Sesión 2
Tecnología Orientada a Objetos

Unidad 1

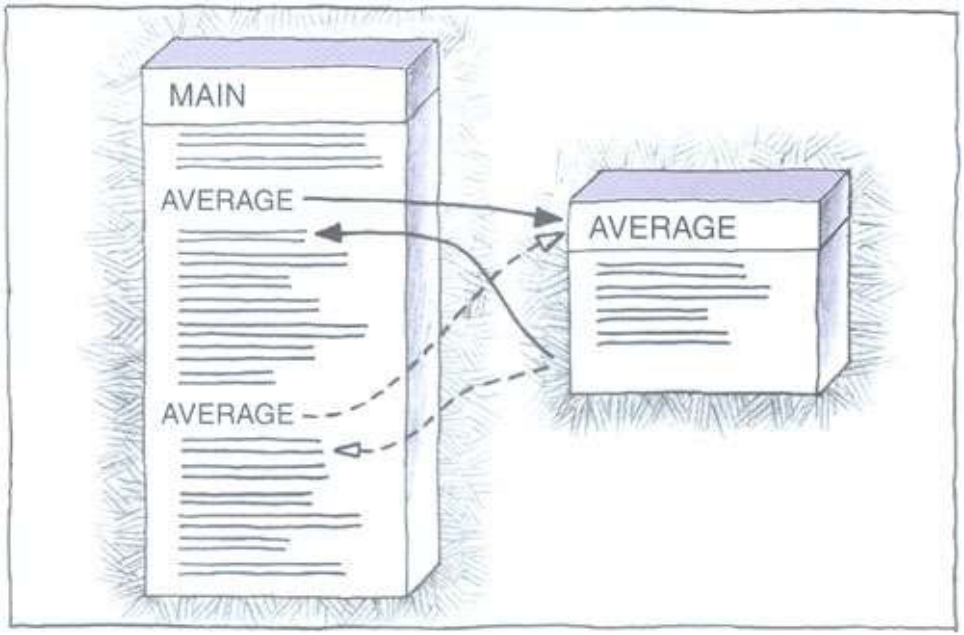
Mg. Gustavo G. Delgado Ugarte

Historia de la tecnología de objetos

- Programación Modular
- Programación Estructurada
- Ingeniería de Software Asistida por Computadora
- Lenguajes de 4ta Generación

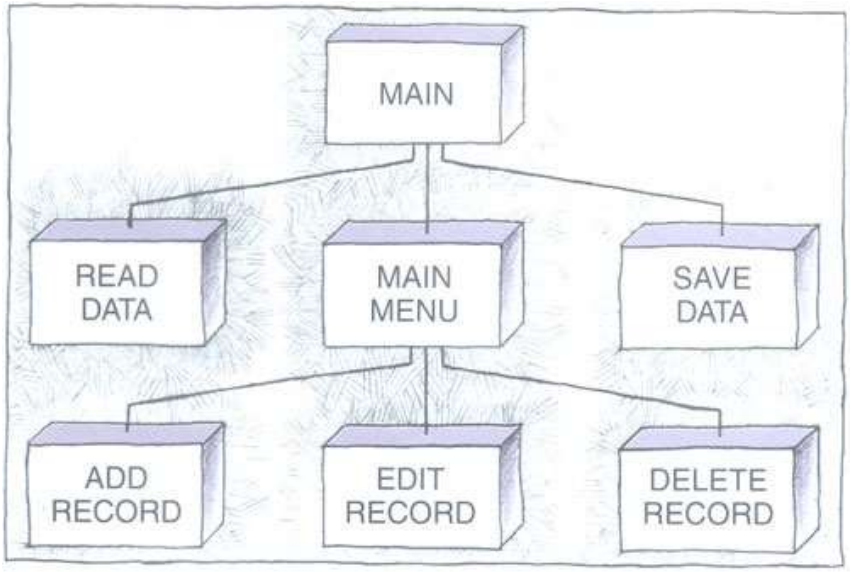
Programación Modular

- Consiste en construir un gran programa, dividiéndolo en pequeñas partes y combinándolas
- El soporte más elemental es la subrutina, que surgió a inicios de los 1950s



