SENATI

INFORMATION TECHNOLOGY

Object-Oriented Design method

SENATI

TECNOLOGÍAS DE LA INFORMACIÓN

Método de diseño orientado a objetos

Objective: To recognize the different terms related to object-oriented design.

Content:

1. Objects
2. Classes
3. Messages
4. Abstraction
5. Encapsulation

Objetivo:

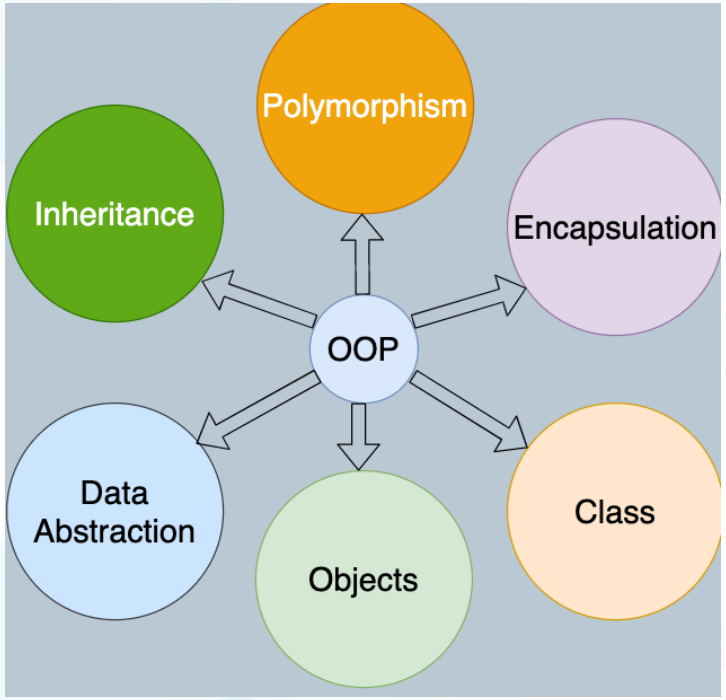
Reconocer los diferentes términos relacionados con el diseño orientado a objetos.

Contenido:

1. Objetos
2. Clases
3. Mensajes
4. Abstracción
5. Encapsulación

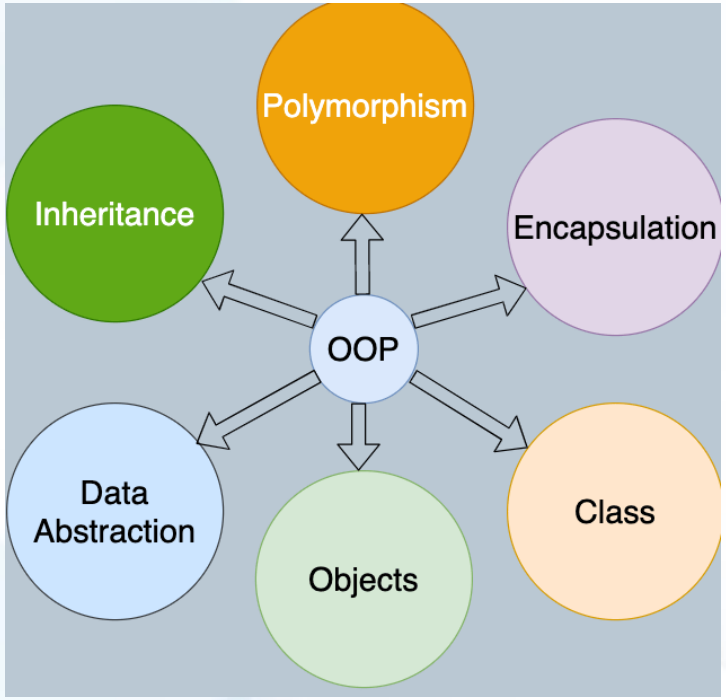
Object-oriented programming (OOP)

A software programming model constructed around objects. This model compartmentalizes data into objects (data fields) and describes object contents and behavior through the declaration of classes (methods).



Programación orientada a objetos (POO)

Un modelo de programación de software construido en torno a objetos. Este modelo compartimenta los datos en objetos (campos de datos) y describe el contenido y el comportamiento de los objetos mediante la declaración de clases (métodos).



OOP features include:

*Objects	*Classes
*Messages	*Abstraction
*Encapsulation	*Inheritance
*Polymorphism	*Refactoring

Object-oriented programming allows for simplified programming. Its benefits include reusability, refactoring, extensibility, maintenance and efficiency.

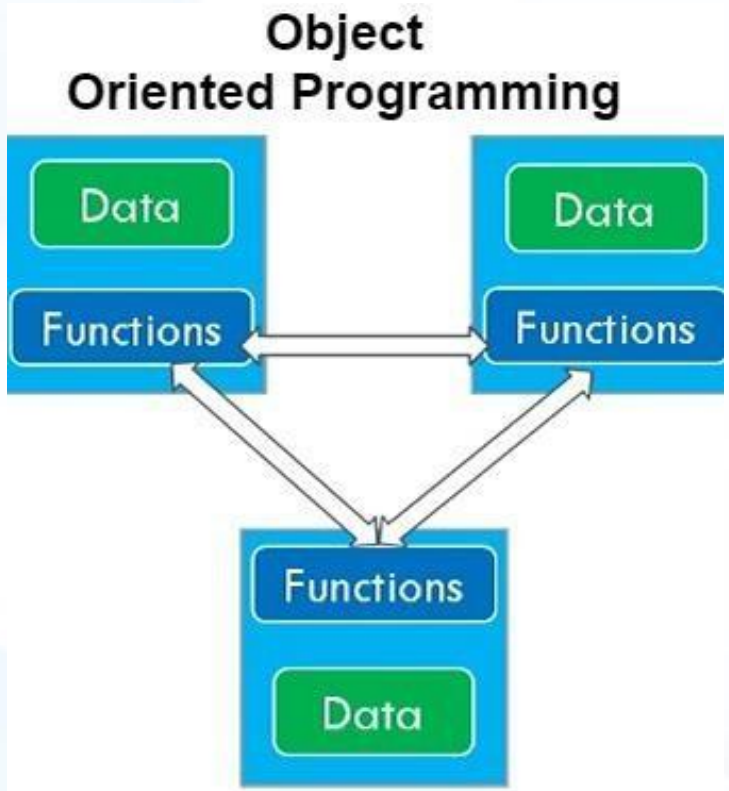
Las características de OOP incluyen:

