Universidad Autónoma del Estado de México

Universidad Autónoma del Estado de México
Centro Universitario UAEM Valle de Teotihuacán

CUVT

Licenciatura en Ingeniería en Computación

Unidad de aprendizaje: Programación orientada a objetos

Clave	Horas teoría	Horas práctica	Total de horas	Créditos
L41007	3	2	5	8

Unidad de competencia II
"Conceptos básicos de la Programación Orientado a Objetos"

L.I.A. Cecilia Bibiana Ramírez Waldo

Septiembre 2019.

Tipo de Unidad de Aprendizaje	Carácter de la Unidad de Aprendizaje	Núcleo de formación	Modalidad
Curso Laboratorio	Obligatoria	Sustantivo	Presencial

Índice

Guion explicativo

- El estudiante conocerá los conceptos de la programación orientado a objetos y su implementación en un lenguaje apropiado, los cuales servirán de base para unidades de aprendizaje encaminadas al análisis, el diseño y la elaboración de aplicaciones informáticas

Objetivo de la unidad de aprendizaje

- Comprender los aspectos históricos y tecnológicos que dan importancia al desarrollo de software orientado a objetos.

Competencias genéricas

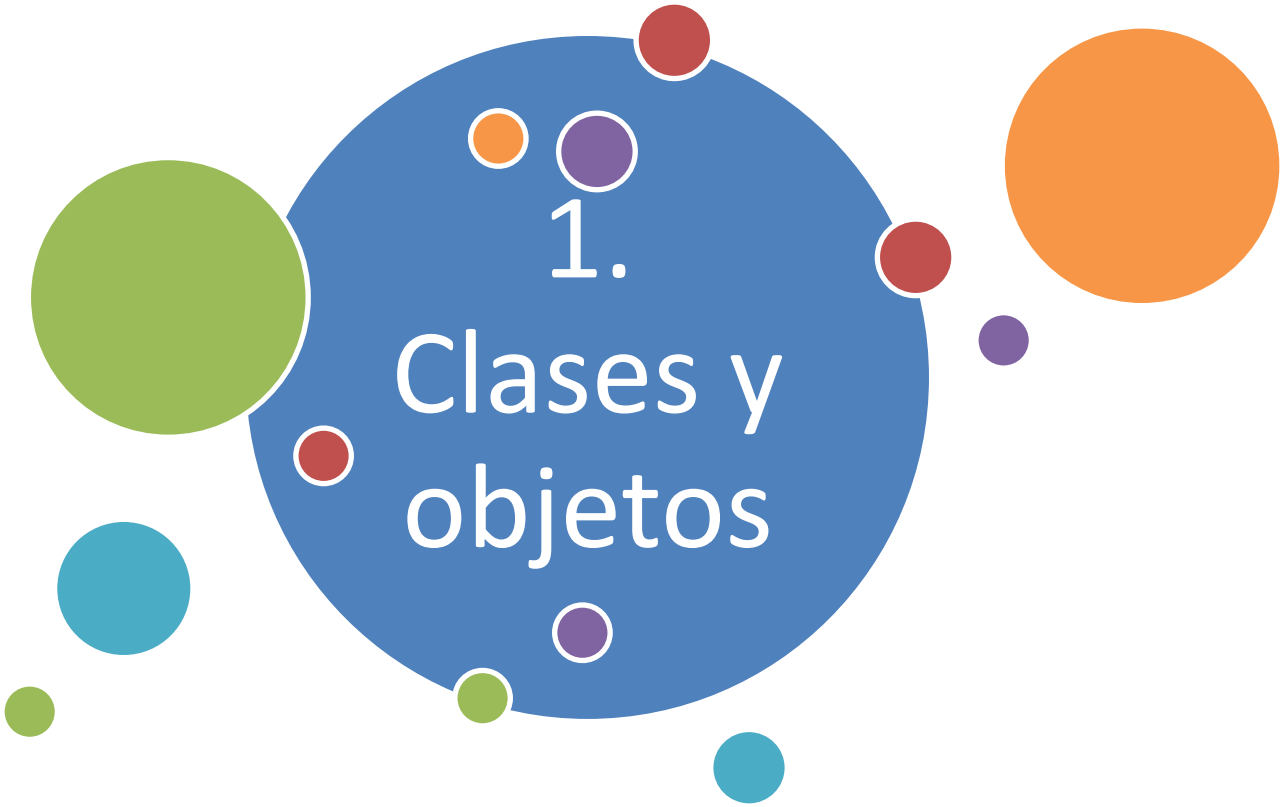
- Contar con bases para comprometerse con la calidad.
- Lograr una comunicación oral y escrita eficaz en su propia lengua y en una segunda lengua.
- Utilizar eficazmente al menos un lenguaje de programación orientado a objetos.
- Responder eficazmente a nuevas situaciones informáticas.
- Comunicarse con expertos de otras áreas.
- Aplicar los conocimientos en la práctica.
- Crear nuevas ideas para la solución de problemas.
- Dar mantenimiento a software.
- Evaluar sistemas.
- Desarrollar software.

