INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

Adrián A. Alarcón O.

¿QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

- Paradigma de programación
- Organiza el código agrupándolo en objetos

¿QUÉ ES UN OBJETO?

Elemento independiente que contienen información y funcionalidad.

Estado y Comportamiento

¿QUÉ ES UNA CLASE?

Es un modelo o plantilla que permite la creación de Objetos y define un conjunto de variables y métodos

CLASE DOG

¿Atributos?

¿Comportamiento?

CLASE DOG

```
public class Dog {

 int age;
 String breed;
 String color;

 void bark(){ }

 void eat(){ }

 void sleep(){ }

 void run(){ }

}
```

CLASE EMPLOYEE

```
class Employee {

 String name;
 String designation;
 int age;
 double salary;

 work() {  }

 talk() { }

}
```

EJERCICIO

```
public class HelloWorld {
        public static void main(String[] args) {
                System.out.println("Hola Mundo");
        }
}
```

CLASES

- Al menos deberá existir una clase por archivo
- Un archivo puede contener 'n' clases
- Al menos debe existir una clase con el mismo nombre del archivo
- Nomenclatura CamelCase
- Las variables y métodos inician con minúsculas

EJEMPLO

```
//Archivo ThisIsMyClass.java

class ThisIsMyClass{
 int thisIsVariableWithVeryLongName;
 boolean boolean_variable;
 String text = "este es el valor de mi variable";
}

class ThisIsAnotherClass{
 // este es un comentario
 // esta clase no tiene variables
}

class ThisIsAnotherAnotherClass{
 // esta clase tiene un método
 void print(){  }
}
```

EJERCICIO

```
// Elaborar las siguientes clases

- Vehículo
- Metro
- Tren
- Camión
- Carro
- Bicicleta
- Submarino
- Bote
- Balsa

// El archivo se deberá llamar Vehicle.java
// Todas las clases deberán contener variables y métodos
```

COMENTARIOS

```
// Comentarios en una linea

/*
* Este es un comentario en
* multiples lineas
*/

/*
 Este es un otro comentario en
 multiples lineas
*/
```

TIPOS DE DATOS

```
// Tipos Primitivos

Numéricos enteros
- byte
- short
- int
- long

Numéricos Reales
- float
- double

Booleanos
- boolean

Caracteres
- char
```

DECLARACIÓN DE VARIABLES

```
int a, b, c;

int e, f;

byte B;

double pi;

char a;
```

ASIGNACIÓN DE VARIABLES

```
int a = 1, b = 2, c=3;

short e=3 , f =3;

byte B = 1;

double pi;

char a = 'c';
```

INSTANCIAS DE CLASE

```
class Printer {

}
```

INSTANCIAS DE CLASE

```
class Printer {

 boolean status;

  void print(){
     System.out.println("printing...");
  }

}
```

INSTANCIAS DE CLASE

```
class Printer {

 boolean status;

  void print(){
     System.out.println("printing...");
  }

}

class MainClass {

 public static void main(String[] args) {

  }

}
```

INSTANCIAS DE CLASE

```
class Printer {

 boolean status;

  void print(){
     System.out.println("printing...");
  }

}

class MainClass {

 public static void main(String[] args) {
   Printer printer;

 }

}
```

INSTANCIAS DE CLASE

```
class Printer {

 boolean status;

  void print(){
     System.out.println("printing...");
  }

}

class MainClass {

 public static void main(String[] args) {
   Printer printer = new Printer();

 }

}
```

INSTANCIAS DE CLASE

```
class Printer {

 boolean status;

  void print(){
     System.out.println("printing...");
  }

}

class MainClass {

 public static void main(String[] args) {
   Printer printer = new Printer();
   printer.print();
   printer.status = true;
 }

}
```

INSTANCIAS DE CLASE

```
class Printer {

 boolean status;

 void print(){
    System.out.println("printing...");
 }

 public static void main(String[] args) {
   Printer printer = new Printer();
   Printer thisIsAnotherPrinter = new Printer();

   Printer thisPrinterIsNotInitialized;
 }

}
```

