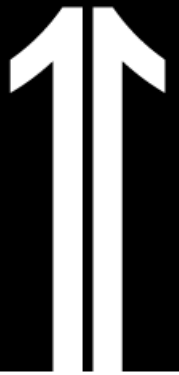
TÉCNICAS DE PROGRAMACIÓN ORIENTADA A OBJETOS

02/04/2024 y 04/04/2024

UPN.EDU.PE

LOGRO DE SESIÓN



Al término de la sesión el estudiante analiza los conceptos fundamentales de el lenguaje unificado de modelado UML para el desarrollo de aplicaciones java , su representación en un diagrama de clases, aplicando su razonamiento lógico en el desarrollo de casos básicas.

TEMARIO

- Constructores y Destructores.
- Métodos De Instancia, Métodos De Clase.
- Categorías De Métodos: Gestión, Implementación.

Introducción al Lenguaje Unificado de Modelado UML

- Introducción al Lenguaje Unificado de Modelado UML
- Representacióndeclases.
- Diagrama de clases.
- Sobrecarga de métodos.
- Sobrecargade operadores.

PROGRAMACIÓN ORIENTADA A OBJETOS
POO

- En Programación Orientada a Objetos (POO), un programa está hecho de clases, con sus atributos y métodos.

- La creación de un programa involucra ensamblar objetos y hacerlos interactuar entre ellos.

CLASES Y OBJETOS

- Una clase describe un grupo de objetos que comparten propiedades y métodos comunes.
- Una clase es una plantilla que define qué forma tienen los objetos de la clase.

CLASES Y OBJETOS

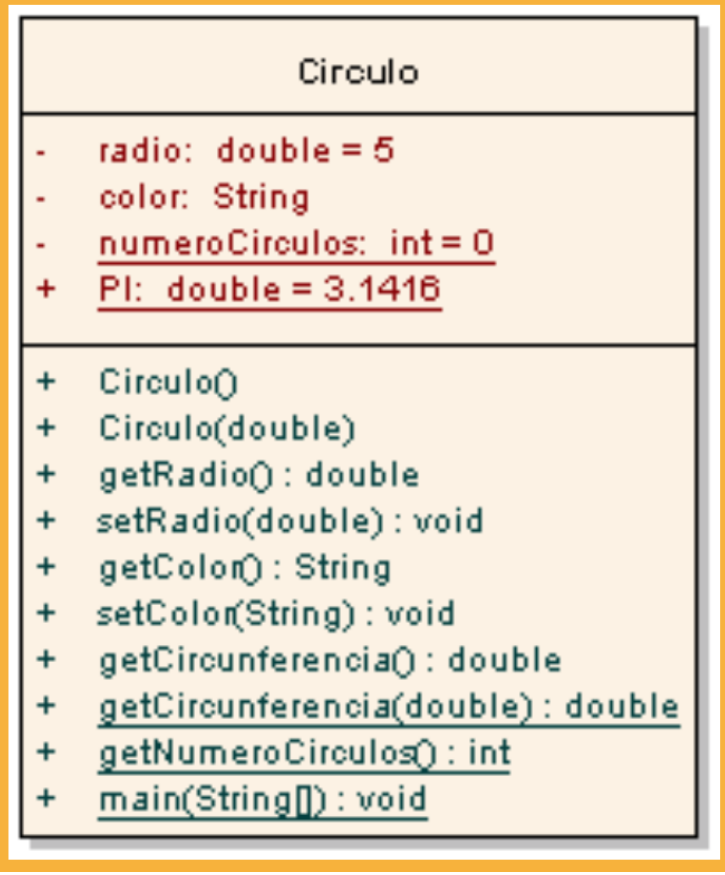
- Una clase se compone de :
    - Información: Atributos (atributos, propiedades).
    - Comportamiento: métodos (operaciones, funciones).
- Un objeto es una instancia de una clase.

CLASES Y OBJETOS

Clase	Objeto
Empresa	Sodimac
Sala	La Moneda
Empleado	Juan Pérez
Ventana (tiempo de diseño)	Ventana (tiempo de ejecución)
String	“Juan Pérez”

DEFINICIÓN DE UNA CLASE

```
class Circulo {
    // Atributos
    // métodos
    // constructores
}
```



ATRIBUTOS

- Los objetos almacenan información en sus Atributos.
- Existen dos tipos de Atributos: de instancia y static (de clase).

ATRIBUTOS DE INSTANCIA

- Hay una copia de un campo de instancia por cada objeto de la clase.
- El campo de instancia es accesible a través del objeto al que pertenece.

ATRIBUTOS STATIC (DE CLASE)

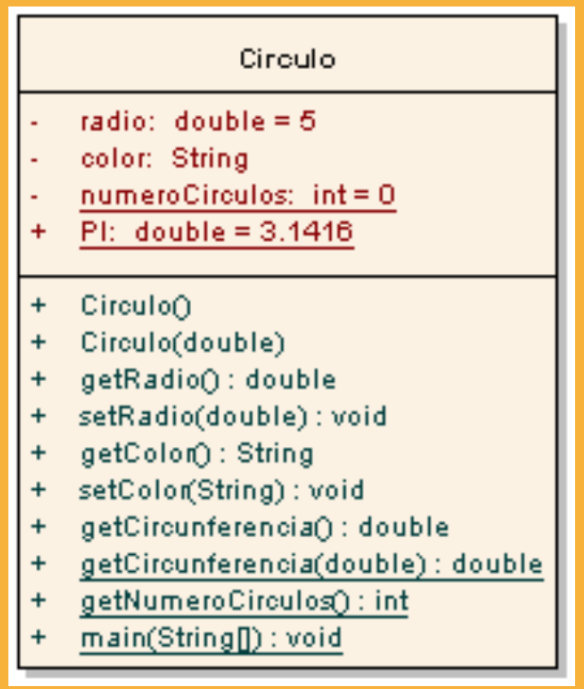
- Hay una única copia de un campo static en el sistema (equivalente a lo que en otros lenguajes es una variable global).
- El campo static es accesible a través de la clase (sin necesidad de instanciar la clase).