Estructura de la unidad de aprendizaje

- 1. Comprender los aspectos históricos y tecnológicos que dan importancia al desarrollo de software orientado a objetos.
- 2. Implementar los conceptos básicos de la POO (encapsulamiento, herencia y polimorfismo).
- 3. Implementar los conceptos avanzados de la POO.
- 4. Utilizar un lenguaje de modelado estándar para lograr una documentación apropiada del software.
- 5. Instrumentar sistemas de software que mediante la aplicación de POO.

Secuencia didáctica

- Unidad 1. Paradigmas de la programación
- Unidad 2. Conceptos básicos de la POO
- Unidad 3. Conceptos avanzados de la POO
- Unidad 4. Lenguaje modelado
- Unidad 5. Interfaz gráfica de usuario


1.
Clases y objetos

Clases y objetos

Una clase es como una especie de patrón conceptual, mientras que un objeto es la materialización de dicho patrón.

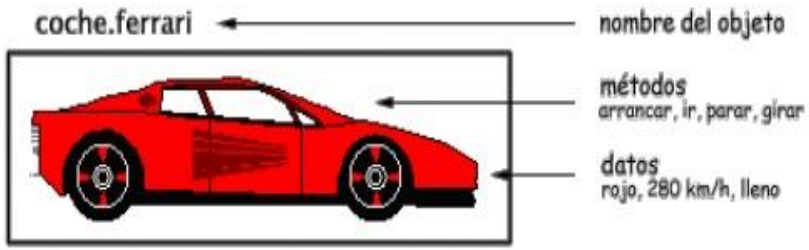
Clase sólo hay una, pero objetos puede haber muchos.

(Albert Gavarró Rodríguez)

Ejemplo de clases y objetos

- **Clase**: *Coche*

- **Objeto**: *Ferrari*



Encapsulamiento

- Es el proceso de almacenar en una misma sección los elementos de una abstracción que constituyen su estructura y su comportamiento; sirve para separar el interfaz contractual de una abstracción y su implantación (Dean, 2009).

NIVELES DE ENCAPSULAMIENTO

- **Existen tres niveles de acceso para el encapsulamiento, los cuales son:**
- **Público (Public):** Todos pueden acceder a los datos o métodos de una clase que se definen con este nivel, este es el nivel más bajo, esto es lo que tu quieres que la parte externa vea.
- **Protegido (Protected):** Podemos decir que estás no son de acceso público, solamente son accesibles dentro de su clase y por subclases.
- **Privado (Private):** En este nivel se puede declarar miembros accesibles sólo para la propia clase.

(Dean, 2009)

Atributos y métodos

- Los métodos son acciones que se pueden realizar a un objeto, estos pueden devolver un valor al acabar su ejecución.
- Los atributos, son las características que posee dicho obieto.