*Objetos
*Clases
*Mensajes
*Abstracción
*Encapsulación
*Herencia
*Polimorfismo
*Refactorización

La programación orientada a objetos permite una programación simplificada. Sus ventajas incluyen reutilización, refactorización, extensibilidad, mantenimiento y eficiencia.

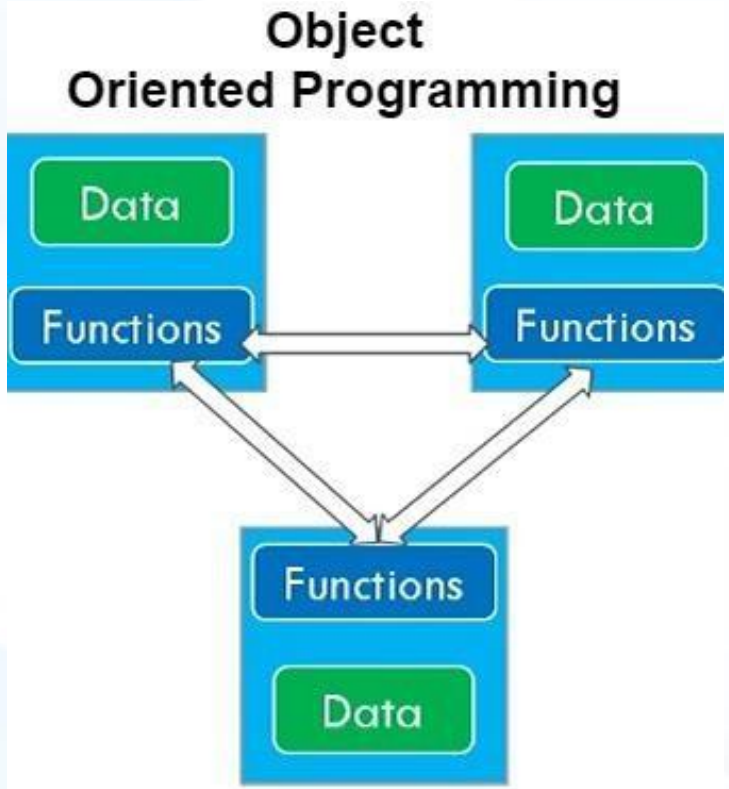
Objects

All entities involved in the solution design are known as objects: people, banks, companies, are considered as objects. Every entity has some attributes associated with it and has some methods to perform on the attributes. Object type is useful where there is a requirement to build generic routines. An object is also known as instance.



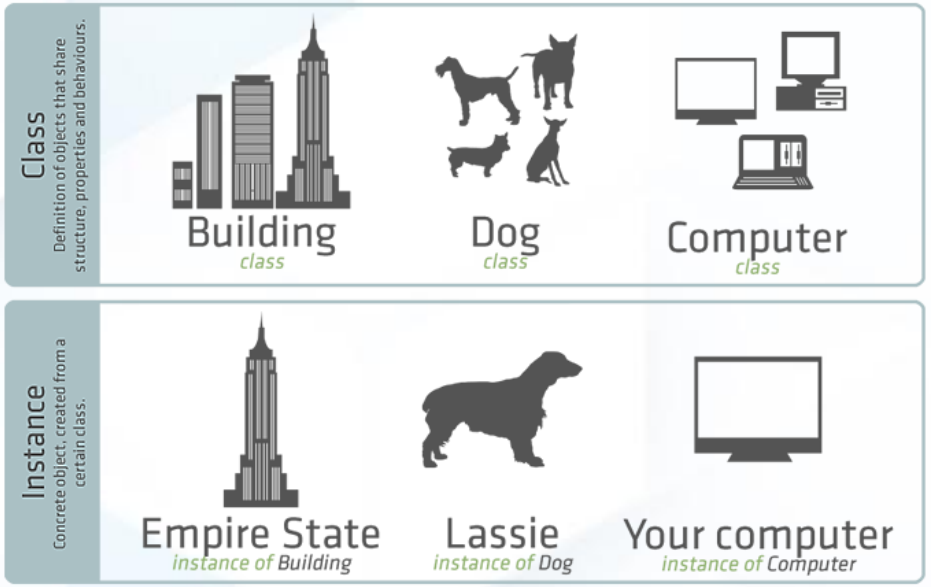
Objetos

Todas las entidades involucradas en el diseño de la solución se conocen como objetos: personas, bancos y empresas. Cada entidad tiene atributos asociados y métodos para ejecutarlos. El tipo de objeto es útil cuando se requiere crear rutinas genéricas. Un objeto también se conoce como instancia.