ATRIBUTOS

radio y color son variables de instancia, hay una copia de ellas por cada objeto Circulo

```
class Circulo {
    // Atributos
    private double radio = 5;
    private String color;
    static int numeroCirculos = 0;
    static final double PI = 3.1416;
    // métodos
    // constructores
}
```

numeroCirculos y PI son variables static, están sólo una vez en memoria; PI además es constante (final): no puede modificarse



ACCESO A ATRIBUTOS

- Acceso a variables de instancia: se utiliza la sintaxis "objeto."

```
Circulo c1 = new Circulo();
c1.radio = 5;
c1.color = "rojo";
// si c1 es null,
// se genera una excepción NullPointerException
```

ACCESO A ATRIBUTOS

- Acceso a variables static: se utiliza la sintaxis "clase."

```
Circulo.numeroCirculos++;
System.out.println(Circulo.PI);
```

MÉTODOS

- Instrucciones que operan sobre los datos de un objeto para obtener resultados.
- Tienen cero o más parámetros.
- Pueden retornar un valor o pueden ser declarados void para indicar que no retornan ningún valor.

MÉTODOS

- Pueden ser de instancia o static:
- Un método de instancia se invoca sobre un objeto de la clase, al cual tiene acceso mediante la palabra this (y sus variables de instancia son accesibles de manera directa).
- Un método static no opera sobre un objeto de la clase, y la palabra this no es válida en su interior.

MÉTODOS

- Sintaxis

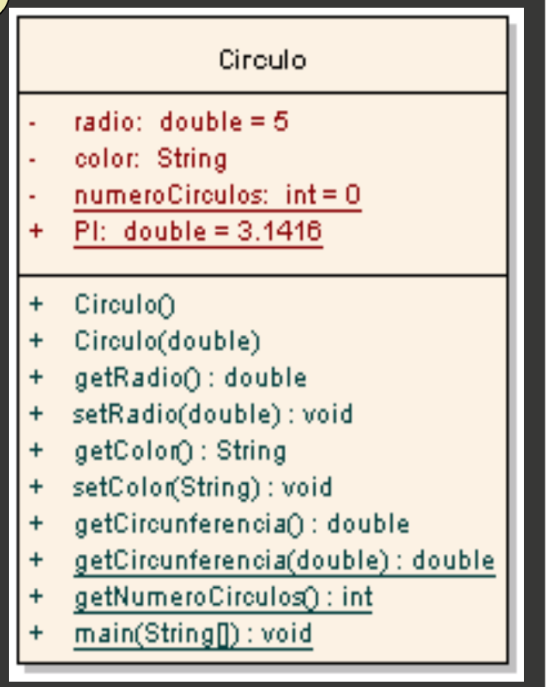
  [static] <tipo retorno> <nombre método> (<tipo> parámetro1, ...) {
  // cuerpo del método
  return <valor de retorno>; }

MÉTODOS

```
class Circulo {
    // Atributos
    double radio = 5;
    String color;
    static int numeroCirculos =
    static final double PI = 3    16;
    // métodos
    double getCircunferencia() {
        return getCircunferencia(radio);
    }
    static double getCircunferencia(double r) {
        return 2 * r * PI;
    }
    // constructores
}
```

Método de instancia, tiene acceso directo a las variables de instancia del objeto sobre el que se invoca

Método static, no tiene acceso directo a variables de instancia



ACCESO A MÉTODOS

- Acceso a Atributos y métodos de instancia: se utiliza la sintaxis "objeto.“

```
Circulo c1 = new Circulo(); c1.radio = 5;
c1. color = "rojo";
double d = c1.getCircunferencia(); // Si c1 es null

// se genera una excepción NullPointerException;
```

ACCESO A MÉTODOS

- Acceso a Atributos y métodos static: se utiliza la sintaxis "clase.“ Circulo.numeroCirculos++;

  int n = Circulo.getNumeroCirculos(); System.out.println(Circulo.PI);

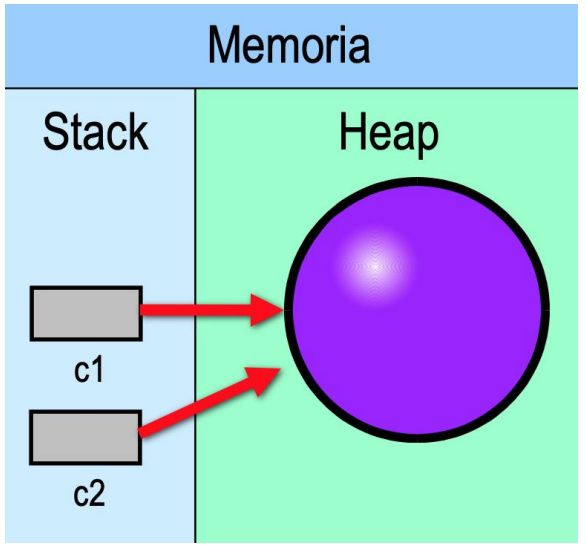
SOBRECARGA (OVERLOAD) DE MÉTODOS

- Métodos de una clase pueden tener el mismo nombre pero diferentes parámetros.

- Cuando se invoca un método, el compilador compara el número y tipo de los parámetros y determina qué método debe invocar.