- Arrancar motor
- Parar motor
- Acelerar
- Frenar
- Girar a la derecha (grados)
- Girar a la izquierda (grados)
- Cambiar marcha (nueva marcha)

método

argumentos o parámetros

(Azpitarte, Marzo, 2009)

Modos de acceso

- **atributos y métodos privados** como miembros de una clase que no pueden ser accedidos desde otra clase, solo pueden serlo dentro de la propia clase. Estos miembros poseen el modificador de acceso **`private`**.

- **atributos y métodos públicos** como miembros de una clase que pueden ser accedidos desde otra clase. Estos miembros poseen el modificador de acceso publico friendly.
- Se considera el acceso como friendly cuando no se posee ningún modificador explicito (modificador de acceso por defecto)

(Azpitarte, Marzo, 2009)

Declaración de variables

- Una variable, en Java, es un identificador que representa un espacio de memoria que contiene información. La información es del tipo con que se declaro la variable. Hay 2 categorías de tipos para variables, los tipos primitivos (como int, short, byte, long, char, boolen y double) y las referencias a objetos (como String, Array y otros objetos).

Tipos de variables

Java tiene además, tres tipos de variables, que son:

- De instancia
- De clase
- Locales

Las variables de instancia, se usan para guardar los atributos de un objeto particular.
Las variables de clase, son variables que guardan el mismo valor, para todos los objetos de una clase determinada.
Las variables locales, se utilizan dentro de los métodos, en Java, las variables locales pueden declararse en el momento en que serán utilizadas, y una buena costumbre es inicializar las variables al momento de declararlas.

(Albert Gavarró Rodríguez)

Declaración de variables en java

- Para definir una variable en Java, basta con escribir un tipo, seguido del nombre que queremos dar a la variable.

```
int i;
Date date;
```

- Aquí se define la variable ***i*** del tipo primitivo entero *int* y la variable ***date*** de tipo *Date*. El punto y coma es el separador de instrucciones en Java.

Operadores

Operadores aritméticos	Operadores de asignación	Operadores relacionales	Operadores lógicos
+ suma - resta * multiplicación / división % residuo ++var preincremento --var predecremento var++ postincremento var-- postdecremento	= asignación += asignación de suma -= asignación de resta *= asignación de multiplicación /= asignación de división %= asignación de residuo	< menor que <= menor o igual que > mayor que >= mayor o igual que == igual a != no igual	&& AND || OR ! NOT ^ OR exclusivo

Objetos

- Un objeto es una unidad dentro de un programa de computadores que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución. (Albert Gavarró Rodríguez)

Objetos en java

- Cuando se crea un objeto en Java, como en "new Date()" , el objeto se guarda en una parte de la memoria llamada *Heap*. Cuando asignamos el objeto a una variable como en "date = new Date()", lo que guardamos en *date* es la dirección de memoria *Heap* donde está el objeto. (Dean, 2009)

```
int i1 = 5;
int i2 = i1;
Date date1 = new Date();
Date date2 = date1;
```

Mensajes

- Los objetos se comunican e interacción entre si por medio de mensajes.
- Si un objeto desea que otro haga algo le envía un mensaje que puede tener información adicional en forma de parámetros.
- Cuando un objeto recibe un mensaje ejecutara un método u operación. (Dean, 2009)

Componentes de un mensaje

- Objeto destinatario del mensaje (miCoche).
- Método que se debe ejecutar como respuesta (cambiar marcha).
- Parámetros necesarios del método (segunda) (Dean, 2009)

Constructores

- Es un método que se ejecuta automáticamente cuando se crea un objeto de una clase.
- Si no existe se crea un constructor por defecto sin parámetros.
- Si la clase tiene algún constructor, el constructor por defecto deja de existir

Reglas:

- El constructor tiene el mismo nombre que la clase.
- Puede tener cero o mas argumentos.
- No tiene tipo de retorno. (Dean, 2009)

Ejemplo de constructor

- La clase rectángulo contiene un constructor con cuatro parámetros.
- Al crear un objeto, se pasan los valores de los argumentos al constructor con la misma sintaxis que la de llamada aun método.

