Evolución en la Programación

2) Lenguaje ensamblador (instrucciones con nombre): podían realizarse programas más largos y complejos usando una representación simbólica en vez de instrucciones de máquina.
3) Lenguajes de alto nivel: incorporan más herramientas para gestionar la creciente complejidad de los programas. El primero fue Fortran.
4) Programación estructurada (C y Pascal): Se pueden escribir de una forma sencilla programas moderadamente complejos.
5) Programación Orientada a Objetos: toma las mejores ideas de la programación estructurada y las combina con nuevos y poderosos conceptos que alientan a una nueva visión de las tareas de la programación.

Programación estructurada: ventajas

Con la programación estructurada, elaborar programas de computador sigue siendo una labor que demanda esfuerzo, creatividad, habilidad y cuidado. Sin embargo, con este estilo podemos obtener las siguientes ventajas:

1. Los programas son más fáciles de entender, ya que pueden ser leído de forma secuencial, sin necesidad de hacer seguimiento a saltos de línea (GOTO) dentro bloques de código para entender la lógica.
2. La estructura del programa es más clara puesto que las instrucciones están más ligadas o relacionadas entre sí.
3. Reducción del esfuerzo en las pruebas. El seguimiento de las fallas ("debugging") se facilita debido a la lógica más visible, por lo que los errores se pueden detectar y corregir más fácilmente.
4. Reducción de los costos de mantenimiento.
5. Programas más sencillos y más rápidos.
6. Los bloques de código son auto explicativos, lo que apoya a la documentación.

Programación estructurada: desventajas

- El principal inconveniente de este método de programación, es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático su manejo, esto se resuelve empleando la programación modular, definiendo módulos interdependientes programados y compilados por separado.
- La forma de resolver los problemas y organizar la información para realizar un sistema informático (datos por un lado y algoritmos por otro) está poco relacionada con la forma natural de observar y resolver el resto de las situaciones de la vida diaria.

Programación Orientada a Objetos

La programación orientada a objetos, plantea una nueva manera de pensar a la hora de resolver un sistema. Ya no se subdividen tareas complejas como un conjunto de tareas simples (sin tener en cuenta los datos), sino que se observa el problema y se lo subdivide en partes relacionadas, que contienen datos y código.

Se buscan (o diseñan) entidades independientes que colaboren en la resolución de un problema, cada una de estas entidades tendrá un conjunto de habilidades propias y responsabilidades designadas (cada entidad es responsable de algunas tareas).

Para resolver un sistema, se toma un conjunto de estas entidades y se las pone a interactuar entre si, enviando mensajes a cada una de ellas (entre ellas), proporcionando en cada mensaje la información necesaria para que la entidad pueda resolver la tarea y devolver una respuesta a quien le invocó.

Estas entidades, desde ahora “objetos”, colaboran entre si, comunicándose (enviándose mensajes) para poder resolver un problema complejo. Haciendo cada una de ellas una parte de la resolución, según sean sus habilidades o responsabilidades.

Estructurado vs. Objetos

Esto difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el **procesamiento de unos datos de entrada para obtener otros de salida**. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada se escriben funciones y después les pasan datos.

Los programadores que emplean lenguajes orientados a objetos definen **objetos con datos y métodos** y después envían mensajes a los objetos diciendo que realicen esos métodos en sí mismos.

Mirando la realidad como POO. Ejemplo

Enviando flores a mi tía que vive en España:

En este ejemplo el usuario del Sistema sería yo (actor) y el Sistema estaría formado por los siguientes objetos:

Florista en Mar del Plata.
Florista en España.
Transportista en España.
Mi tía en España.

Cómo se realiza un pedido:

1.- El actor (yo) envía el siguiente mensaje a la Florista de Mar del Plata: **enviarFlores.** Junto con el mensaje también envía la dirección en España, y una descripción del ramo.
2.- La florista de Mar del Plata se comunica con la de España y le envía a su vez el siguiente mensaje: **enviarFlores.** Junto con el mensaje también envía la dirección en España y una descripción del ramo. En definitiva delega el trabajo a otra persona.
3.- La florista de España prepara el pedido (ramo de flores), llama al transportista y le da el mensaje: **enviarPedido**. Junto con el mensaje le pasa el ramo de flores y la dirección. En definitiva, realiza parte de la tarea y delega otra parte al transportista.
4.- El transportista realiza la tarea de llevar las flores a mi tía de España.

Mirando la realidad como POO. Ejemplo

Cada uno de los objetos conoce su trabajo y lo realiza al recibir un mensaje. En nuestro ejemplo, **enviarFlores** es el mensaje enviado y también es el nombre de una función (método) propia del objeto **Florista de España**. Se envía un mensaje invocando a un método propio del objeto que recibe el mensaje. Los datos que recibe el mensaje (descripción del ramo y dirección) son los parámetros que se le pasan al método para que pueda ejecutar su tarea.

Desde el punto de vista del actor (quien inicia las acciones) solamente se ha comunicado con la Florista de Mar del Plata y desconoce la forma en que ésta solucionará su pedido.