INSTANCIACIÓN Y REFERENCIAS

- Los objetos se crean con el operador new, y se manejan mediante referencias.
- Los objetos se crean en el área de memoria dinámica conocida como el heap.
- Una referencia contiene la dirección de un objeto (es similar a los punteros de otros lenguajes).
- Una asignación entre objetos es una asignación de referencias.




```
Circulo c1 = new Circulo();
Circulo c2 = c1;
```

PASO DE PARÁMETROS

En Java el paso de parámetros se realiza "por valor”.

- Argumentos de tipos primitivos
- Si un método modifica el valor de un parámetro, este cambio sólo ocurre al interior del método; al retornar el método, se mantiene el valor original.

PASO DE PARÁMETROS

- Argumentos de tipo referencia (objetos).
    - Al retornar el método, la referencia pasada como parámetro sigue referenciando al mismo objeto; sin embargo, los Atributos del objeto podrían haber sido modificados por el método

CONSTRUCTORES

Un constructor es un método especial invocado para instanciar e inicializar un objeto de una clase.

- Invocado con la sentencia new
- Tiene el mismo nombre que la clase
- Puede tener cero o más parámetros
- No tiene tipo de retorno, ni siquiera void
- Un constructor no público restringe el acceso a la creación de objetos.

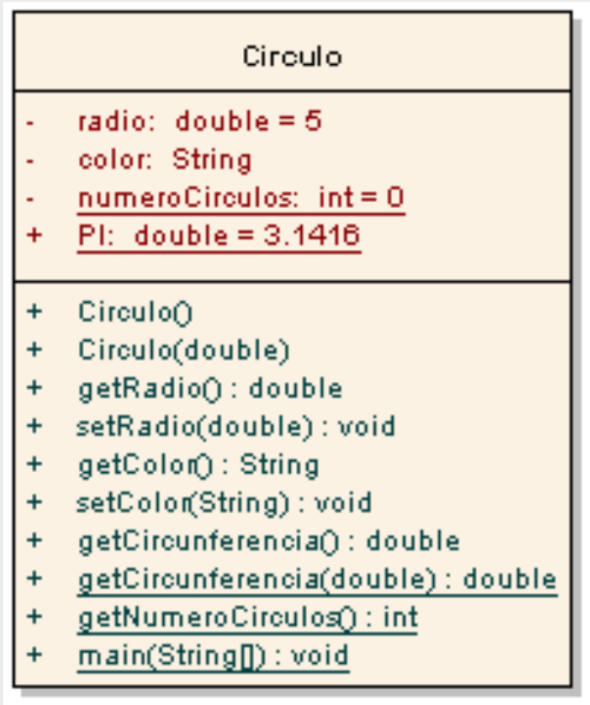
CONSTRUCTORES

- Si la clase no tiene ningún constructor, el sistema provee un constructor default, sin parámetros.

- Si la clase tiene algún constructor, debe usarse alguno de los constructores definidos al instanciar la clase (el sistema no provee un constructor default en este caso).

EJEMPLO:
USO DE CONSTRUCTORES

```
class Circulo {
   ...
   // constructores
   public Circulo() {
      radio = 1;
   }
   public Circulo(double r) {
      radio = r;
   }
   void f() {
      Circulo c = new Circulo(30);
      ...
   }
}
```