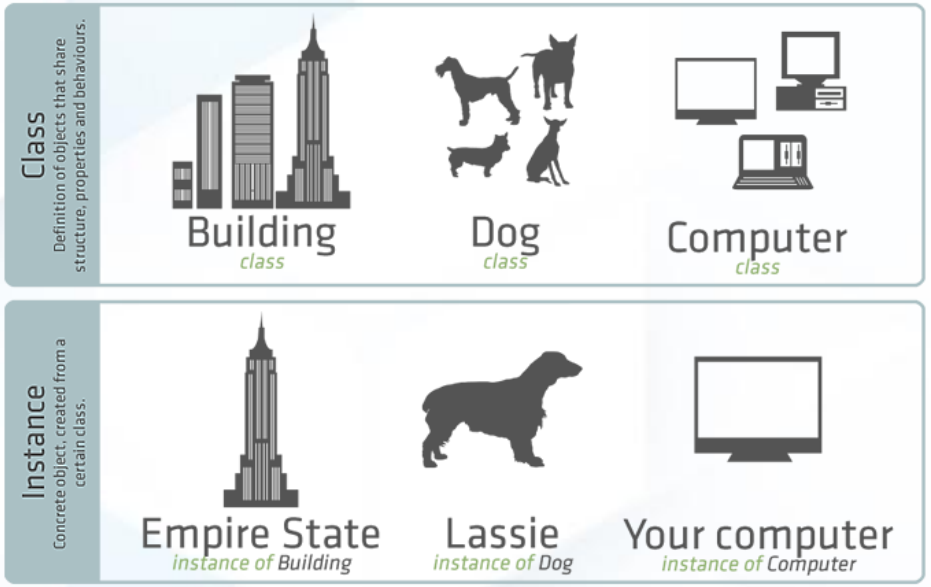
Classes

A class is a generalized description of an object. An object is an instance of a class. A class defines all the attributes, which an object can have and methods, which represents the functionality of the object.



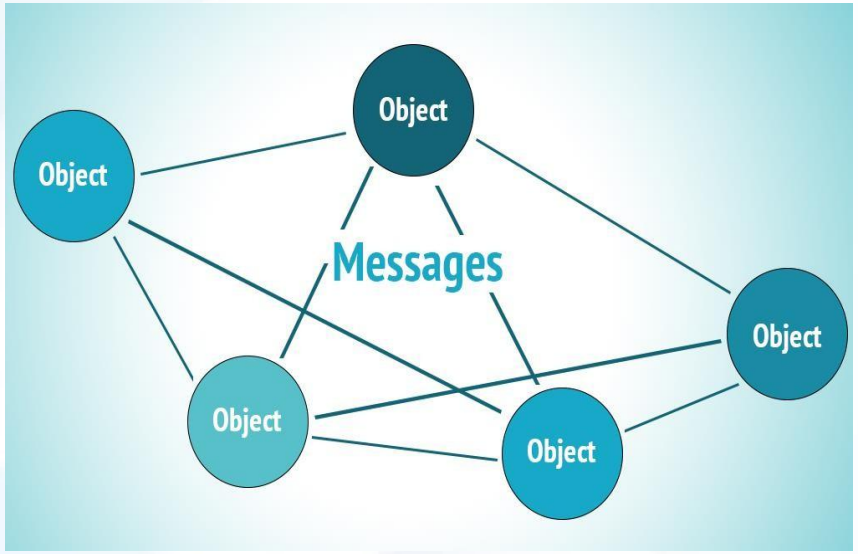
Clases

Una clase es una descripción generalizada de un objeto. Un objeto es una instancia de una clase. Una clase define todos los atributos que un objeto puede tener y los métodos que representan su funcionalidad.



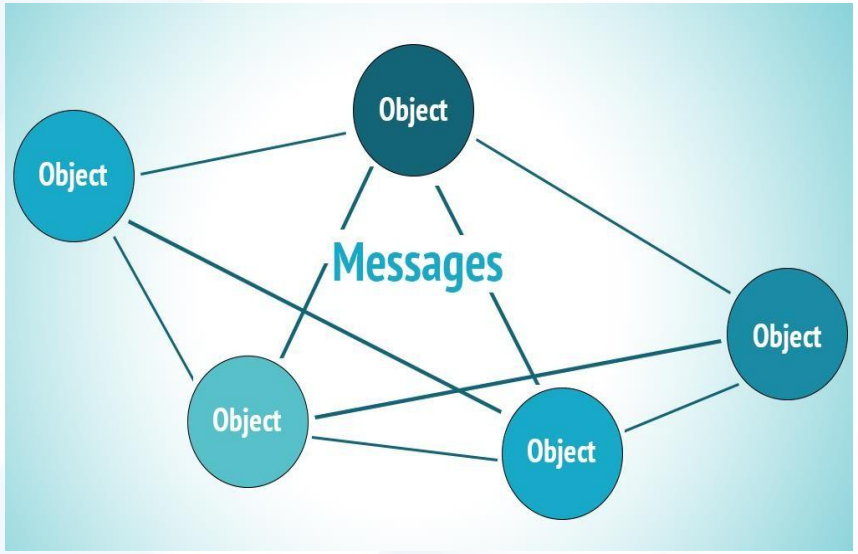
Messages

Objects communicate by message passing. Messages consist of the integrity of the target object, the name of the requested operation, and any other action needed to perform the function. Messages are often implemented as procedure or function calls.



