma de orientación a objetos

Permite crear aplicaciones basadas en módulos de software que representan objetos del entorno del sistema, por lo que no son apropiados para dar solución a otros problemas.

F. La orientación a objetos tiene la ventaja de que una vez definido un módulo (objeto) puede utilizarse en cualquier aplicación en la pueda ser útil.

Ventajas de la orientación a objetos

V o F sobre el paradigma de orientación a objetos

Tiene como objetivo la creación de aplicaciones basadas en abstracciones de datos estáticas y de difícil ampliación.

Ventajas de la orientación a objetos

V o F sobre el paradigma de orientación a objetos

Tiene como objetivo la creación de aplicaciones basadas en abstracciones de datos estáticas y de difícil ampliación.

F, los principios de encapsulación y modularidad permiten la construcción de abstracciones con un estado y un comportamiento propio que suele ser sencillo de ampliar gracias a la independencia del resto de objetos del sistema.

Ventajas de la orientación a objetos

V o F sobre el paradigma de orientación a objetos

Permite crear aplicaciones cuyo mantenimiento es complicado porque las modificaciones influyen a todos los objetos del sistema.

Ventajas de la orientación a objetos

V o F sobre el paradigma de orientación a objetos

Permite crear aplicaciones cuyo mantenimiento es complicado porque las modificaciones influyen a todos los objetos del sistema.

F, el mantenimiento de estas aplicaciones suele ser más sencillo porque afecta a un solo módulo de forma que el efecto que puede producir sobre el resto de la aplicación es nulo o muy pequeño.

Ventajas de la orientación a objetos

V o F sobre el paradigma de orientación a objetos

Permite crear aplicaciones basadas en módulos que pueden reutilizarse, de fácil modificación y que permiten su ampliación en función del crecimiento del sistema.

Ventajas de la orientación a objetos

V o F sobre el paradigma de orientación a objetos

Permite crear aplicaciones basadas en módulos que pueden reutilizarse, de fácil modificación y que permiten su ampliación en función del crecimiento del sistema.

V, gracias a que cumple con las propiedades de modularidad, encapsulación, reutilización y escalabilidad.

Clases, objetos, atributos y métodos

Los objetos de un sistema se abstraen, **en función de sus características comunes**, en **clases**.

Una **clase** está formada por un **conjunto de procedimientos y datos** que resumen **características similares de un conjunto de objetos**.

La clase tiene dos propósitos: **definir abstracciones y favorecer la modularidad**.

Clases, objetos, atributos y métodos

Los **atributos** son un conjunto de **características asociadas a una clase.**

Cuando toman valores concretos dentro de su dominio definen el estado del objeto. Se definen por su **nombre y su tipo**.

Por ejemplo, lo atributos serán cada una de las piezas o características del coche: nombre, color, modelo, plazas, caballaje, velocidad, etc.

Clases, objetos, atributos y métodos

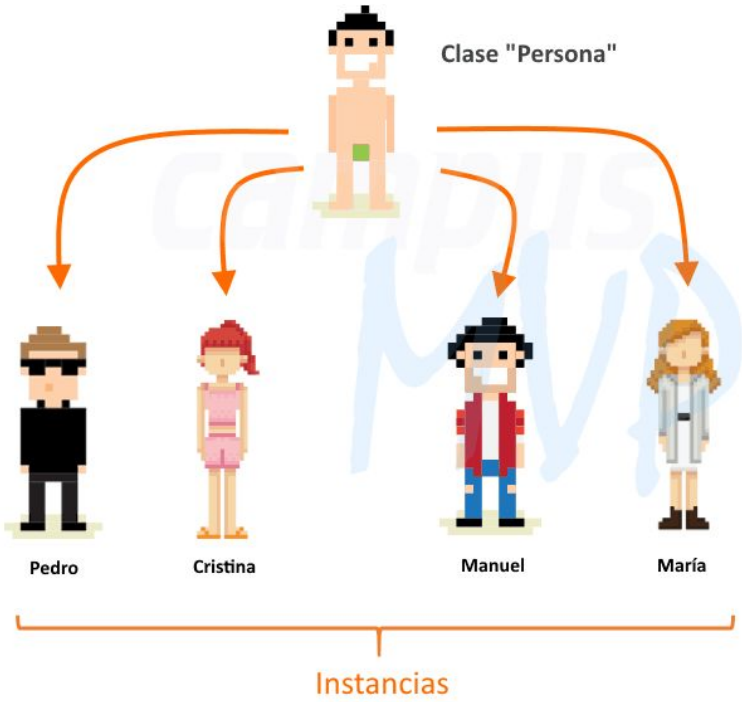
Un **método** es el **procedimiento o función que se invoca para actuar sobre un objeto**.