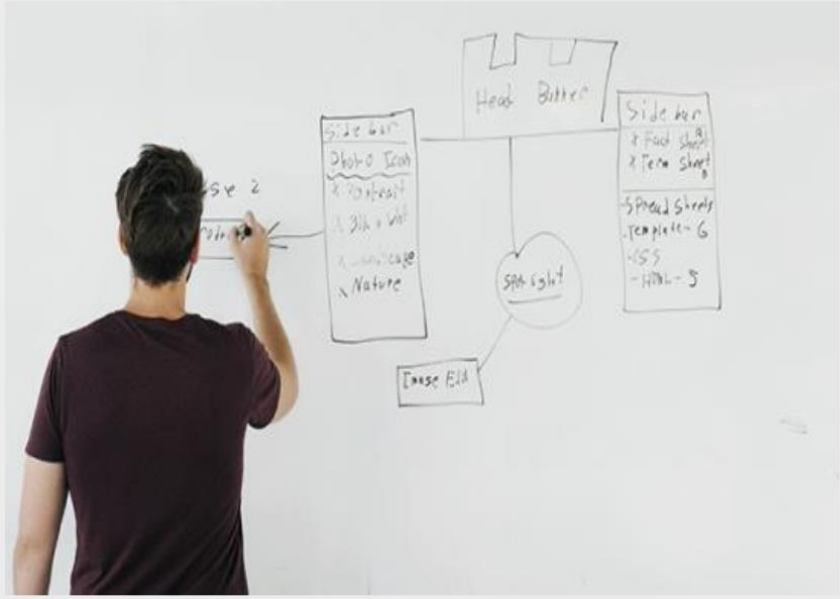
EJEMPLO:
USO DE CONSTRUCTORES

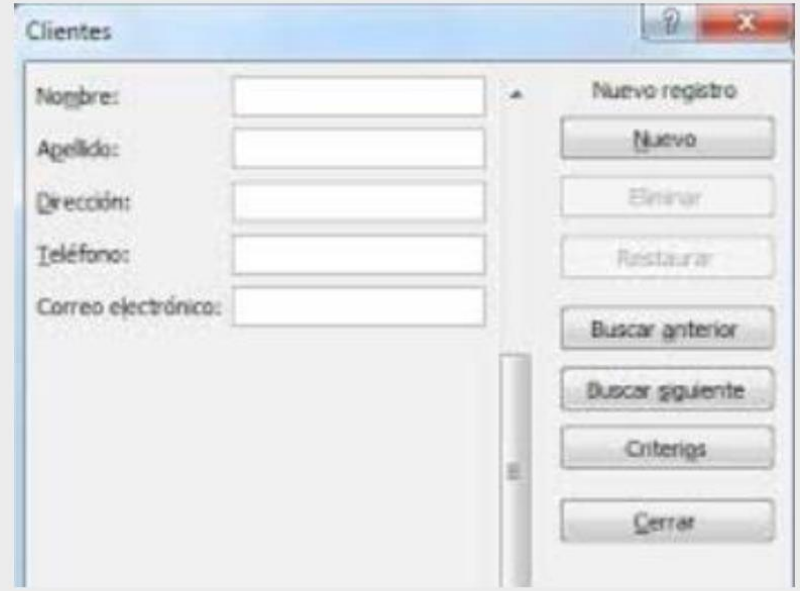
- La palabra this puede ser utilizada en la primera línea de un constructor para invocar a otro constructor

```
class Circulo {
   private double radio;
   private static int numeroCirculos = 0;

Circulo(double radio) {
   this.radio = radio;
   Circulo.numeroCirculos++;
 }

Circulo() {
   this(10);   // radio default: 10
}
}
```


Header Bottom
Sidebar
Sidebar
Nature
SPA right
Footer Edit


Clientes
Nombre:
Apellido:
Dirección:
Teléfono:
Correo electrónico:
Nuevo registro
Nuevo
Eliminar
Restaurar
Buscar anterior
Buscar siguiente
Criterios
Cerrar

REFLEXIONA

¿Cúal es la importancia de modelamiento en las aplicaciones?

¿Debemos añadir los atributos y métodos desde el inicio a una clase?

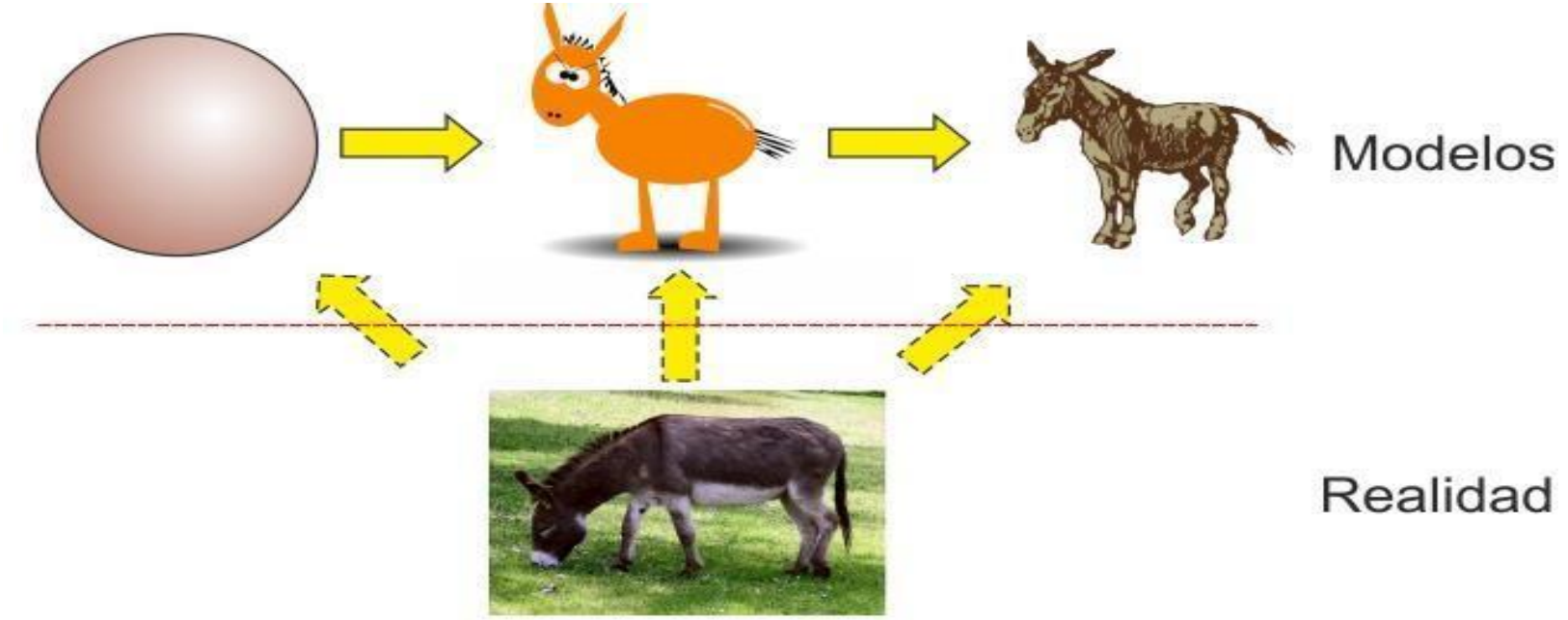
MODELADO DE PROCESOS

- Es una secuencia de pasos dispuesta con algún tipo de lógica que se enfoca en lograr algún resultado específico.

- Es un conjunto de actividades encadenadas lógicamente que toman un insumo y le agregan un valor con sentido específico para un Cliente o Grupo e interés, generando así un resultado o servicio.