La clase `Rectangulo` *tiene un constructor con cuatro parámetros.*

```
public class Rectangulo
{
   private int izdo;
   private int superior;
   private int dcha;
   private int inferior;
   // constructor
   public Rectangulo(int iz, int sr, int d, int inf)
   {
      izdo = iz;
      superior = sr;
      dcha = d;
      inferior = inf;
   }
   // definiciones de otros métodos miembro
}
```

Constructor por Defecto

- Muchos lenguajes de OOP, permiten definir una clase sin crear un constructor para la clase. El leguaje, entonces, utiliza el constructor por defecto (interno al lenguaje) para crear objetos de esa clase. Este método se limita a reservar el espacio de memoria necesario para almacenar los datos del objeto, pero estos datos no están inicializados o no lo están correctamente, ya que el constructor por defecto no puede saber qué valores son los apropiados para los datos de la clase (Dean, 2009).

Ejemplo de constructor por defecto

- Java crea automáticamente un constructor por defecto cuando no existen otros constructores.

- Tal constructor inicializa las variables de tipo numérico (int, float ...) a cero, las variables de tipo boolean a true y las referencias a null.

```
public class Punto
{
   private int x;
   private int y;

   public Punto()     // constructor por defecto
   {
      x = 0;

      Estructuras de datos en Java

      y = 0;
   }
}
```

Constructor Sobrecargado

- Al igual que se puede sobrecargar un método de una clase, también se puede sobrecargar el constructor de una clase. De hecho, los constructores sobrecargados son bastante frecuentes y proporcionan diferentes alternativas para inicializar objetos.

(Dean, 2009)

Bloques inicializadores

- Java dispone de inicializadores para evitar que haya variables sin inicializar, estos pueden ser static (para la clase) o de objetos (Jalon, 2000).

Inicializador static

- Es algo parecido aun método, que se llama automáticamente al crear la clase.
- También se diferencia del constructor en que no es llamado para cada objeto, sino una sola vez para toda la clase.
- Se crean dentro de la clase, como métodos sin nombre, sin argumentos y sin valor de retorno.
- Pueden utilizar para dar valor a las variables static, también conocidos como variables nativos (Jalon, 2000)

Inicializadores de objeto

- Estos inicializadores no utilizan la palabra static. Se utilizan para las clases anónimas, por no tener nombre no pueden tener constructor.
- Permiten asignar valores a cualquier campo o propiedad accesible de un objeto en el momento de su creación sin tener que invocar un constructor seguido de líneas de instrucciones de asignación. La sintaxis de inicializador de objetos permite especificar argumentos para un constructor u omitir los argumentos

(Jalon, 2000)

Firma de métodos

- Es la combinación del nombre y los tipos de los parámetros o argumentos.
- Los métodos sobrecargados son dos o más métodos en la misma clase que utilizan el mismo nombre. Puesto que utilizan el mismo nombre, el compilador necesita algo más aparte del nombre con el fin de poder distinguirlos. Para distinguir dos métodos sobrecargados, éstos se definen con un número de parámetro diferente o con tipos de parámetros diferentes. La combinación del nombre del método, el número de sus parámetros, y los tipos de sus parámetros se denomina firma del método. Cada método diferente tiene su firma diferente.

(Dean, 2009)

Ocultamiento de información

- Se basa en la posibilidad que tiene los objetos de restringir el acceso a si conocimiento, por parte de otros objetos, o de otra forma.
- Definir cuales son los criterios y los mecanismos para garantizar dicho acceso a quienes a si lo requieran.
- Reduce la cantidad de detalle que deben recordar o comunicar los programadores. (Dean, 2009)

Visibilidad

- Puede haber distintos niveles de visibilidad para los miembros (atributos y métodos) de una clase.
- Existen cuatro tipos de modificadores:
  - Public: Se puede acceder desde cualquier lugar.
  - Private: solo se puede acceder desde la propia clase.
  - Protectec: solo se puede acceder desde la propia clase o desde una clase que herede de ella.
  - Por defecto (package): Se puede acceder desde las clases perteneciente al mismo paquete. (Dean, 2009)