EJERCICIO

```
// Crear una instancia de las siguientes clases
// en el método main de la clase Vehiculo

- Metro
- Tren
- Camión
- Carro
- Bicicleta
- Submarino
- Bote
- Balsa
```

EJERCICIO

```
class Vehicle{

  public static void main(String[] args) {
    Car myCar = new Car();
    Truck truck = new Truck();
    Boat b = new Boat();
  }

}

class Car{

}
class Truck{

}
class Boat{

}
```

CONSTRUCTOR

Se utiliza para construir un objeto de una clase

- Pueden existir mas de 1 constructor por clase
- Es un método perteneciente a la clase
- No regresa nada (return)
- Se llama igual que la clase

EJEMPLO

```
class Person{

  String name;
  int age;

}
```

EJEMPLO

```
class Person{

  String name;
  int age;

  Person(){
    name = "anonymus";
    age = 0;
  }
}
```

EJEMPLO

```
class Person{

  String name;
  int age;

  Person(){
    name = "anonymus";
    age = 0;
  }
}

class MainClass{

  public static void main(String[] args) {
    Person p = new Person();
  }

}
```

EJEMPLO

```
class Person{

  String name;
  int age;

  Person(String nameParam, int ageParam){
    name =  nameParam;
    age = ageParam;
  }
}
```

EJEMPLO

```
class Person{

  String name;
  int age;

  Person(String nameParam, int ageParam){
    name =  nameParam;
    age = ageParam;
  }
}

class MainClass{

  public static void main(String[] args) {
    Person p = new Person("Pancho Pantera", 15);
  }

}
```

EJERCICIO

```
// Agregar un constructor a cada clase de vehículo
// El constructor deberá inicializar al menos una variable

- Metro
- Tren
- Camión
- Carro
- Bicicleta
- Submarino
- Bote
- Balsa
```

EJERCICIO

```
class Vehicle{

  public static void main(String[] args) {
    Car myCar = new Car(2018);
  }

}

class Car{

  int model;

  Car(int modelParam){
    model = modelParam;
  }

}
```

PAQUETE

Se utiliza para categorizar las clases e interfaces

PAQUETE

```
package mx.uach.fing;

class MyClass{

}

// mx.uach.fing.MyClass
```

IMPORTAR UN PAQUETE

```
// Si una clase esta asignada a un paquete debemos
// especificar al compilador donde localizar la clase

se utiliza la palabra reservada

- import
```

IMPORTAR UN PAQUETE

```
import mx.uach.fing.MyClass;

import mx.uach.fing.*;

class MainApp{

        public static void main(String[] args) {

                MyClass oneClass = new MyClass();

        }

}
```

MODIFICADORES

MODIFICADORES DE CONTROL DE ACCESO

Cambia la visibilidad de un elemento

MODIFICADORES DE CONTROL DE ACCESO

- public
- protected
- default
- private

MODIFICADORES DE CONTROL DE ACCESO

- public: visible a todo el mundo
- protected: visible a todo el paquete y a las subclases
- default: visible a todo el paquete
- private: visible solo a la clase

MODIFICADOR	CLASE	PACKAGE	SUBCLASE	TODOS
Public	Sí	Sí	Sí	Sí
Protected	Sí	Sí	Sí	No
Private	Sí	No	No	No
Por defecto	Sí	Sí	No	No

MODIFICADORES

```
package com.uach.app.printer

class Printer {

  boolean status;

  void print(){

  }

}
```

MODIFICADORES

```
package com.uach.app.printer

public class Printer {

  public boolean status;

  public void print(){

  }

}
```

MODIFICADORES

```
package com.uach.app.printer

public class Printer {

  private boolean status;

  protected void print(){

  }

}
```

MODIFICADORES

```
package com.uach.app.printer

public class Printer {

  private boolean status;

  protected void print(){

    System.out.println("este es el estatus: " + status);

  }

}
```