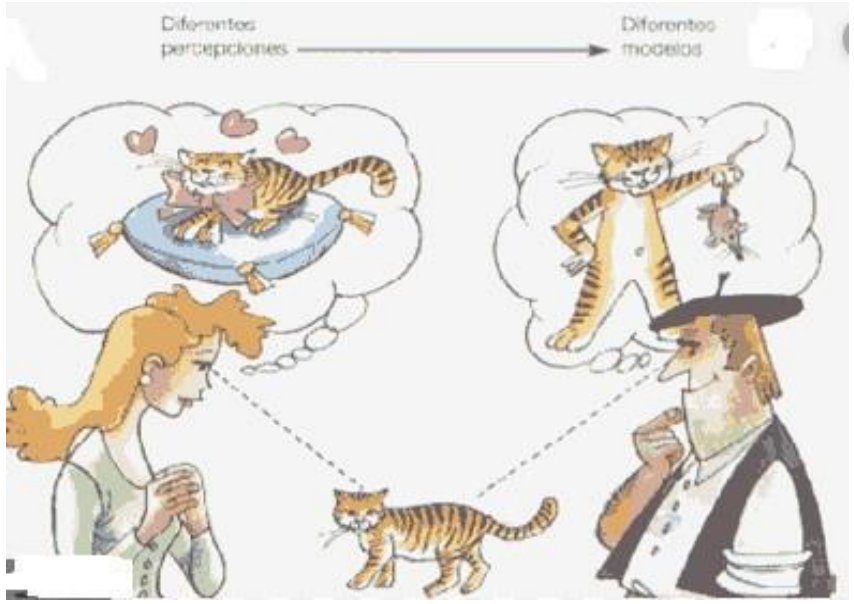
MODELO

- Un modelo es una representación de una realidad compleja.



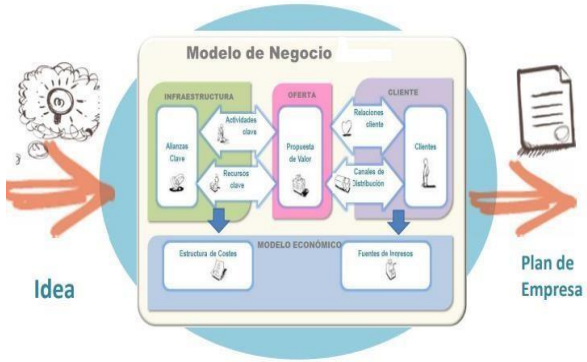
MODELO

- Es desarrollar una descripción lo más exacta posible de un sistema y de las actividades llevadas a cabo en él.



MODELO DE NEGOCIO:

- El modelo de negocios es el estudio de la organización.
- Examina el flujo de trabajo de la organización, los procesos principales dentro de la compañía y cómo ellos trabaja.
- Formar una base para mejorar el negocio actual.
- Representar la estructura del negocio mejorado.
- Formar una base para un sistema informático que apoya el negocio.



LENGUAJE UNIFICADO DE MODELADO UML

- UML es el sucesor de la ola de métodos de A y DOO que aparecieron a finales de los 80 y principios de los 90.
- UML es un lenguaje de modelado no un método.
- UML está basado en la orientación a objetos.
- UML es una notación destinada al modelado de sistemas y de procesos mediante objetos , ésta no contiene ninguna metodología sino que constituye un soporte de modelado.
- El UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. El UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos.

LENGUAJE UNIFICADO DE MODELADO
UML

- Originalmente concebido por Rational Software Corporation – Grady Booch – Booch Method.
- James Rumbaugh – Object Modeling Technique (OMT).
- Ivar Jacobson – Object Oriented Software Engineering (OOSE).
- Apoyado inicialmente por un consorcio de empresas (UML partners) que incluye a Rational, Microsoft, HP, Oracle, Unisys y otras.