Un **mensaje** es el **resultado de cierta acción efectuada por un objeto**.

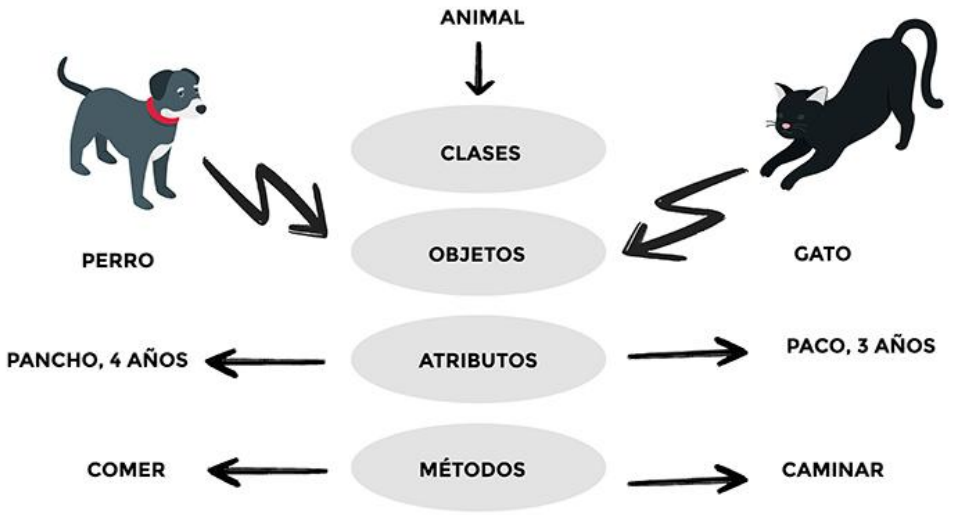
Determinan **cómo actúan los objetos cuando reciben un mensaje**, es decir, cuando se requiere que el objeto realice una acción descrita en un método se le envía un mensaje.

Por ejemplo, podríamos tener los métodos: arrancar, acelerar, frenar, encenderLuces, cambiarColor, etc

Clases, objetos, atributos y métodos



Clases, objetos, atributos y métodos



profile

Clases, objetos, atributos y métodos

V o F sobre el paradigma de orientación a objetos

Un objeto es una concreción de una clase, es decir, en un objeto se concretan valores para los atributos definidos en la clase, y además, estos valores podrán modificarse a través del paso de mensajes al objeto.

Clases, objetos, atributos y métodos

V o F sobre el paradigma de orientación a objetos

Un objeto es una concreción de una clase, es decir, en un objeto se concretan valores para los atributos definidos en la clase, y además, estos valores podrán modificarse a través del paso de mensajes al objeto.

V, el objeto tiene un estado formado por los valores concretos que toman los atributos de la clase a la que pertenece, además para modificar estos valores tenemos que hacerlo utilizando los métodos de la clase a través del paso de mensajes.