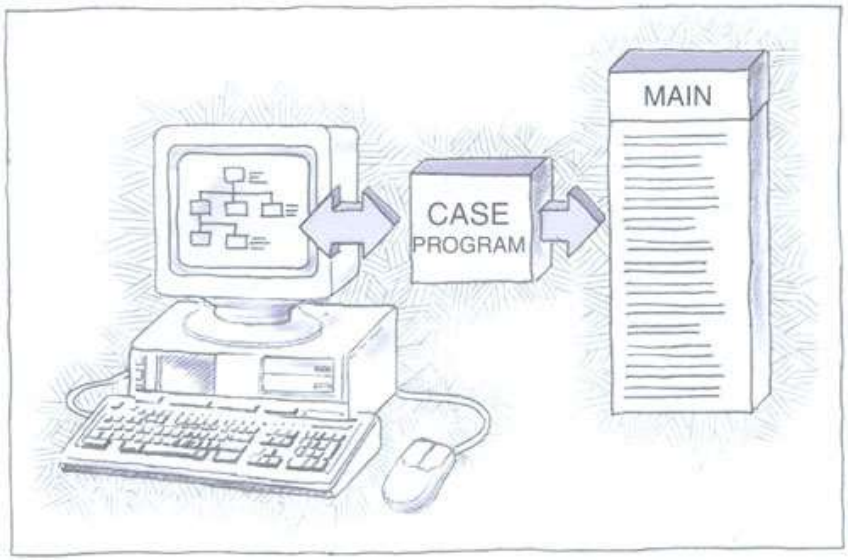
Programación Estructurada

- Surge en los 1960s
- Se basa en la descomposición funcional
- Un programa se divide en componenetes, y estos en subcomponentes, y así sucesidamente



Ingeniería de Software Asistida por Computadora

- Principal innovación en la programación estructurada
- Permite a las computadoras manejar el proceso de descomposición funcional, gráficamente define subrutinas en diagramas anidados y verifica que todas las interacciones entre subrutinas sigan correctamente la forma especificada
- Sistemas CASE avanzados generan automáticamente estructuras de programas completos a partir de estos diagramas, una vez que toda la información de diseño se ha ingresado



Lenguajes de Cuarta Generación

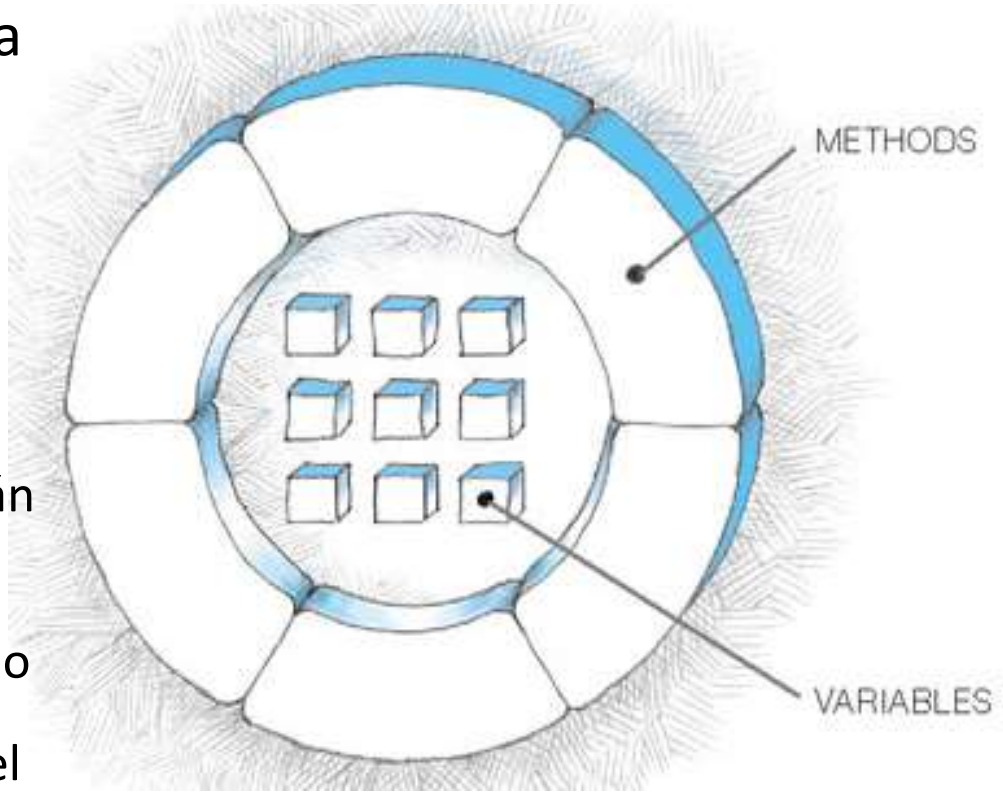
- Otro enfoque de programación automática
- Incluyen una amplia gama de herramientas para ayudar a automatizar la generación de aplicaciones de negocios de rutina, incluyendo la creación de formularios, informes y menús
- **Ventaja:** Personas que no son programadores los pueden utilizar
- **Desventaja:** Sólo puede generar programas muy simples y para problemas bien conocidos

