Prof. Jannelly Bello

Implementación POO

DECLARACIÓN DE CLASES

```
<modificador> class <nombreClase>
{
        <declaracion de atributos>
        <declaracion del constructor>
        <declaracion de metodos>
}
public class Vehiculo
{
        private double cargaMax;
        public void setCargaMax(double valor)
        {

            cargaMax = valor;
        }
}
```

DECLARACIÓN DE ATRIBUTOS

<modificador> <tipo> <nombreAtributo> [= <valorInicial>];

```
public class example
{
        private int x;
        private float y = 1000.0;
        private String cadena = "bienvenido a Algoritmos 3";
}
```

DECLARACIÓN DE MÉTODOS

```
<modificador> <tipoRetorno> <nombreMetodo> ( <argumentos>)
{
    <sentencias>
}
public class dog
{
        private int peso;
        public int getPeso() {…}
        public void setPeso(int nuevoPeso) {…}
}
```

ACCESO A MIEMBROS DE UNA CLASE

Mecanismos que determinan cómo se puede acceder a un miembro de una clase, ellos son los ESPECIFICADORES(Modificadores) DE ACCESO.

private

El nivel de acceso más restringido.
Un miembro privado es accesible sólo para la clase en la que está definido.

Para declarar un miembro privado se utiliza la palabra clave **private** en su declaración.

Ejemplo:

```
class Alpha {
        private int soyPrivado;
        private void metodoPrivado()
                {
                 System.out.println("metodoPrivado");
                }
}
```

ACCESO A MIEMBROS DE UNA CLASE

Public

Permite acceder a la clase desde cualquier código del programa.

Para declarar un miembro público se utiliza la palabra clave **public** en su declaración.

Ejemplo:

```
class Alpha {
        public int soyPublico;
        public void metodoPublico()
                {
                 System.out.println("metodoPublico");
                }
}
```

NOTA: Cuando no se utiliza un especificador de acceso, por defecto se considera el miembro de la clase **public** dentro de su propio paquete, pero no se puede acceder al mismo desde fuera de su paquete

CONSTRUCTORES

- ✓Inicializa un objeto inmediatamente después de su creación.
- ✓El constructor **debe** tener el mismo nombre de la clase.
- ✓Sintácticamente es similar a un método.
- ✓No devuelven ningún tipo: el tipo implícito que devuelve un constructor de clase es el propio tipo de la clase.
- ✓Cada clase que se declare puede tener un constructor, el cual puede utilizarse para inicializar un objeto de una clase al momento de crear ese objeto.
- ✓De manera predeterminada, el compilador proporciona un constructor predeterminado sin parámetros, en cualquier clase que no incluya un constructor de manera explícita.
- ✓Java requiere una llamada al constructor para cada objeto que se crea.
- ✓Cuando una clase solo tiene el constructor predeterminado, sus variables de instancia se inicializan con sus valores predeterminados. (int, long, byte, short =0; boolean=false).

CONSTRUCTORES

✓**Diferencia entre constructores y métodos:** Los constructores no devuelven valores por lo cual no especifican ningún valor de retorno (ni siquiera void).

✓Generalmente se declaran public.

✓Si se declaran uno o mas constructores para una clase, el compilador de java no creara un constructor predeterminado para esa clase.

DECLARACION

```
[<modificador> ] <NombreClase> ( <argumentos> )
{
    <sentencias>
}
```

CONSTRUCTORES

CONSTRUCTORES CON PARÁMETROS

El objeto es inicializado como se especifica en los parámetros del constructor.

EJEMPLO

```
public class Perro {
        private int peso;
//
        public Perro()
        {
            peso = 42;
        }
}
```

```
//declara, reserva memoria e inicializa los objetos de Perro
Perro miPerro=new Perro(); //Sin Parámetros
Perro miPerro=new Perro(10); // Con Parámetros
```

OPERADOR NEW

Reserva memoria dinámicamente para un objeto.

variable = **new** nombre_de_clase**;**

Donde:

Variable: Es una variable del mismo tipo de la clase creada

Nombre_de_clase: el nombre de la clase que está siendo instanciada.

El constructor de la clase se especifica colocando paréntesis después del nombre_de_clase

CONSTRUCCION E INICIALIZACION DE OBJETOS

✓Cuando se crea una clase se crea un nuevo tipo de datos que se utilizará para crear objetos de ese tipo.