Ejercicio clases

Define

- dos clases de tu elección,
- tres objetos que se intancien de cada una de esas clases,
- tres atributos que correspondan a cada uno de los objetos anteriores
- y tres métodos de cada para cada objeto

Visibilidad

Principio de ocultación: propiedad que consiste en **aislar el estado** de manera que sólo se puede cambiar mediante las **operaciones definidas** en una clase

Protege a los datos de que sean modificados por alguien que no tenga derecho a acceder a ellos.

Existen **distintos niveles de ocultación** que se implementan en lo que se denomina **visibilidad**.

Es una característica que define el **tipo de acceso que se permite a atributos y métodos**.

Visibilidad

Público: Se pueden acceder desde cualquier clase y cualquier parte del programa.

Privado: Sólo se pueden acceder desde operaciones de la clase.

Protegido: Sólo se pueden acceder desde operaciones de la clase o de clases derivadas en cualquier nivel.

Objetos. Instanciación

Cada vez que se construye un **objeto** en un programa informático a **partir de una clase** se crea lo que se conoce como **instancia de esa clase**.

Un objeto se define por:

- Su **estado**: es la concreción de los atributos definidos en la clase a un valor concreto.
- Su **comportamiento**: definido por los métodos públicos de su clase.
- Su **tiempo de vida**: intervalo de tiempo a lo largo del programa en el que el objeto existe. Desde su creación a su destrucción.

UML

UML (**Lenguaje de Modelado Unificado**) es una familia de **diagramas gráficos** que ayudan a **describir y diseñar sistemas de software**.

Es un estándar controlado por el **OMG** (Object Management Group), un consorcio de **empresas**. Nació en 1997 como una unificación de los **diversos sistemas de modelado gráfico** que existían hasta el momento.

¿Por qué modelar?

Modelar es **diseñar la aplicación de software antes de escribir su código**. Equivaldría a elaborar los planos de un edificio, en los que queda claro dónde van los ascensores, el cableado, etc. antes de empezar a construirlo.

- Permite utilizar un **lenguaje común** que facilita la comunicación entre el equipo de desarrollo.
- Podemos **documentar todos los artefactos de un proceso de desarrollo** (requisitos, arquitectura, pruebas, versiones...) por lo que se dispone de documentación que trasciende al proyecto.
- Permite **especificar todas las decisiones de análisis, diseño e implementación**, construyéndose modelos precisos, no ambiguos y completos.

Tipos de diagramas UML

UML define un sistema como una colección de modelos que describen sus diferentes perspectivas. Los modelos se implementan en una serie de **diagramas** que son **representaciones gráficas de una colección de elementos de modelado**.

- **Diagramas estructurales**: Representan la visión estática del sistema. Especifican **clases** y **objetos** y **cómo se distribuyen físicamente** en el sistema.
- **Diagramas de comportamiento**: Muestran la **conducta** en tiempo de ejecución del sistema, tanto desde el punto de vista del sistema completo como de las instancias u objetos que lo integran. Dentro de este grupo están los diagramas de interacción.