¿Qué es Tecnología de Objetos?

- La definición estándar de la industria de Tecnología Orientada a Objetos, y puede resumirse en términos de tres conceptos clave
  - Los objetos proporcionan encapsulación de los procedimientos y datos
  - Los mensajes soportan el polimorfismo a través de objetos
  - Las clases implementan la herencia dentro de las jerarquías de clase

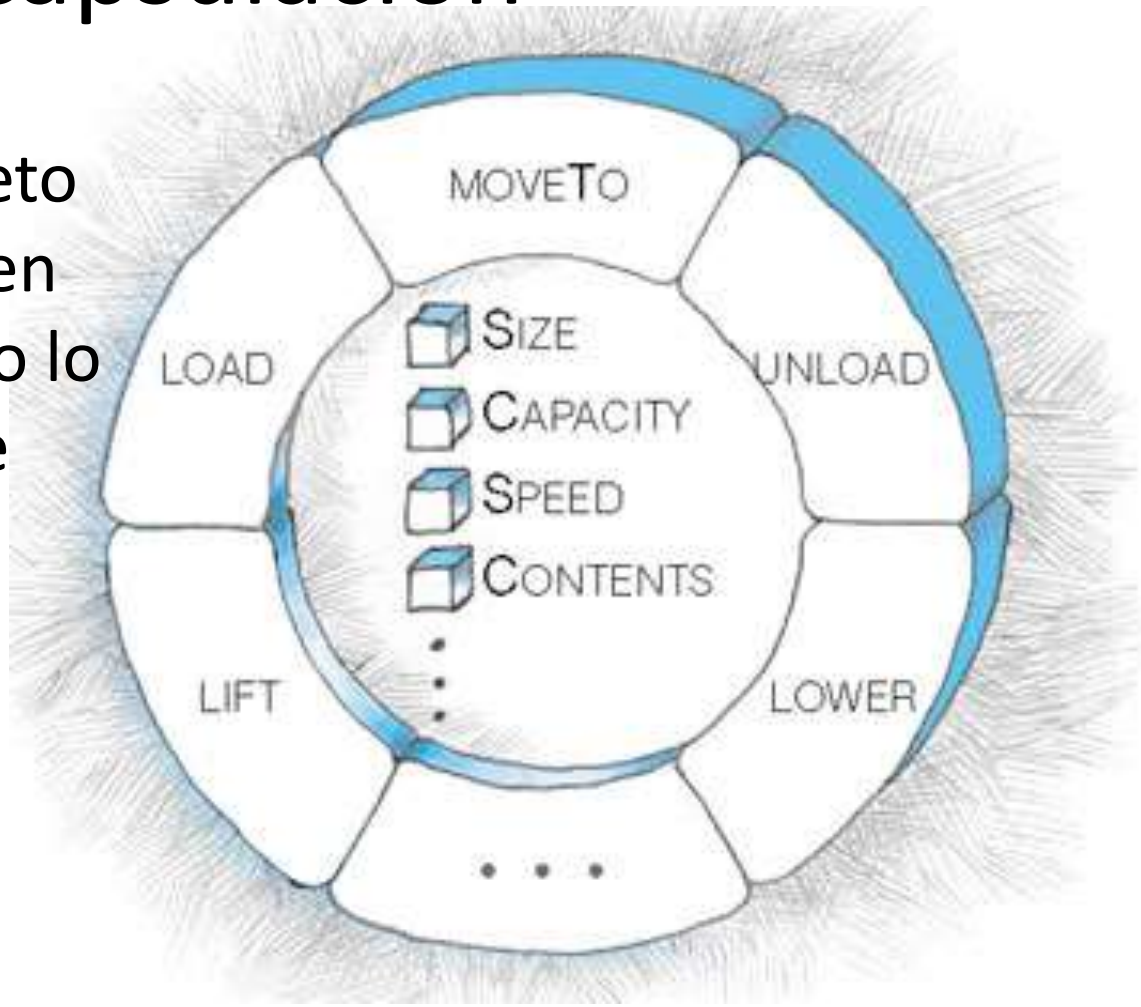
Encapsulación

- Un objeto es un paquete de software que contiene una colección de procedimientos y datos relacionados