Desde el punto de vista de la Florista de Mar del Plata, solamente se ha comunicado con la Florista de España y desconoce la forma en que ésta solucionará su pedido.

Desde el punto de vista de la Florista de España, solamente confecciona el pedido, se comunica con el Transportista y desconoce la forma en que el transportista solucionará el envío de su pedido.

El transportista, lleva su encomienda a destino, desconociendo su origen (pedido generado desde Mar del Plata) y tal vez hasta su

Mirando la realidad como POO. Ejemplo

El transportista, lleva su encomienda a destino, desconociendo su origen (pedido generado desde Mar del Plata) y tal vez hasta su contenido.

Ambas Floristas tienen algo en común, su actividad. O sea, su comportamiento es similar, son “objetos” que pertenecen a una misma clase (la clase de los Floristas).

El Transportista pertenece a otra clase distinta.

La forma en que cada uno de estos objetos lleva a cabo su actividad, es desconocida por los otros objetos, por ejemplo: cuántas flores posee en stock cada florista, la cantidad de móviles del transportista, la ruta empleada por el transportista, etc. A estos datos o actividades se los denomina **privados**, y a los demás **públicos**. Por ejemplo, el mensaje **enviarFlores** es de dominio público, ya que es un método que se usa para pasar mensajes entre objetos distintos (el actor a la florista de Mdp y a su vez, la florista de Mdp a la de España).

Este mismo tipo de enfoque y análisis de un ejemplo de la vida real, se pretende trasladar al análisis y resolución de un problema computacional.

Un ejemplo simple en C++

```
#include <iostream.h>
class cola
{
  int c[100];
  int ppio, fin;

  public:
  void ini();
  void meter(int i);
  int sacar(void);

};
```

```
void cola::ini()
{
  fin = ppio = 0;
}

void cola::meter(int i)
{
  if(ppio==100)
  {
     cout<< "la cola esta llena";
  }
  else
  {
     ppio++;
     c[ppio] = i;
  }
}
```

Los Objetos

- **¿Por qué Orientación a Objetos (OO)?**
  - Se parece más al mundo real
  - Permite representar modelos complejos
  - Muy apropiada para aplicaciones de negocios
  - Las empresas ahora sí aceptan la OO
  - Las nuevas plataformas de desarrollo la han adoptado (Java / .NET)

¿Qué es un Objeto?

- Informalmente, un objeto representa una entidad del mundo real
- **Entidades Físicas**
  - (Ej.: Vehículo, Casa, Producto)
- **Entidades Conceptuales**
  - (Ej.: Proceso Químico, Transacción Bancaria)
- **Entidades de Software**
  - (Ej.: Lista Enlazada, Interfaz Gráfica)

¿Qué es un Objeto?

- **Definición Formal (Rumbaugh):**
  - "Un objeto es un concepto, abstracción o cosa con un significado y límites claros en el problema en cuestión"
- **Un objeto posee (Booch):**
  - Estado
  - Comportamiento
  - Identidad

Un objeto posee Estado

Lo que el objeto sabe

- El estado de un objeto es una de las posibles condiciones en que el objeto puede existir
- El estado normalmente cambia en el transcurso del tiempo
- El estado de un objeto es implementado por un conjunto de propiedades (atributos), además de las conexiones que puede tener con otros objetos

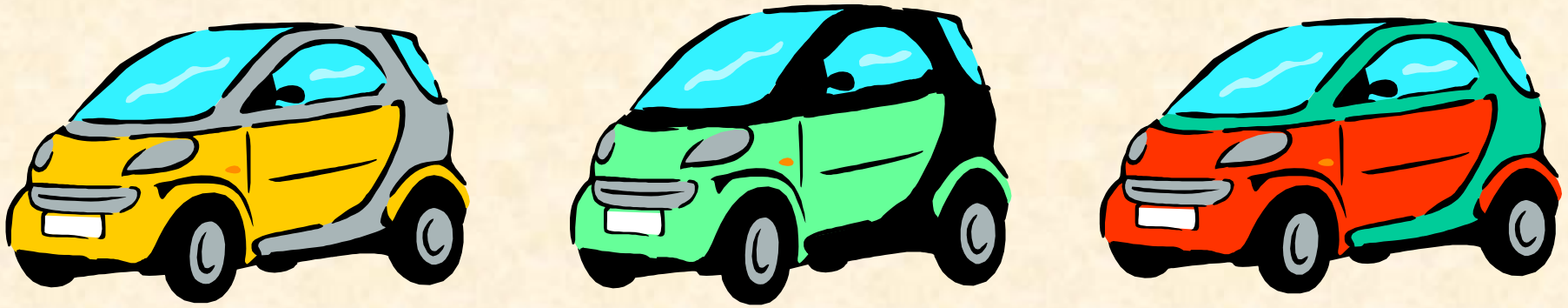
Un objeto posee Comportamiento

Lo que el objeto puede hacer

- El comportamiento de un objeto determina cómo éste actúa y reacciona frente a las peticiones de otros objetos
- Es modelado por un conjunto de mensajes a los que el objeto puede responder (operaciones que puede realizar)
- Se implementa mediante métodos