LENGUAJE UNIFICADO DE MODELADO UML

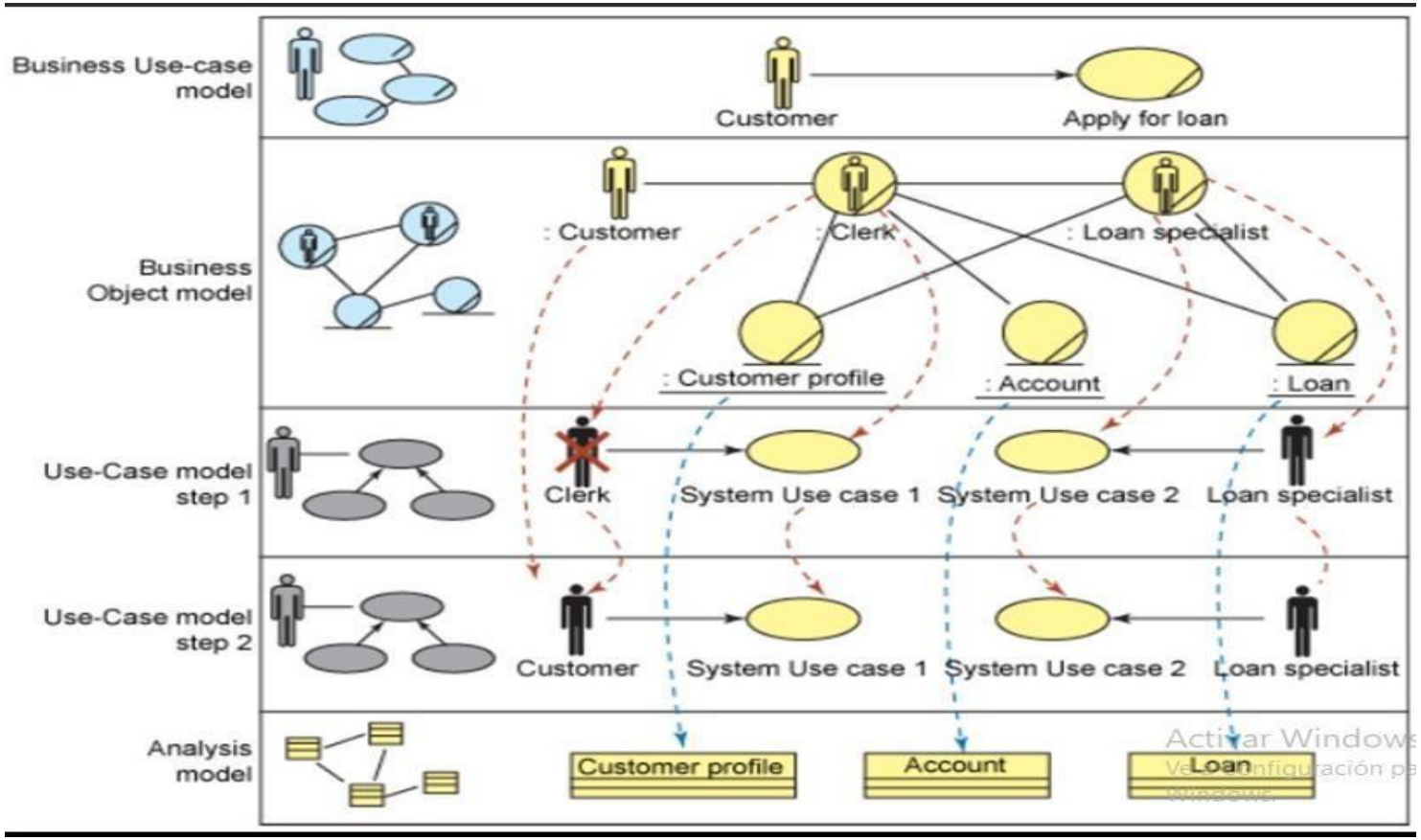


DIAGRAMA GENERAL DE UNA CLASE EN JAVA

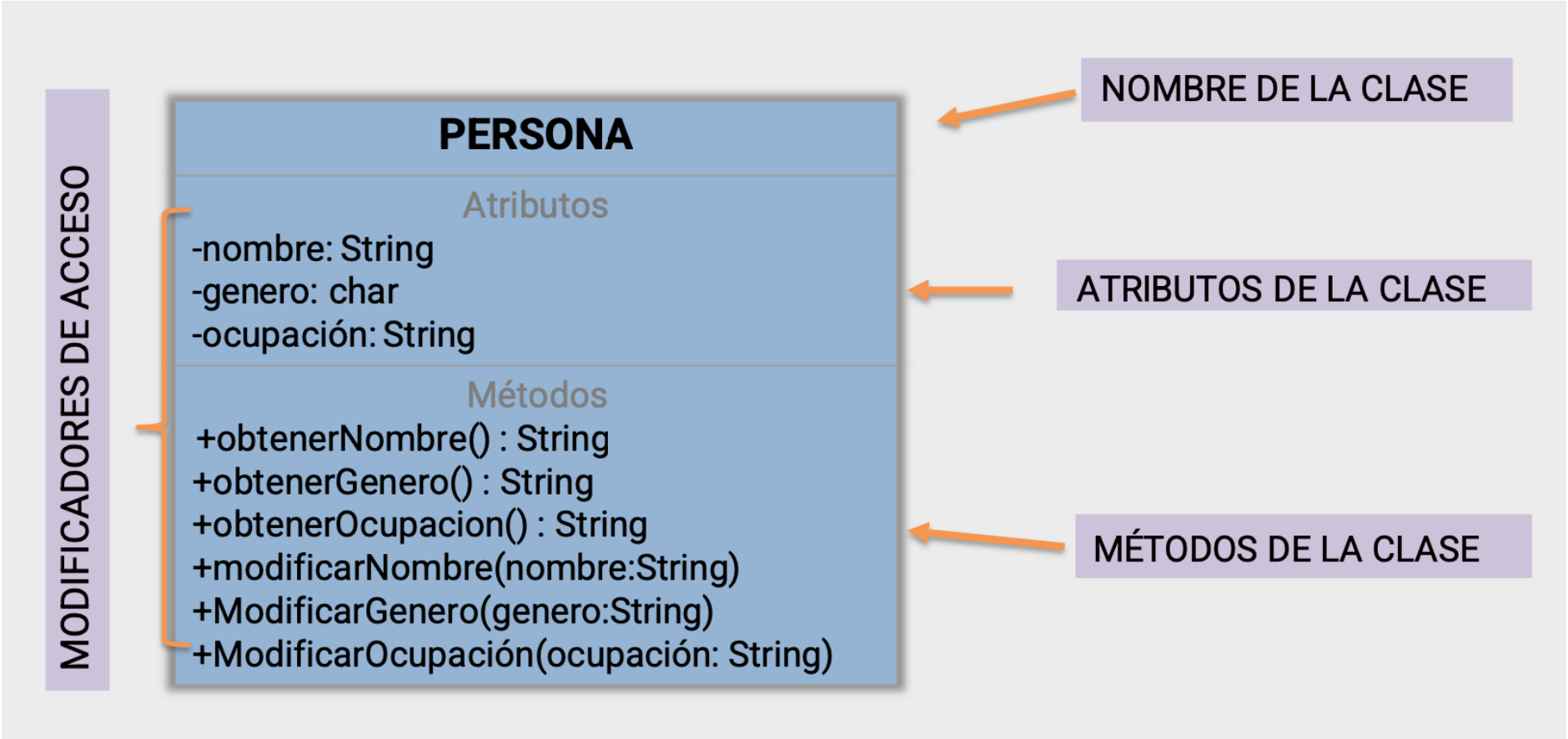


DIAGRAMA GENERAL DE UNA CLASE EN JAVA

- Tanto los atributos como Los métodos incluyen al principio de su descripción la visibilidad que tendrá. Esta visibilidad se identifica escribiendo un símbolo y podrá ser:
- **(+) Pública.** Representa que se puede acceder al atributo o función desde cualquier lugar de la aplicación.
- **(–) Privada.** Representa que se puede acceder al atributo o función únicamente desde la misma clase.
- **(#) Protegida.** Representa que el atributo o función puede ser accedida unicamente desde la misma clase o desde las clases que hereden de ella (clases derivadas).