  - Los procedimientos son llamados **métodos** para distinguirlos de los procedimientos convencionales que no están unidos a objetos
  - Los elementos de datos se refieren generalmente como **variables** debido a que sus valores pueden variar con el tiempo



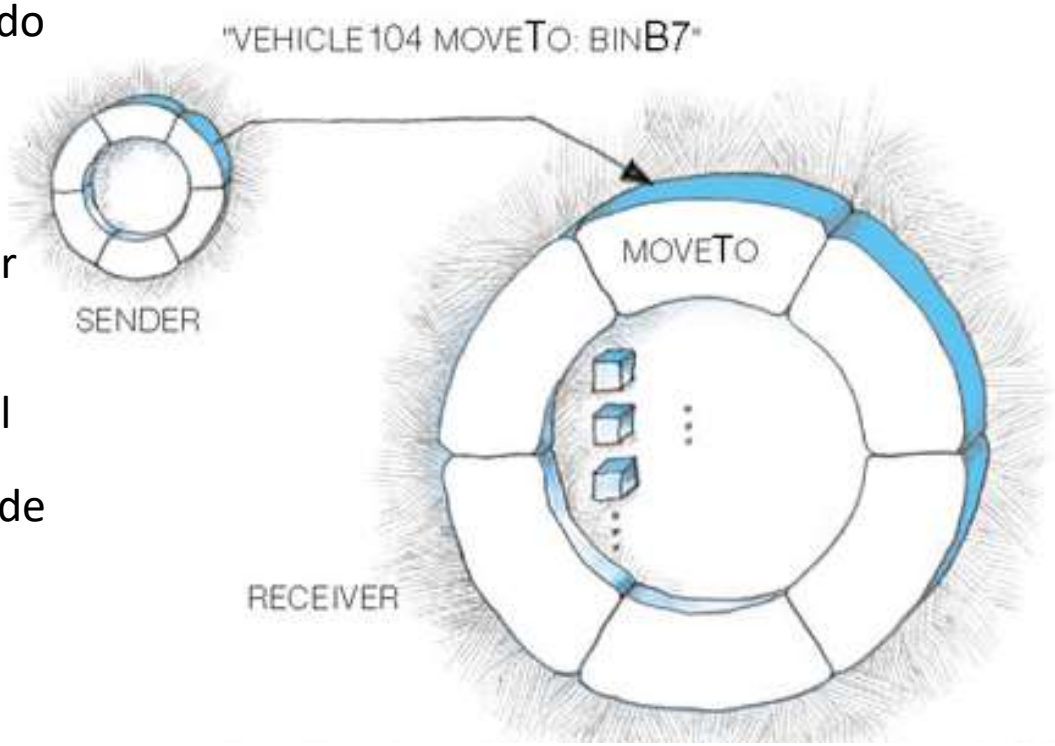
Encapsulación

- Todo lo que un objeto "sabe “ se captura en sus variables, y todo lo que puede hacer se expresa en sus métodos.