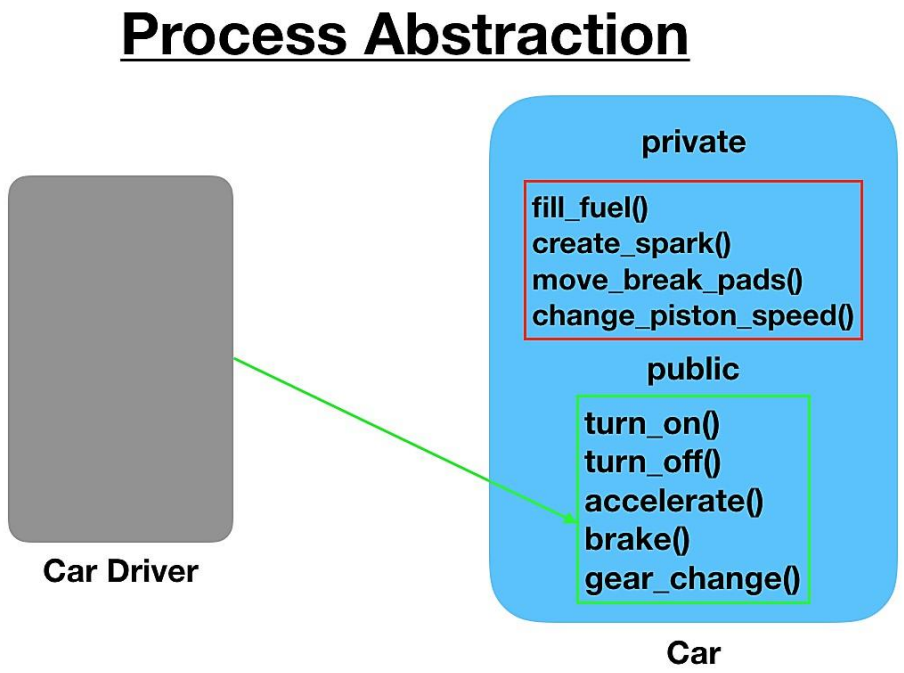
Mensajes

Los objetos se comunican mediante el paso de mensajes. Los mensajes contienen la integridad del objeto de destino, el nombre de la operación solicitada y cualquier otra acción necesaria para realizar la función. Los mensajes suelen implementarse como llamadas a procedimientos o funciones.



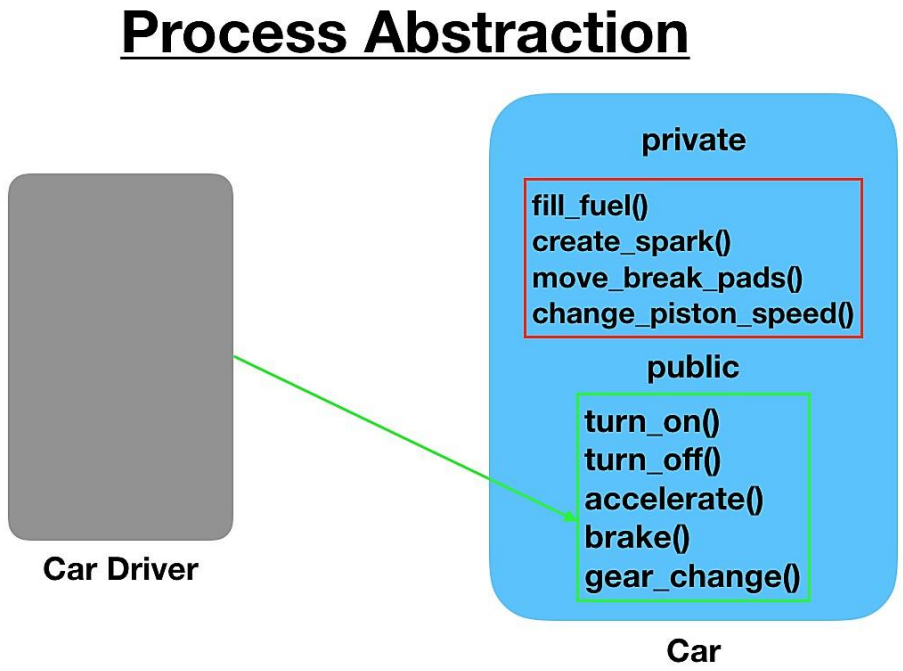
Abstraction

In object-oriented design, complexity is handled using abstraction. Abstraction is the removal of the irrelevant, and the amplification of the essentials.



Abstracción

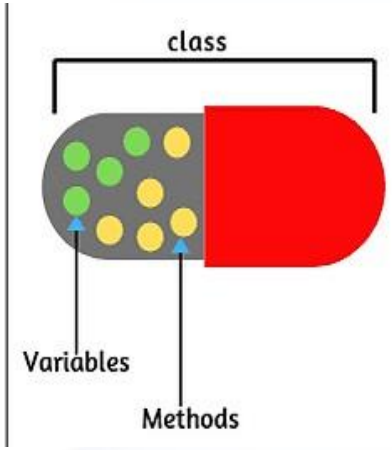
En el diseño orientado a objetos, la complejidad se gestiona mediante la abstracción. La abstracción consiste en eliminar lo irrelevante y ampliar lo esencial.



Encapsulation

Also called an information hiding, doesn`t only bundles essential information of an object together but also restricts access to the data and methods from the outside world. It refers to an object's ability to hide data and behavior that are not necessary to its user. Encapsulation enables a group of properties, methods and other members to be considered a single unit or object.