RELACIONES:

- Una relación identifica una dependencia. Esta dependencia puede ser entre dos o más clases (más común) o una clase hacía sí misma (menos común, pero existen), este último tipo de dependencia se denomina dependencia reflexiva. Las relaciones se representan con una linea que une las clases, esta línea variará dependiendo del tipo de relación.

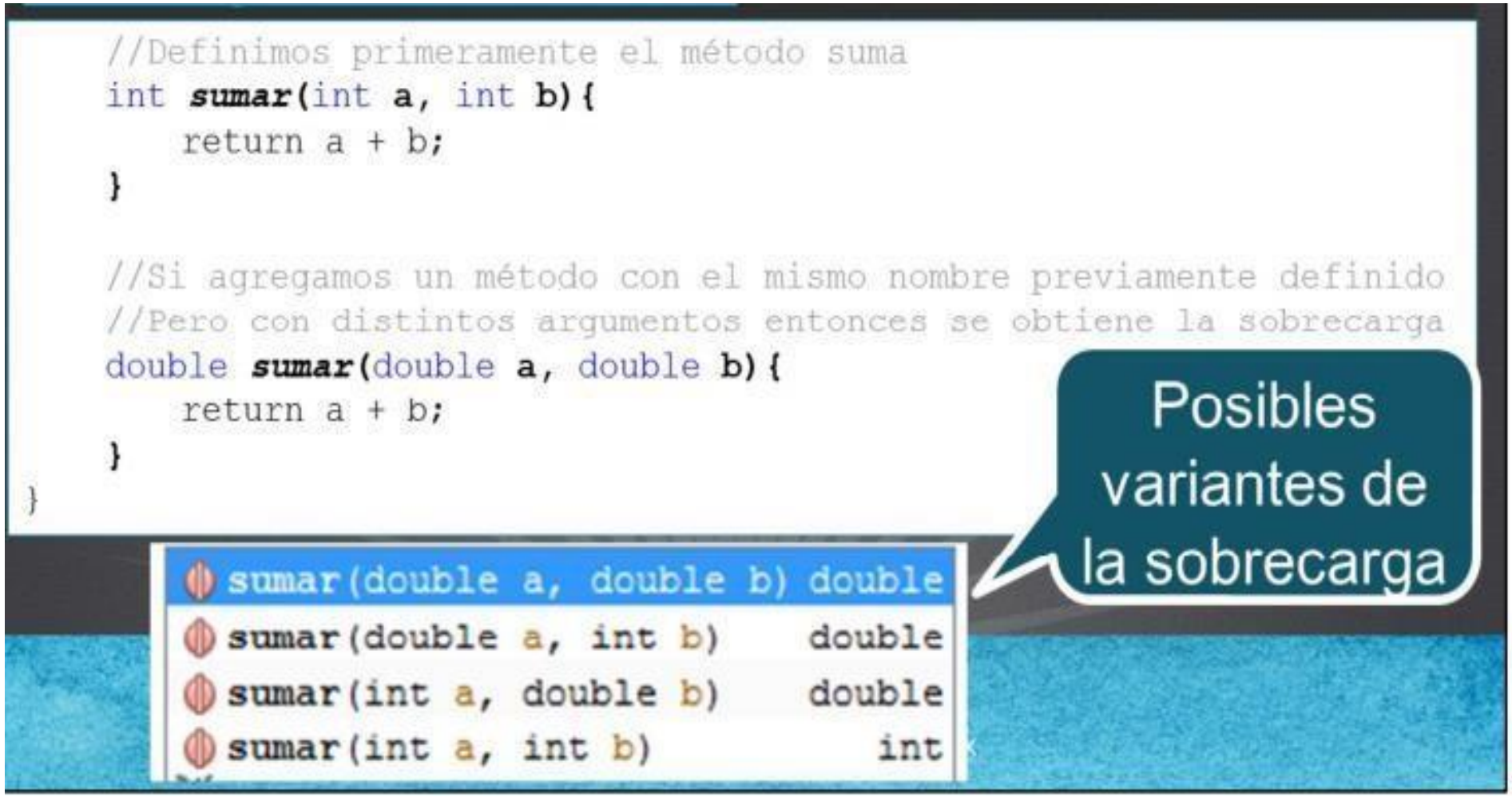
- Las relaciones en el diagrama de clases tienen varias propiedades, que dependiendo la profundidad que se quiera dar al diagrama se representarán o no. Estas propiedades son las siguientes:

RELACIONES:

- **Multiplicidad**. Es decir, el número de elementos de una clase que participan en una relación. Se puede indicar un número, un rango... Se utiliza n o * para identificar un número cualquiera.

- **Nombre de la asociación.** En ocasiones se escriba una indicación de la asociación que ayuda a entender la relación que tienen dos clases. Suelen utilizarse verbos como por ejemplo: “Una empresa contrata a n empleados”.

SOBRECARGA DE MÉTODOS:



SOBRECARGA DE MÉTODOS

- El tema de sobrecarga de métodos en java es similar al tema de sobrecarga de constructores. En tal sentido cualquier método puede ser sobrecargado.

- Ofrece mas de una opción de alguno de los métodos definidos en nuestras clases.

SOBRECARGA DE MÉTODOS

Entonces, para que una sobrecarga sea válida debe cumplir con lo siguiente:

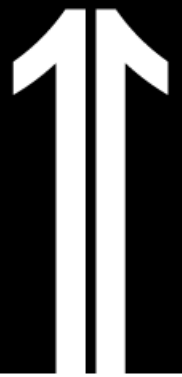
- El nombre del método debe ser igual al método que se desea sobrecargar.
- Los argumentos del método deben ser distintos al método que se desea sobrecargar, únicamente se revisa el tipo y el orden en que se agregan, no se revisa el nombre del argumento.
- El tipo de retorno no afecta si es igual o distinto al del método a sobrecargar.
- El método a sobrecargar puede estar definido en nuestra clase o en alguna clase superior.