Tipos de diagramas UML

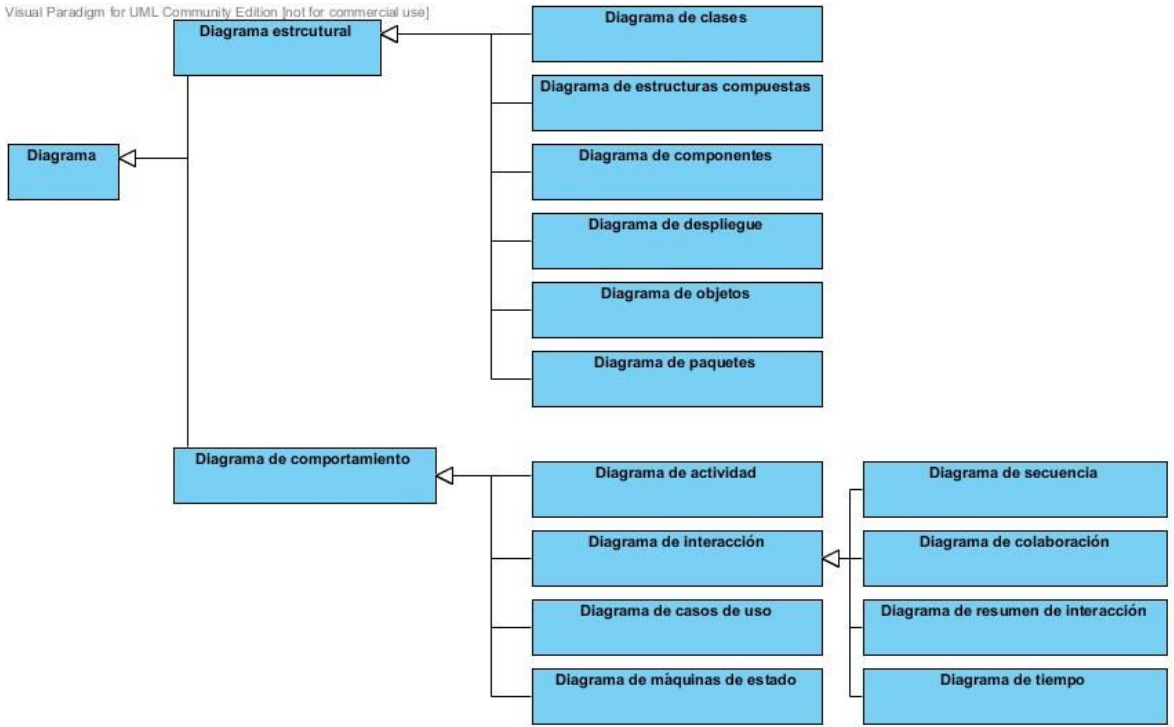


Diagrama de Clases

Muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones.

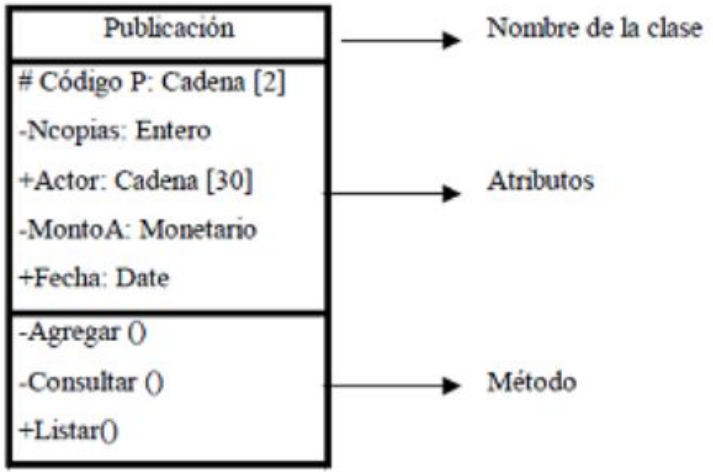
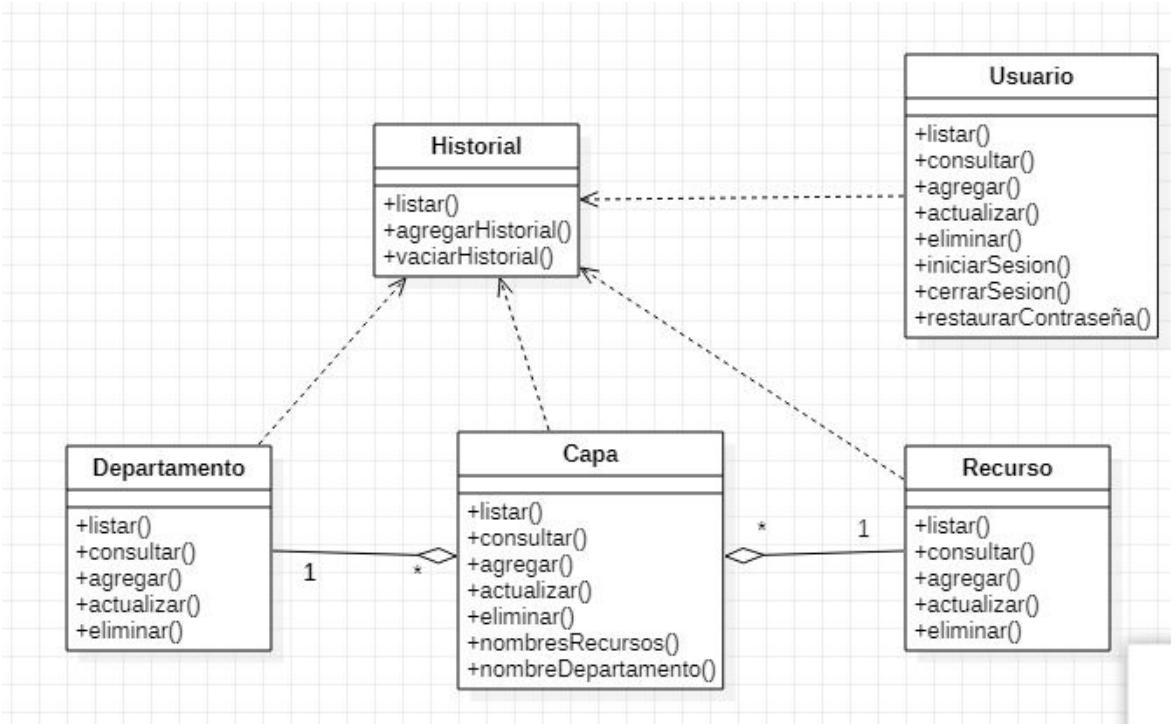