Alcance de los identificadores

- Al escribir código fuente, los programadores tienen que dar nombres a diferentes elementos del programa: variables, constantes, objetos, métodos.
- Para ello se usan identificadores. En Java, los identificadores se forman con letras mayúsculas y minúsculas (Java las distingue), dígitos y también los caracteres dólar y guion bajo, $,- . Un identificador valido debe comenzar siempre con una letra, y no puede coincidir con una palabra reservada del lenguaje. (Azpitarte, Marzo, 2009)

Estado de un objeto

- El estado de un objeto se refiere al conjunto de atributos y sus valores en un instante de tiempo dado. El comportamiento de un objeto puede modificar el estado de este. Cuando una operación de un objeto modifica su estado se dice que esta tiene "efecto colateral".

(Azpitarte, Marzo, 2009)

Copia vs clonación de objetos

Clonación

- La clonación de objetos se refiere a la creación de una copia exacta de un objeto. Crea una nueva instancia de la clase de objeto actual e inicializa todos sus campos con exactamente los contenidos de los campos correspondientes de este objeto.

Copia

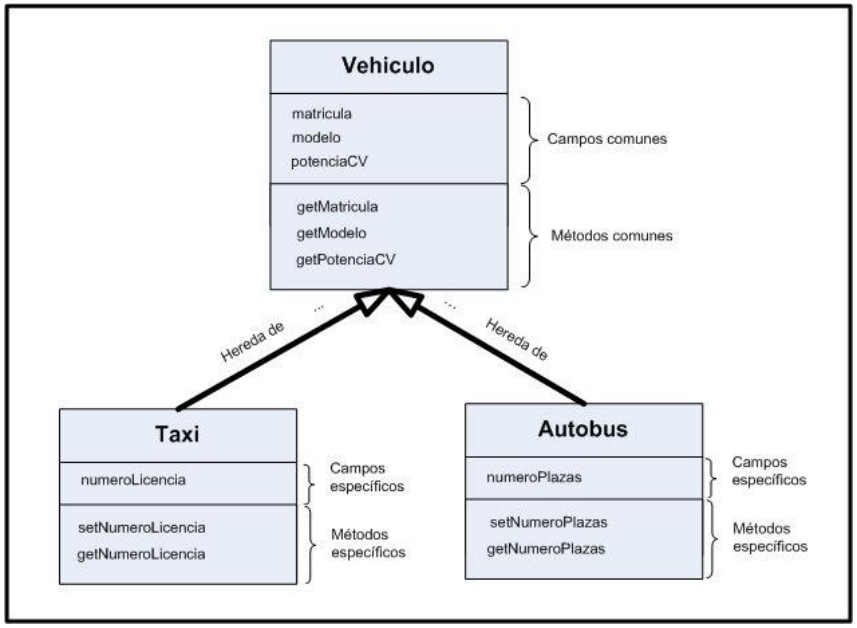
- Construye un objeto como una copia del que se le pasa .
- No se usa mucho dentro de las bibliotecas de clase de java
- Existe la clase String y las colecciones.
- La forma preferida de obtener la copia de un objeto es utilizar el método clone. (Dean, 2009)


2.
Herencia

Herencia

Concepto:

- Se puede definir como clases basadas en clases existentes, con el fin de reutilizar el código previamente desarrollado, generando una jerarquía de clases dentro de la aplicación.



Implementación

- La sentencia *"extends"* permite implementar el concepto de herencia. Se incluye para que una clase herede a otra clase.