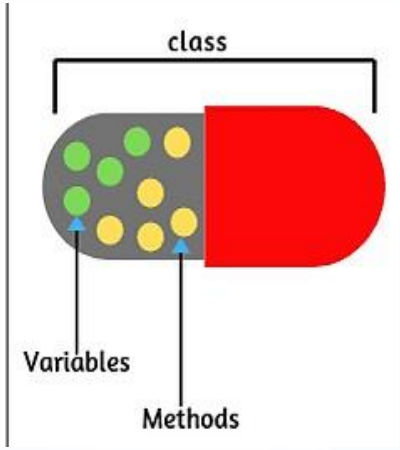
```
class
{

  data members
        +
  methods (behavior)

}
```



Encapsulación

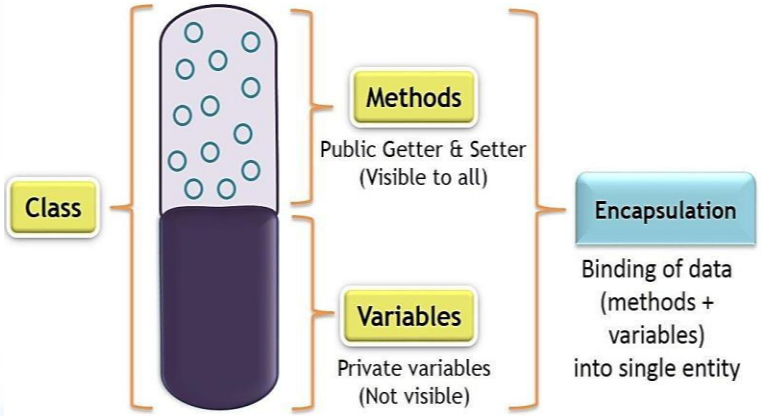
También denominada ocultación de información, no solo agrupa la información esencial de un objeto, sino que también restringe el acceso a los datos y métodos desde el exterior. Se refiere a la capacidad de un objeto para ocultar datos y comportamientos innecesarios para su usuario. La encapsulación permite que un grupo de propiedades, métodos y otros miembros se considere una sola unidad u objeto.

```
class
{

  data members
        +
  methods (behavior)

}
```



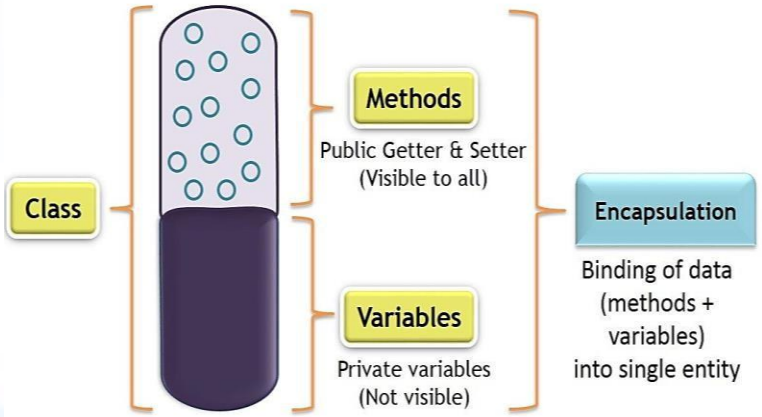
Benefits of encapsulation:

- Protection of data from accidental corruption.
- Specification of the accessibility of each of the members of a class to the code outside the class.
- Flexibility and extensibility of the code and reduction in complexity.
- Lower coupling between objects and hence improvement in code maintainability.



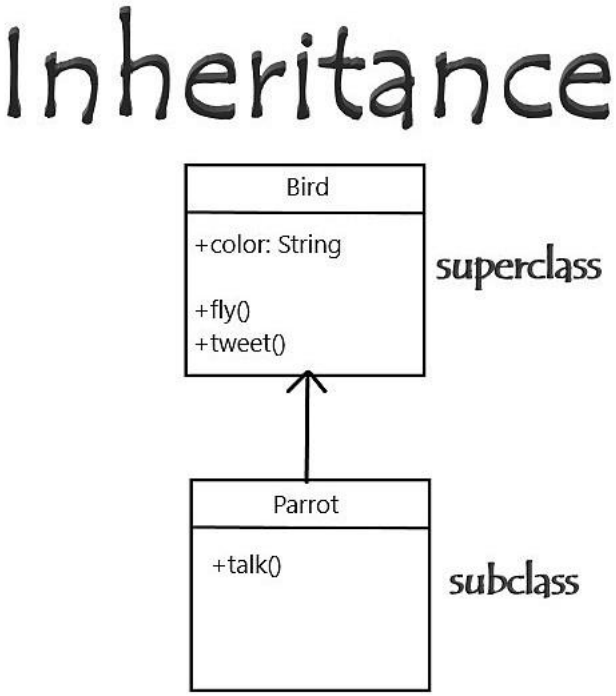
Beneficios de la encapsulación:

- Protección de datos contra corrupción accidental.
- Especificación de la accesibilidad de cada miembro de una clase al código externo.
- Flexibilidad y extensibilidad del código, así como reducción de la complejidad.
- Reducción del acoplamiento entre objetos y, por consiguiente, mejora de la mantenibilidad del código.



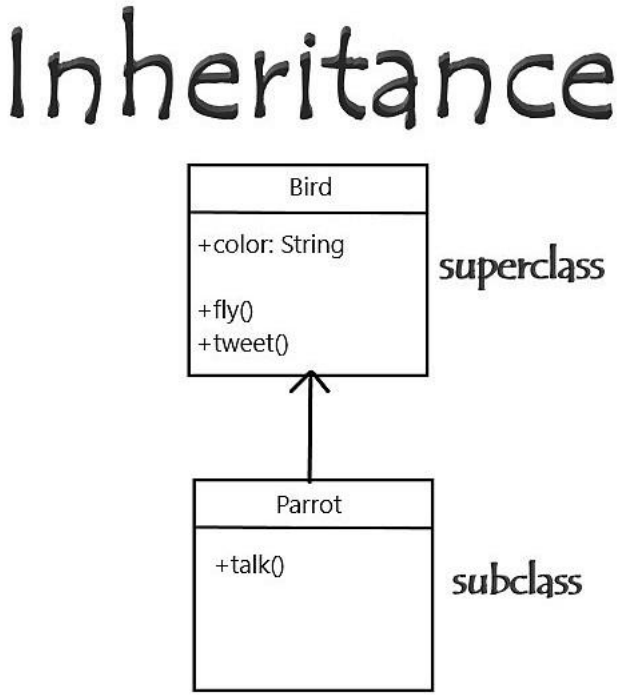
Inheritance

OOD allows similar classes to stack up in a hierarchical manner where the lower or sub-classes can import, implement, and re-use allowed variables and functions from their immediate superclasses. This property of OOD is called an inheritance. This makes it easier to define a specific class and to create generalized classes from specific ones.