Un objeto posee Identidad

- Cada objeto tiene una identidad única, incluso si su estado es idéntico al de otro objeto

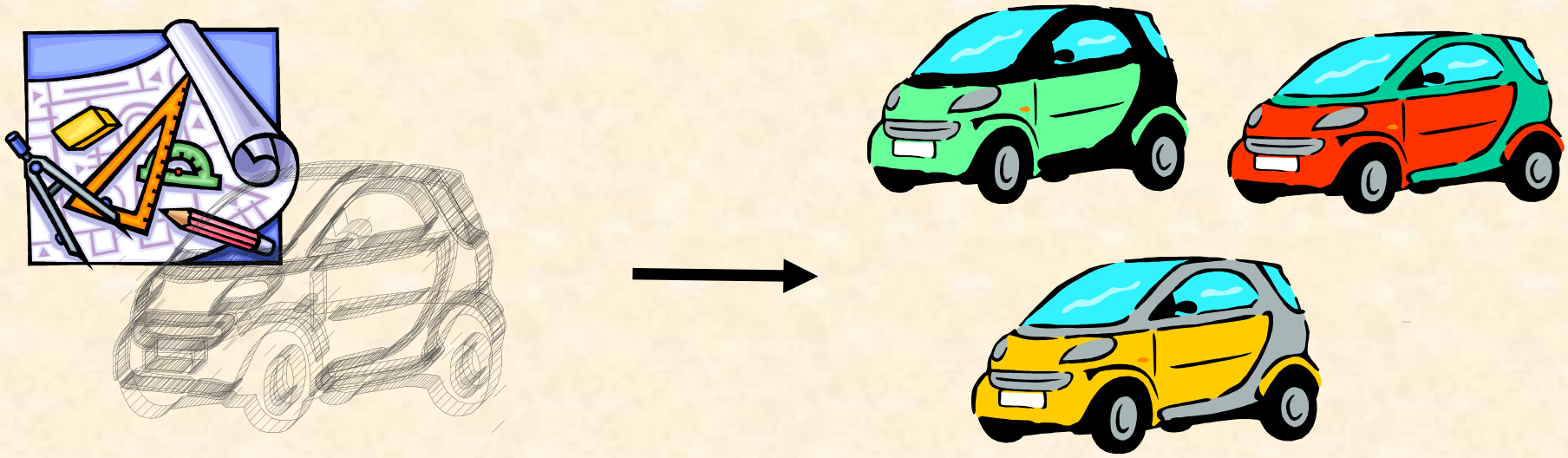


¿Qué es una Clase?

- Una clase es una descripción de un grupo de objetos con:
  - Propiedades en común (atributos)
  - Comportamiento similar (operaciones)
  - La misma forma de relacionarse con otros objetos (relaciones)
  - Una semántica en común (significan lo mismo)
- Una clase es una abstracción que:
  - Enfatiza las características relevantes
  - Suprime otras características (simplificación)
- Un objeto es una instancia de una clase

Objetos y Clases

- Una clase es una definición abstracta de un objeto
  - Define la estructura y el comportamiento compartidos por los objetos
  - Sirve como modelo para la creación de objetos
- Los objetos pueden ser agrupados en clases



La orientación a objetos es una manera natural de pensar acerca del mundo y de escribir programas de computadora.

Los objetos tienen atributos (como tamaño, forma, color, peso y aspecto) y exhiben un comportamiento.

Los humanos aprendemos acerca de los objetos mediante el estudio de sus atributos y la observación de su comportamiento.

Objetos distintos pueden tener muchos atributos iguales y exhibir un comportamiento similar.

La programación orientada a objetos (POO) modela objetos del mundo real con contrapartes en software.

Toma ventaja de las relaciones entre clases, en donde los objetos de cierta clase tiene las mismas características y comportamientos. Aprovecha las relaciones de herencia e incluso las relaciones de herencia múltiple, en donde las clases recién creadas se derivan mediante la herencia de las características de clases existentes, aunque contiene características únicas propias.

La programación orientada a objetos proporciona una manera intuitiva de ver el proceso de programación, a saber, mediante el modelado de objetos reales, sus atributos y sus comportamientos.

La POO también modela la comunicación entre objetos mediante mensajes.

La POO encapsula los datos (atributos) y las funciones (comportamiento) dentro de los objetos.

Los objetos tienen la propiedad de ocultar información. Aunque los objetos pueden saber como comunicarse entre sí a través de interfaces bien definidas, los objetos por lo general no están autorizados para saber los detalles de la implementación de otros objetos (para eliminar dependencias innecesarias).

El ocultamiento de información es crucial para una buena ingeniería de software.

En C y en otros lenguajes de programación por procedimientos, la programación tiende a ser orientada a acciones. En realidad, los datos son importantes en C, pero la visión es que los datos existen primordialmente para permitir las acciones que realizan las funciones.

Los programadores en C++ se concentran en la creación de sus propios tipos definidos denominados clases. Cada clase contiene datos, así como el conjunto de funciones que manipulan los datos. Los componentes de datos de una clase se denominan datos miembro. Los componentes de función de una clase se denominan funciones miembro o métodos.