Diagrama de Clases



Diagramas de Casos de Uso

Muestra un conjunto de **casos de uso y actores y sus relaciones**

Se define el sistema desde el punto de vista del **usuario**

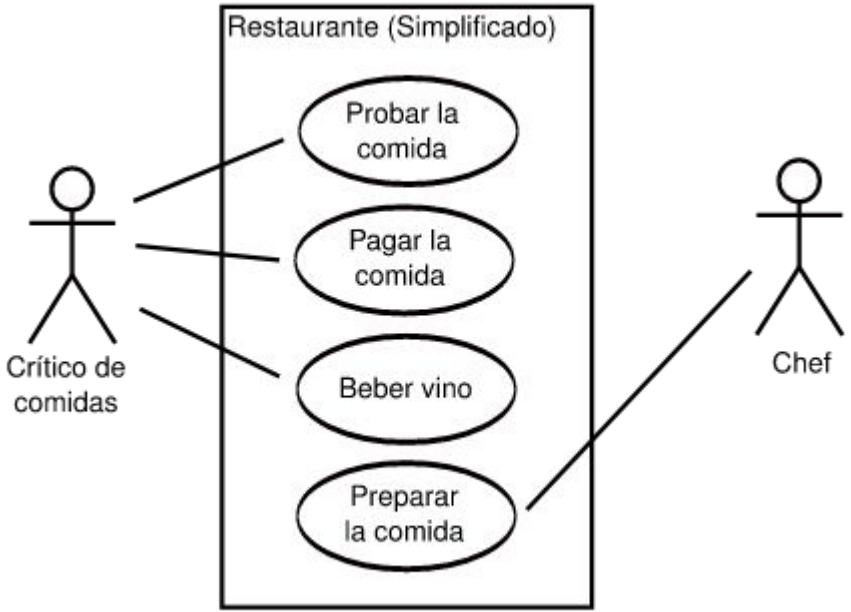


Diagrama de Secuencia

En un sistema funcional los **objetos interactúan entre sí**, y tales interacciones sucede con el **tiempo**.