Mensajes y Polimorfismo

- Los objetos interactúan unos con otros, enviándose mensajes pidiendo a los objetos llevar a cabo sus métodos
- Un **mensaje** es el nombre de un objeto seguido del nombre de un método que el objeto sabe ejecutar
- Si un método requiere cualquier información adicional con el fin de conocer con exactitud qué hacer, el mensaje incluye esa información como una colección de elementos de datos llamados **parámetros**
- El objeto que inicia un mensaje se llama el **remitente**, y el objeto que recibe el mensaje se llama el **receptor**.

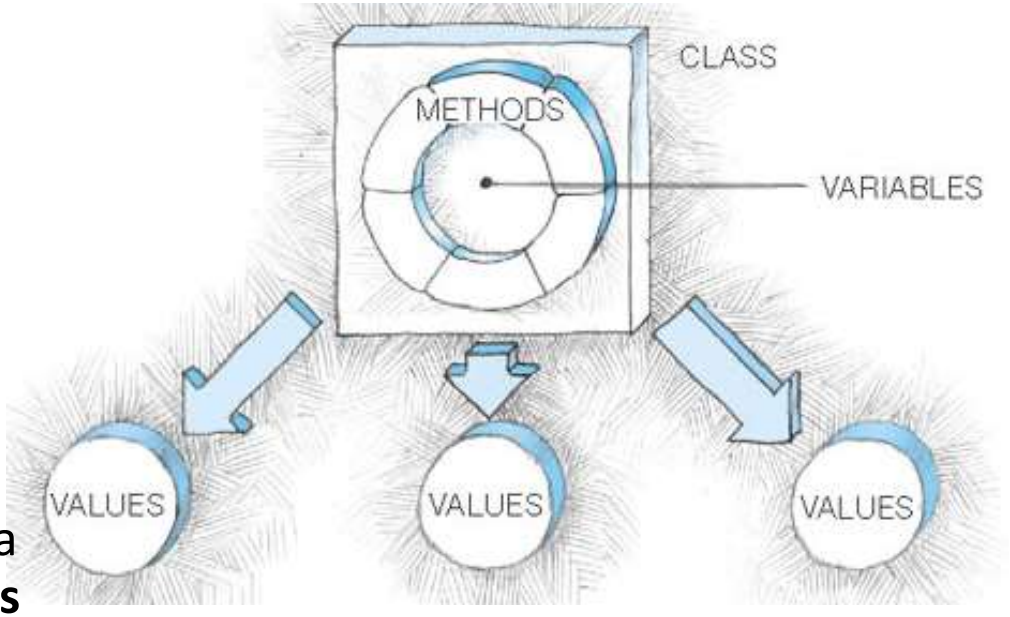


Mensajes y Polimorfismo

- Para que un mensaje tenga sentido, el remitente y el receptor deben ponerse de acuerdo sobre el formato del mensaje. Este formato se establece en **la firma del mensaje** que especifica el nombre del método a ejecutar y los parámetros que se incluirán.
- La capacidad de que diferentes objetos puedan responder al mismo mensaje, de diferentes maneras se llama **polimorfismo**