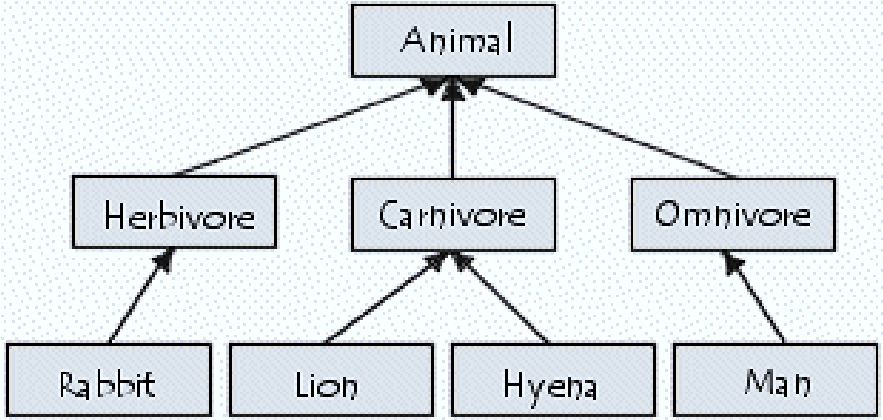


Herencia

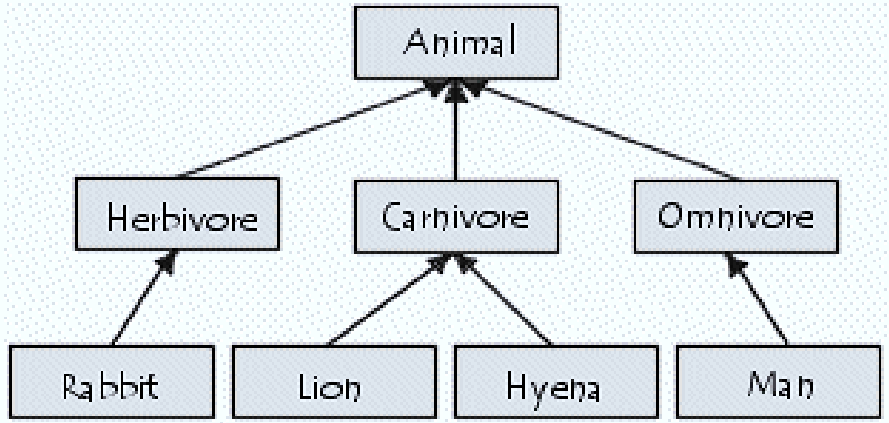
OOD permite que clases similares se apilen jerárquicamente, donde las clases inferiores o subclases pueden importar, implementar y reutilizar las variables y funciones permitidas de sus superclases inmediatas. Esta propiedad de OOD se denomina herencia. Esto facilita la definición de una clase específica y la creación de clases generalizadas a partir de clases específicas.



- In Java, classes may inherit or acquire the properties and methods of other classes.
- A class derived from another class is called a subclass, whereas the class from which a subclass is derived is called a superclass.
- A subclass can have only one superclass, whereas a superclass may have one or more subclasses.

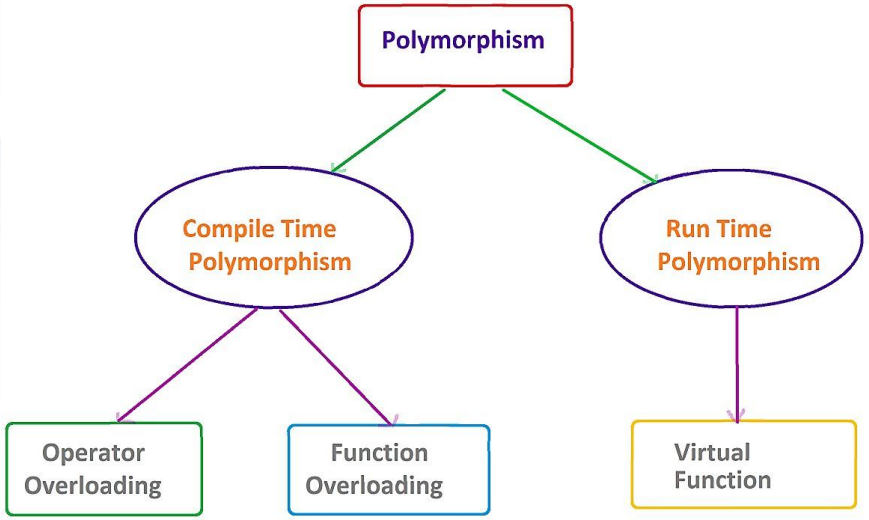


- En Java, las clases pueden heredar o adquirir las propiedades y métodos de otras clases.
- Una clase derivada de otra se denomina subclase, mientras que la clase de la que se deriva una subclase se denomina superclase.
- Una subclase solo puede tener una superclase, mientras que una superclase puede tener una o más subclases.