SOBRECARGA DE OPERADORES

- Un operador sobrecargado es aquel al que se le ha dado más de una función, por ejemplo, el operador <<, este operador ha sido sobrecargado por el mismo lenguaje C++, así se le puede emplear como operador de desplazamiento de bits (a=b<<3;) o como operador de flujo de datos (cout<<"mensaje"). El Lenguaje C++ permite que los programadores sobrecarguen ciertos operadores en sus aplicaciones.

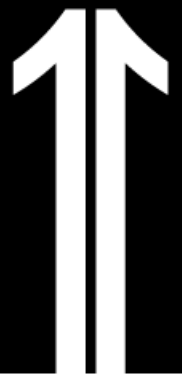
- Java no soporta la sobrecarga de operadores, pero el sistema de manera interna utiliza, un claro ejemplo es el operador “+” que se usa para suma y concatenación.

CONCLUSIONES



- Una clase es una plantilla a partir de la cual se instancian objetos.
- Los objetos contienen información (en Atributos de instancia y static) y comportamiento (en métodos de instancia y static).
- Los miembros de instancia se utilizan con la sintaxis "objeto."
- Los miembros static se utilizan con la sintaxis "clase."
- Una clase puede tener varios métodos con el mismo nombre (sobrecarga), siempre que tengan diferentes parámetros.

CONCLUSIONES



- Para instanciar una clase (crear un objeto) se utiliza el operador new.
- Los constructores son métodos especiales invocados al instanciar una clase.
- Los objetos se manejan a través de referencias.
- La palabra this representa una referencia al objeto sobre el cual se invoca un método de instancia.
- Los modificadores de acceso controlan quién tiene acceso a los miembros de una clase.

ACTIVIDAD DE CLASE

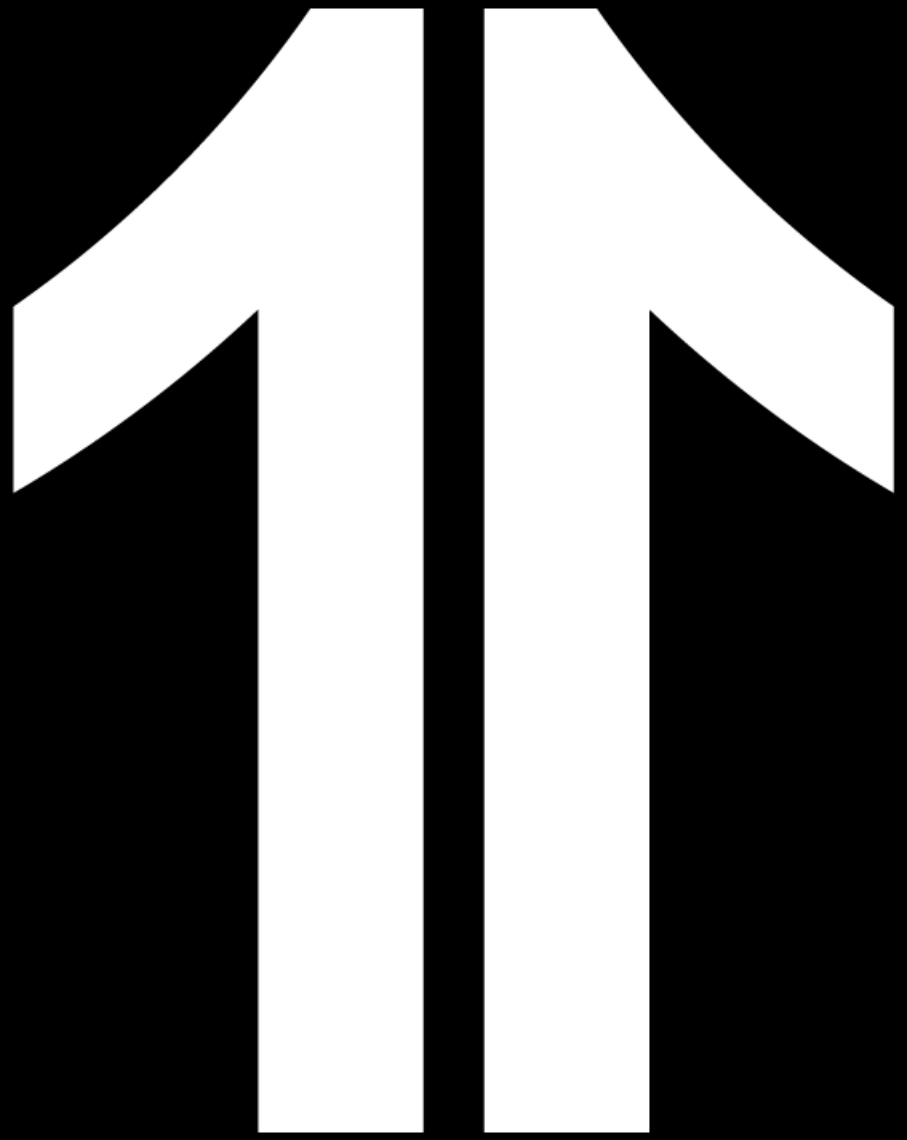
Ingresa a Recursos / Materiales para la sesión de clase: : Allí se encuentra las indicaciones y el instrumento con que será evaluado tu producto/evidencia de aprendizaje:

El docente establecerá la fecha y hora de entrega.

REFERENCIAS BIBLIOGRÁFICAS

Academia Oracle:

- https://myacademy.oracle.com/lmt/clmscoursecalendar.pr Main?site=oa&in_language_ logged_out=es



GRACIAS


UPN
UNIVERSIDAD
PRIVADA
DEL NORTE