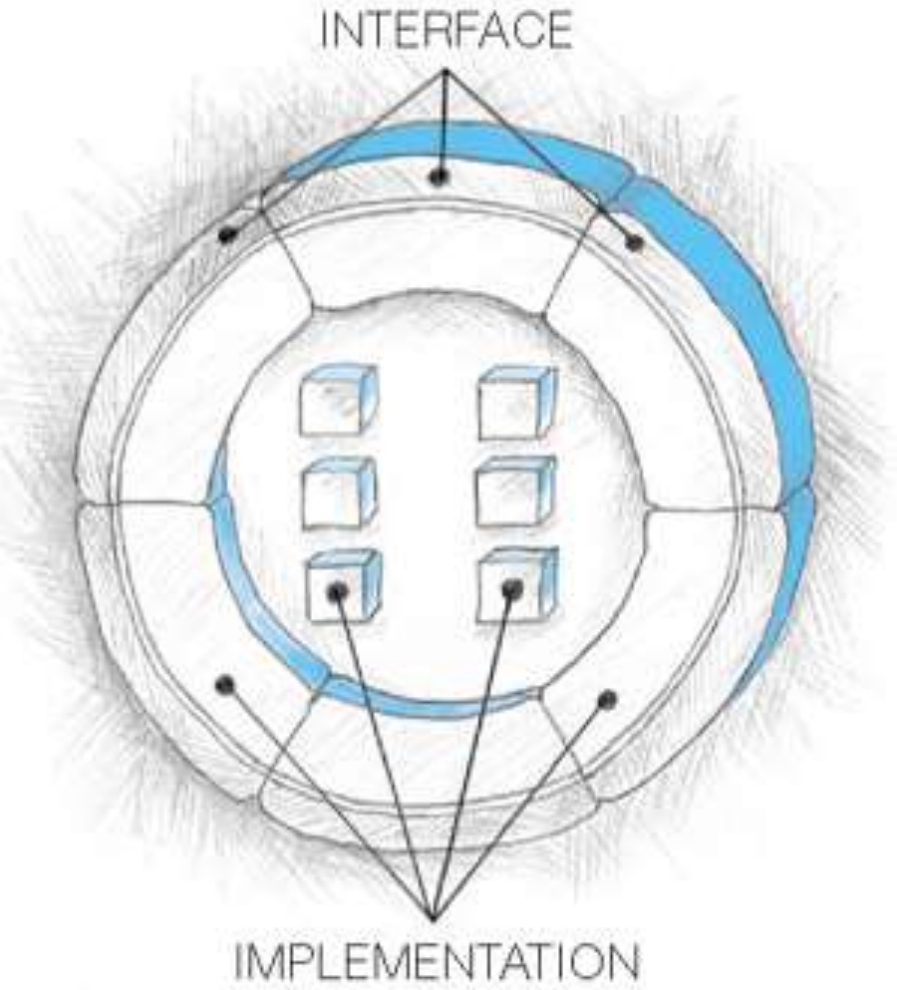
Clases y Herencia

- Una **clase** es una plantilla de software que define los métodos y las variables que se incluirán en un determinado tipo de objeto
  - Los métodos y las variables que componen el objeto se definen una sola vez, en la definición de la clase
  - Los objetos que pertenecen a una clase, llamados **instancias** de esa la clase, contienen sus propios valores particulares para las variables



Clases y Herencia

- El conjunto de mensajes que un objeto se compromete a responder se llama **interfaz del mensaje**.
- Esta interfaz se especifica como una colección de firmas de mensajes, cada uno de los cuales define el nombre y los parámetros para un mensaje concreto.
- El único requisito de diseño es que una clase proporcione un método para implementar cada mensaje especificado en su interfaz.
- Los detalles internos de la clase están totalmente ocultos detrás de esta interfaz y puede incluir cualquier número de variables, así como métodos "invisibles" que son utilizados sólo por el objeto en sí.