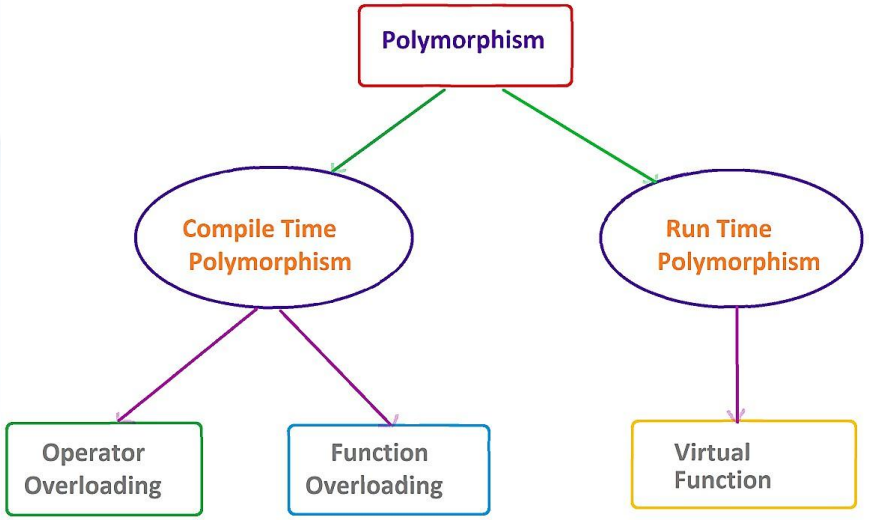
Polymorphism

OOD languages provide a mechanism where methods performing similar tasks but vary in arguments, can be assigned the same name. This is known as polymorphism, which allows a single interface is performing functions for different types. Depending upon how the service is invoked, the respective portion of the code gets executed.

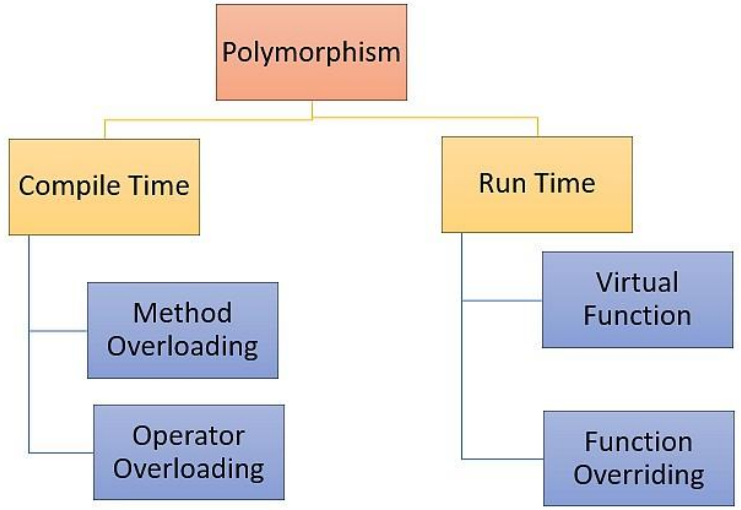


Polimorfismo

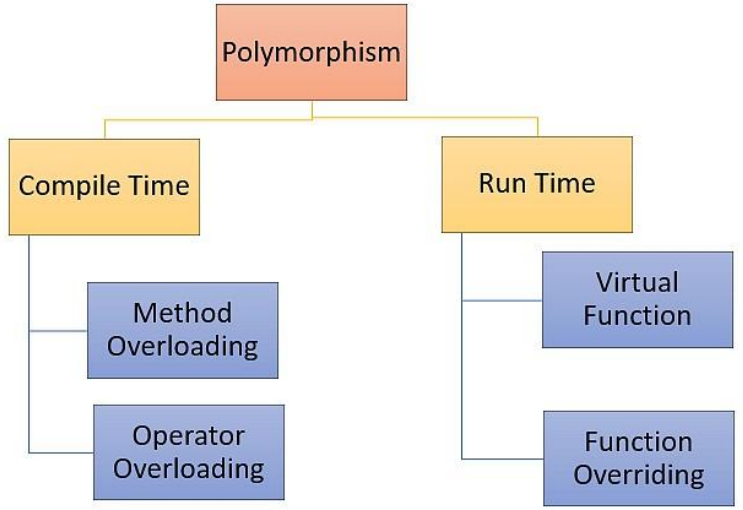
Los lenguajes OOD proporcionan un mecanismo que permite asignar el mismo nombre a métodos que realizan tareas similares, pero que varían en sus argumentos. Esto se conoce como polimorfismo, y permite que una sola interfaz realice funciones para diferentes tipos. Según cómo se invoque el servicio, se ejecutará la parte correspondiente del código.



Polymorphism is the ability of objects of different types to provide a unique interface for different implementations of methods. It is usually used in the context of late binding, where the behavior of an object to respond to a call to its method members is determined based on object type at run time. Polymorphism enables redefining methods in derived classes and forms one of the fundamental concepts of object-oriented programming, along with encapsulation and inheritance.