Muestra la mecánica de la interacción en el tiempo.

Diagrama de interacción con la **relación temporal** de los **mensajes y los objetos.**

Diagrama de Secuencia

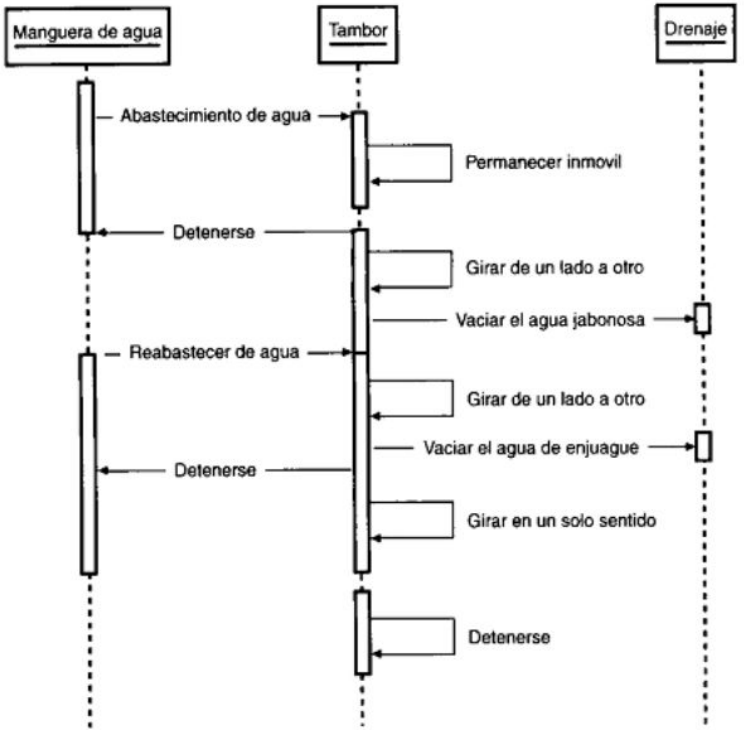
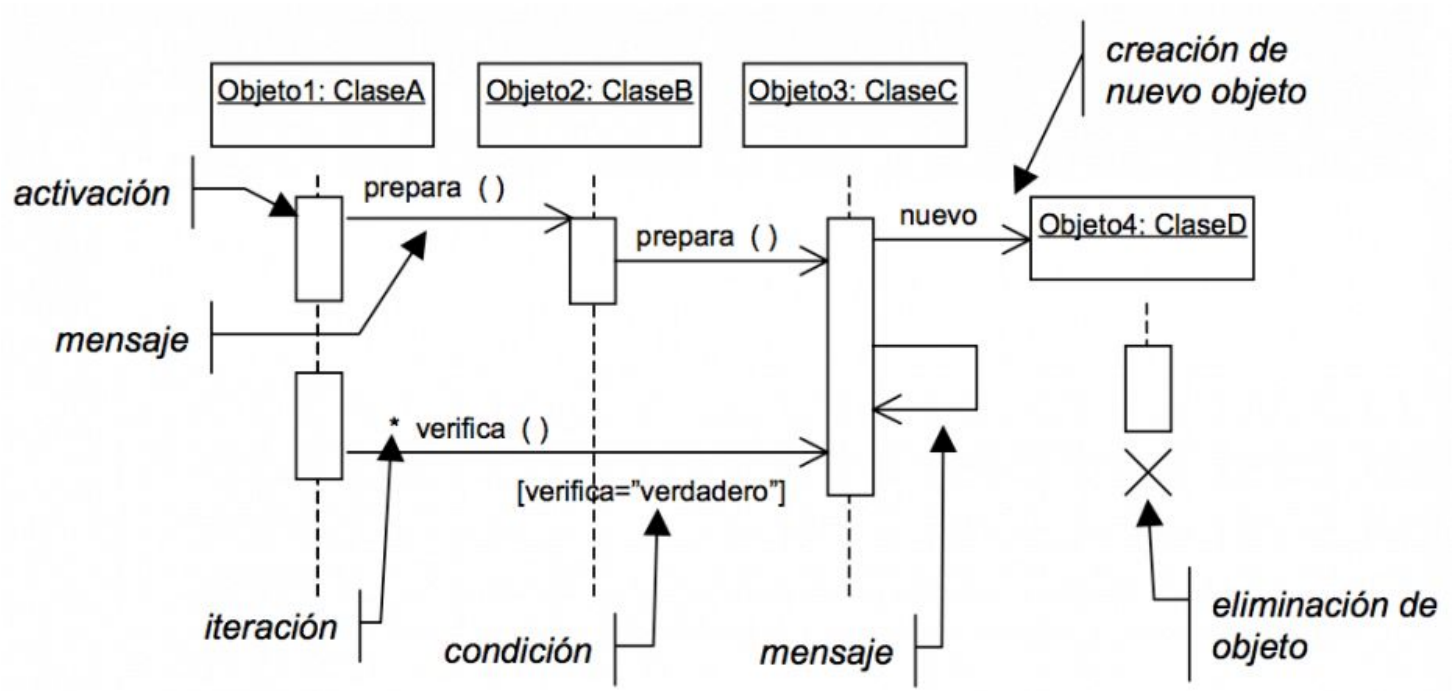