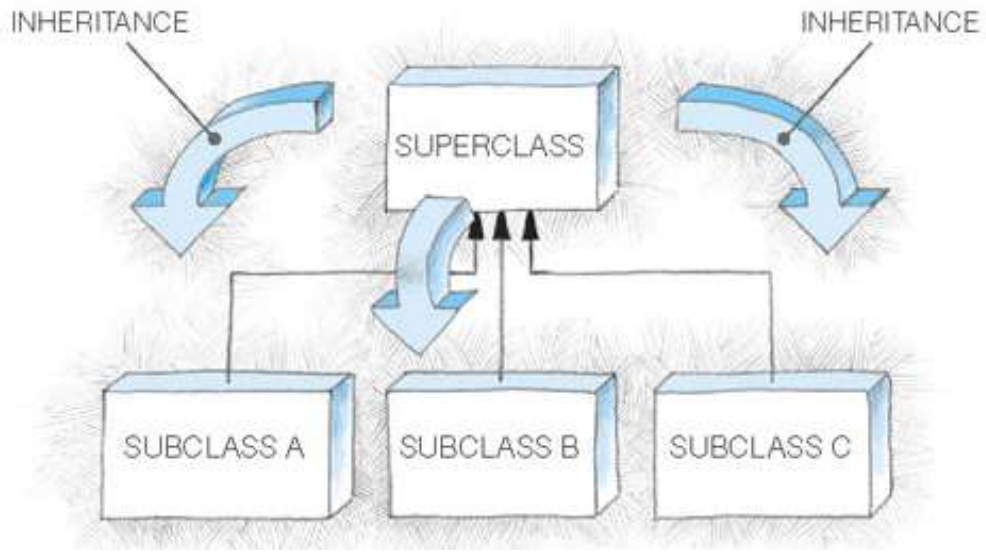


Clases y Herencia

- En resumen
  - Un **objeto** es una **instancia** de una **clase** determinada.
  - Los **métodos** y las **variables(Atributos)** se definen en la **clase**, y sus valores se almacenan en la **instancia**

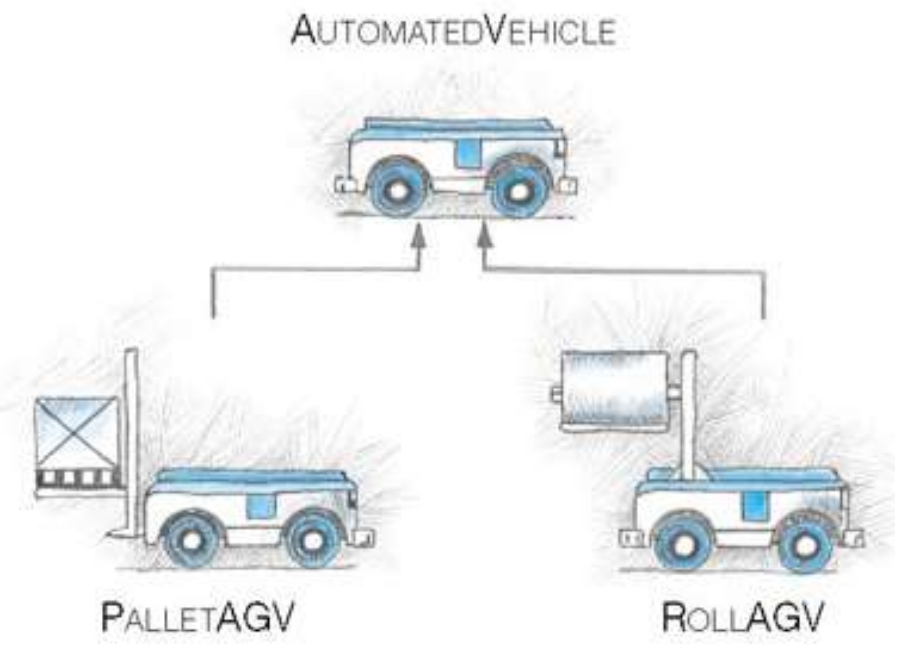
Clases y Herencia

- **Herencia.-** Definición de una clase a partir de la definición de otra clase.
  - **Subclases**.- clases heredan de una clase
  - **Superclase.-** clase que hereda a otras clases



Clases y Herencia

- Además de los métodos y las variables que heredan, las **subclases** pueden definir sus propios métodos y variables.
- También pueden redefinir cualquiera de los métodos heredados, una técnica conocida como de **overriding (sobrescribir)**.