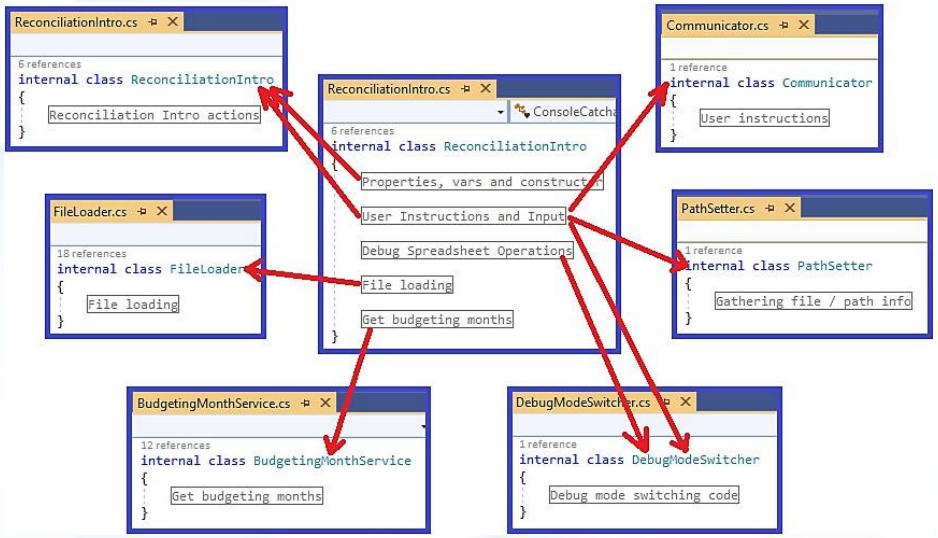


Polimorfismo Es la capacidad de objetos de diferentes tipos de proporcionar una interfaz única para diferentes implementaciones de métodos. Se utiliza generalmente en el contexto de enlace tardío, donde el comportamiento de un objeto para responder a una llamada a sus miembros de método se determina en función del tipo de objeto en tiempo de ejecución. El polimorfismo permite redefinir métodos en clases derivadas y constituye uno de los conceptos fundamentales de la programación orientada a objetos, junto con la encapsulación y la herencia..



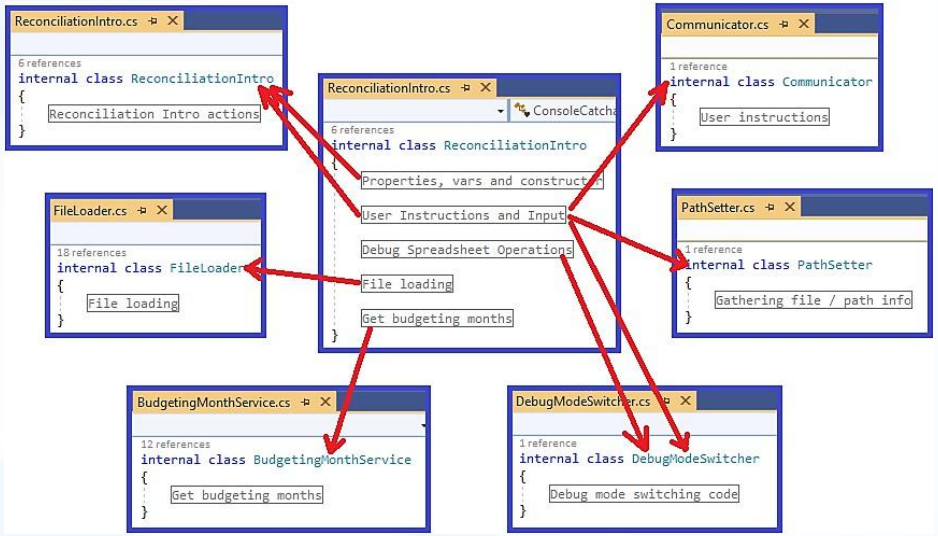
Refactoring

Also known as reengineering, it is the process of altering an application's source code without changing its external behavior. Its purpose is to improve some of the nonfunctional properties of the code, such as readability, complexity, maintainability and extensibility. Refactoring can extend the life of source code.



Refactorización

También conocida como reingeniería, es el proceso de modificar el código fuente de una aplicación sin modificar su comportamiento externo. Su propósito es mejorar algunas propiedades no funcionales del código, como la legibilidad, la complejidad, la mantenibilidad y la extensibilidad. La refactorización puede prolongar la vida útil del código fuente.



Quiz

1. A software programming model constructed around objects.

a. OJT b. OOP c. Michel Paradis d. Michel Barnier

2. _____is useful where there is a requirement to build generic routines.

a. Inheritance b. Polymorphism c. Object d. Classes

3. _____makes easier to define a specific class and to create generalized classes.

a. Inheritance b. Refactoring c. Object d. Classes

4. _____allows a single interface is performing functions for different types.

a. Classes b. Polymorphism c. Object d. Refactoring

5. Also known as reengineering.

a. Classes b. Polymorphism c. Object d. Refactoring

Quiz

1. A software programming model constructed around objects.

a. OJT b. OOP c. Michel Paradis d. Michel Barnier

2. _____is useful where there is a requirement to build generic routines.

a. Inheritance b. Polymorphism c. Object d. Classes

3. _____makes easier to define a specific class and to create generalized classes.

a. Inheritance b. Refactoring c. Object d. Classes

4. _____allows a single interface is performing functions for different types.

a. Classes b. Polymorphism c. Object d. Refactoring

5. Also known as reengineering.

a. Classes b. Polymorphism c. Object d. Refactoring

Conclusions

Object-oriented design (OOD) is the process of using an object-oriented methodology to design a computing system or application. This technique enables the implementation of a software solution based on the concepts of objects. OOD serves as part of the object-oriented programming (OOP) process or lifecycle. In object-oriented system design and development, OOD helps in designing the system architecture or layout - usually after completion of an object-oriented analysis (OOA). The designed system is later created or programmed using object-oriented based techniques and/or an object-oriented programming language (OOPL).

Conclusions

El diseño orientado a objetos (DOO) es el proceso de utilizar una metodología orientada a objetos para diseñar un sistema o aplicación informática. Esta técnica permite la implementación de una solución de software basada en el concepto de objetos. El DOO forma parte del proceso o ciclo de vida de la programación orientada a objetos (POO). En el diseño y desarrollo de sistemas orientados a objetos, el DOO facilita el diseño de la arquitectura o el diseño del sistema, generalmente tras la finalización de un análisis orientado a objetos (AOO). El sistema diseñado se crea o programa posteriormente utilizando técnicas basadas en la orientación a objetos o un lenguaje de programación orientado a objetos (LPO).