Diagrama de Secuencia



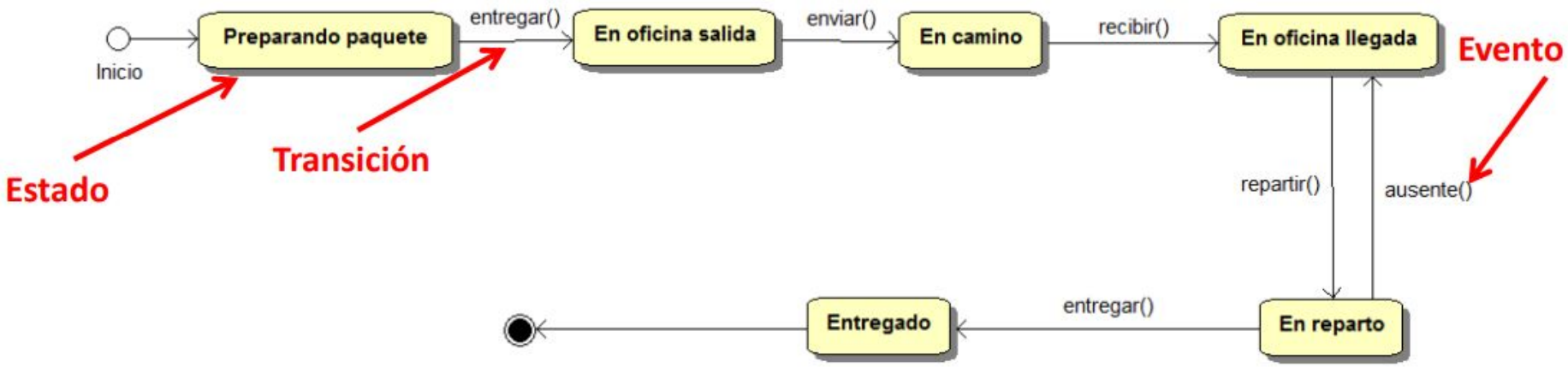
Diagramas de Estados

Muestra los **estados** por los que pasa una máquina de estados finitos, es decir, un modelo de comportamiento que consiste en acciones y estados o transiciones a otros estados

Consta de **estados**, **transiciones**, **eventos** y **actividades**.

El diagrama de estado permite representar el **ciclo de vida** completo de cualquier sistema, subsistema o componentes o clases del mismo.

Diagramas de Estados



Diagramas de Estados