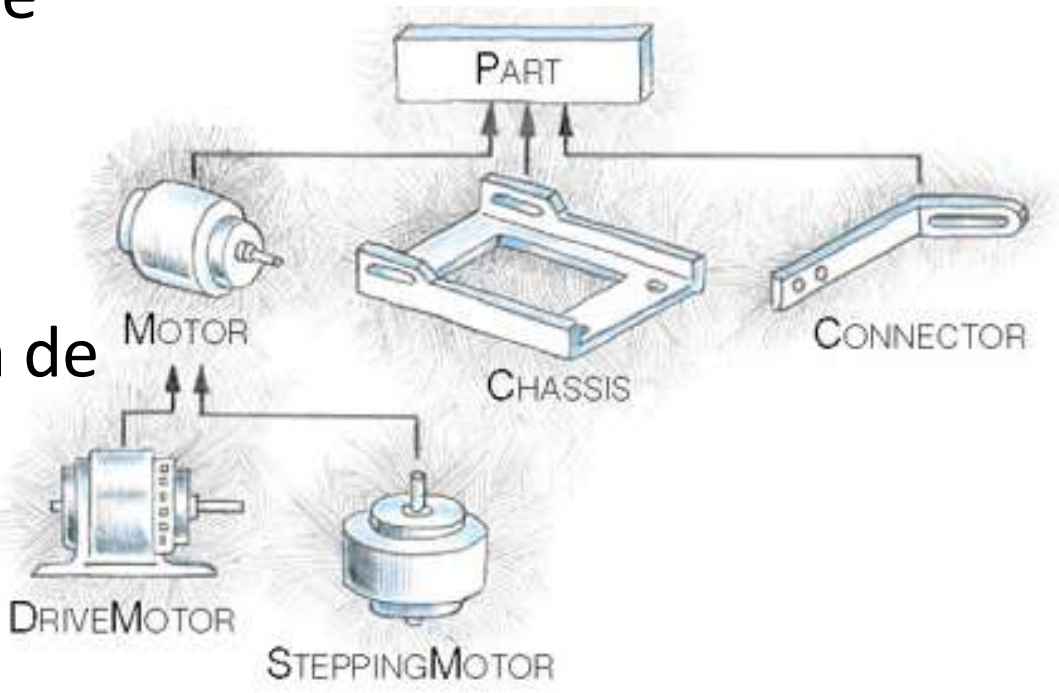


Clases y Herencia

- Dado que las clases definen interfaces de mensaje, estas interfaces son también heredados por sus subclases. Esto significa que todas las subclases de una clase dada están garantizados para responder a los mensajes que pueden ser manejados por la clase padre.

Clases y Herencia

- Las clases se pueden anidar en cualquier grado, y la herencia se acumulará automáticamente a través de todos los niveles. La estructura de árbol resultante se conoce como una **jerarquía de clases**.



Clases y Herencia

- Las jerarquías de clase aumentar la capacidad de los objetos para reflejar la forma en que vemos el mundo real.
  - Generalización y Especialización

Lenguajes Orientados a Objetos

- Muchos lenguajes de programación adicción las características del objeto, por lo que la lista de Lenguajes OO está en crecimiento
- Lo importante no es si un lenguaje es "verdaderamente" orientados a objetos, sino lo fácil que es aplicar los principios de objetos en el entorno que proporciona el lenguaje.

Lenguajes Orientados a Objetos

- Smalltalk
- C++
- Java
- Eiffel
- Object COBOL
- Visual
- Ada
- C#
- VB.NET
- Clarion
- Delphi
- Lexico (en castellano)
- Objective-C
- OCAML
- Oz
- PHP
- PowerBuilder
- Python
- Ruby
- Object Pascal
- CLIPS
- Actionscript
- PERL
- Simula
- ABAP
- Gambas

Aplicaciones de la Tecnología Orientada a Objetos

- Arquitecturas basadas en Componentes
  - Common Object Request Broker Architecture (CORBA) de Object Management Group (OMG)
  - Component Object Model (COM) de Microsoft

Aplicaciones de la Tecnología Orientada a Objetos

- Bases de Datos
  - Prototipos Experimentales
    - ORION
    - OpenOODB
    - IRIS
    - ODE
    - proyecto ENCORE/ObServer
  - Sistemas Comerciales
    - GEMSTONE/OPAL de ServicLogic
    - ONTOS de Ontologic
    - Objectivity de Objectivity Inc.
    - Versant de Versant Technologies
    - ObjecStore de Object Design
    - O2 de O2 Technology

Aplicaciones de la Tecnología Orientada a Objetos

- Objetos Distribuidos
  - Common Object Request Broker Architecture (CORBA) de Object Management Group (OMG)
  - Distributed Component Object Model (DCOM) de Microsoft
  - Remote Method Invocation (RMI) de Java, Sun Microsystems

Aplicaciones de la Tecnología Orientada a Objetos

- Análisis y Diseño
  - Lenguaje de Modelado Unificado (UML) de la OMG
  - Proceso Unificado Racional (RUP)