Para obtener objetos de una clase se tiene que:

- Declarar una variable del tipo de la clase. Esta variable es una referencia a un objeto.
- Obtener copia física del objeto y asignarla a esa variable. Esto se hace a través del operador **new** (asigna dinámicamente memoria a un objeto y devuelve una referencia al mismo).
- Luego se almacena esa referencia en la variable.

CONSTRUCCION E INICIALIZACION DE OBJETOS

Ejemplo:

```
public class Caja{
/*Atributos y Metodos*/
}

Caja caja1; // declara la referencia a un objeto.
caja1 = new Caja();          //reserva espacio para el objeto.

Caja caja1 = new Caja();  //reserva espacio para el objeto.
```

REFERENCIA THIS

Es una referencia al objeto sobre el cual ha sido llamado el método.

Ejemplo 1:

```
Caja(double w, double h, double d){
        this.ancho = w;
        this.alto = h;
        this.largo = d;
}
```

Ejemplo 2:

```
Caja(double ancho, double alto, double largo){
        this.ancho = ancho;
        this.alto = alto;
        this.largo = largo;
}
```

DISEÑO DE CLASES

- **Superclase**: Clase Heredada.

- **Subclase:** Clase que hereda.

- **Sintaxis Herencia:**

```
class Nombre_subclase extends Nombre_superclase {
//Atributos
//Constructores
//Métodos
}
```

DISEÑO DE CLASES

- **Control de Acceso**

Una subclase no puede acceder a miembros de a superclase que hayan sido declarados como privados.

- **Palabra clave SUPER**

Se utiliza siempre que una subclase necesite referirse a su superclase inmediata.

USO

✓Llamar al constructor de la superclase.

SUPER (Lista de parametros)

✓Cuando nombres de miembros de una subclase OCULTAN miembros del mismo nombre en la superclase.

SUPER.miembro donde miembro=Metodo, variable de instancia

DISEÑO DE CLASES

- **Jerarquía multinivel**

  Se puede construir jerarquías que contengan tanto niveles de herencia como se quiera.

- **Orden de ejecución de los constructores**

✓ En una jerarquia de clases los constructores se ejecutan en el orden en el que se derivan, desde la superclase a la subclase.

✓Super() debe ser la sentencia que se ejecute en el constructor de una subclase. Este orden es el mismo tanto si se usa super como si no se usa.

✓Si no se utiliza super(), se ejecuta el constructor por defecto o constructor sin parametros de cada superclase.

DISEÑO DE CLASES

- **Sobreescritura de métodos**

•Se da cuando en una jerarquía de clases, un método de una subclase tienen el mismo nombre y tipo que un método de su superclase.

•Cuando se llama un método sobreescrito de una subclase, esta llamada se refiere a la versión de ese método definida por la subclase. La versión del método definida por la superclase queda oculta.

DISEÑO DE CLASES

- **Sobrecarga de métodos**

Es cuando se definen dos o mas métodos que comparten el mismo nombre dentro de la misma clase, aunque la declaración de sus parámetros sea diferente.

- **Sobrecarga de constructores**

Cuando se definen dos o mas constructores para una clase.

- **Selección de métodos dinámica**

La sobreescritura de métodos es la base para la selección de métodos dinámica. Mediante este mecanismo la llamada a una función sobreescrita se resuelve en tiempo de ejecución y no en tiempo de compilación.

La transferencia dinámica de métodos permite implementar el polimorfismo durante el tiempo de ejecución.

DISEÑO DE CLASES

- **Polimorfismo dinámico, resuelto en tiempo de ejecución: Ejemplo**

```
Figura f=new Rectangulo(10,10); //Instancia de Figura
Rectangulo r = new Rectangulo(9,5); //Instancia de Rectangulo
Triangulo t = new Triangulo(10,8);  //Instancia de Triangulo

System.out.println(f.area()); //Imprime area de Figura

f=r; //Asignacion de referencia  de Rectangulo a Figura
System.out.println("El área de rectangulo es: " + f.area());//Imprime area de rectangulo

f=t;//Asignacion de referencia de triangulo a figura
System.out.println("El área de triangulo es: " + f.area()); //imprime area de triangulo
```

ASOCIACIÓN ENTRE CLASES