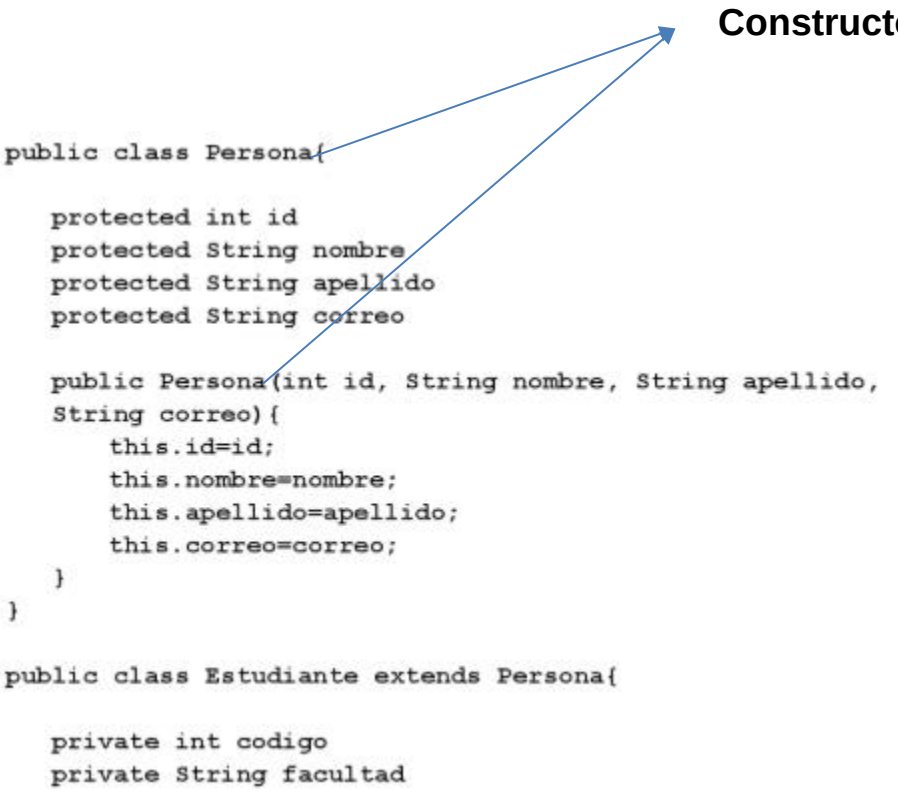
- Ejemplo: En el caso de jerarquía sel personal académico, debe existir una clase *persona* y una clase *estudiante*.
- Al implementar la clase *estudiante* se le debe incluir la sentencia ***extends*** para que herede de la clase *persona* . La sintaxis es:

```
public class Persona{
    ...
}

public class Estudiante extends Persona{
    ...
}
```

Implicaciones contructores

- Al implementar un constructor de la clase *estudiante* que pueden acceder a los atributos, y asignar valores y hacer un llamado al constructor de la clase *persona* enviándole los parámetros definidos en dicha clase.



El uso de *super*

- Dicha sentencia es utilizada para acceder a métodos implementados en la clase superior en el concepto de herencia.

```
public class Estudiante extends Persona{

    private int codigo
    private String facultad

    public Estudiante(int id, String nombre, String apellido,
    String correo, int codigo, String facultad){

    super(id, nombre, apellido, correo);
    this.codigo=codigo;
    this.facultad=facultad;
  }
}
```

Polimorfismo

- Consiste en poder acceder a diferentes servicios en tiempo de ejecución sin necesidad de implementar diferentes referencias a objetos..

- Considera una jerarquía herencia de *FigurasGeométricas* de la siguiente *figura* es posible hacer uso del concepto polimorfismo de la siguiente manera:

```
FiguraGeometrica figura;
figura = new Circulo(5);
figura = new Cuadrado(5);
figura = new Rectangulo(5,2);
figura = new Triangulo(5,2);
figura = new Cubo(5);
```

Conclusiones

- Los conceptos básicos de la programación orientada a objetos nos permite generar la habilidad y capacidad de desarrollar programas que permitan brindar soluciones a situaciones reales utilizando el lenguaje de Programación.

Por su atención, gracias.

Referencias bibliográficas

- Azpitarte, R. L. (Marzo, 2009). Introducción a la Programación Orientada a Objetos con Java. Valencia: Rev. 1.0.1.
- Dean, J. S. (2009). Introducción a la programación con JAVA. México, D.F: McGRAW-HILL.
- Jalon, J. G. (2000). Aprenda Java como si estuviera en primero. San sebastian : San sebastian .