MODIFICADORES

```
package com.uach.printer

public class Printer {
  private boolean status;
  protected void print(){
      System.out.println("este es el estatus: " + status);
  }
}

public class MyMainClass{

  public static void main(String[] args) {
    Printer printer = new Printer();
    printer.status = false;
    printer.print();
    System.out.println("este es el estatus: " + printer.status);
  }

}
```

MODIFICADORES

```
package com.uach.printer

public class Printer {
  protected boolean status;
  protected void print(){
      System.out.println("este es el estatus: " + status);
  }
}

public class MyMainClass{

  public static void main(String[] args) {
    Printer printer = new Printer();
    printer.status = false;
    printer.print();
    System.out.println("este es el estatus: " + printer.status);
  }

}
```

MODIFICADOR STATIC

Hace que el miembro pertenezca a la clase

- Variables
- Métodos

MODIFICADOR STATIC

```
public class MyClass {
  public int number;
}

public class MyMainClass{

  public static void main(String[] args) {
    MyClass a = new MyClass();
    MyClass b = new MyClass();
    MyClass c = new MyClass();

    a.number = 1;
    b.number = 2;
    c.number = 3;
  }

}
```

MODIFICADOR STATIC

```
public class MyClass {
  public static int number;
}

public class MyMainClass{

  public static void main(String[] args) {
    MyClass a = new MyClass();
    MyClass b = new MyClass();
    MyClass c = new MyClass();

    a.number = 1;
    b.number = 2;
    c.number = 3;
  }

}
```

MODIFICADOR STATIC

```
public class MyClass {
  public static int number;
}

public class MyMainClass{

  public static void main(String[] args) {
    MyClass.number = 1;
    MyClass.number = 2;
    MyClass.number = 3;
  }

}
```

MODIFICADOR STATIC

```
public class MyClass {
  public static void read(){

  }
}

public class MyMainClass{

  public static void main(String[] args) {
    MyClass.read();
  }

}
```

MODIFICADOR FINAL

- Clases
- Variables
- Métodos

MODIFICADOR FINAL

- Clases: No puede ser heredada
- Variables: Solo puede ser inicializada una vez
- Métodos: No puede ser sobreescrito

MODIFICADOR FINAL

```
public final class MyClass {

   public final int x = 1;

   public final void read(){

   }

}
```

VARIABLES DE INSTANCIA

Es una variable que se relaciona con una única instancia de una clase. Cada vez que se crea un objeto, el sistema crea una copia de todas las variables que están vinculadas con dicha clase, haciéndolas propias de esa instancia

VARIABLES DE INSTANCIA

```
public class MyClass {

   public int instanceVariable = 1;

   public void read(){

   }

}
```

VARIABLES ESTATICAS

Es una variable con el modificador `static` y que pertenece a la clase

- Puede ser accedida desde la clase sin hacer uso de la instancia

VARIABLES ESTATICAS

```
public class MyClass {

   public static int staticVariable = 1;

   public void read(){

   }

}
```

VARIABLES LOCALES

Es una variable que sólo pueden ser accedidas desde el bloque de código en el que han sido declaradas

VARIABLES LOCALES

```
public class MyClass {

   public void read(){
     public int localVariable = 1;
   }

}
```

SHADOWING

SHADOWING

THIS CONTENT
IS NOT AVAILABLE

SHADOWING

Se refiere a dos variables con el mismo nombre pero en diferentes ambitos

- Se emplea la palabra reservada para diferenciarlas

SHADOWING

```
public class MyClass {

  int x = 1;

  public void print(){
    int x = 10;
    System.out.println(x);
  }

  public static void main(){
    System.out.println(x);
    print();
  }

}
```

SHADOWING

```
public class MyClass {

  int x = 1;

  public void print(){
    int x = 10;
                x = x;
  }

  public static void main(){
    System.out.println(x);
  }

}
```

SHADOWING

```
public class MyClass {

  int x = 1;

  public void print(){
    int x = 10;
                x = this.x;
                System.out.println(x);
  }

  public static void main(){
    print();
  }

}
```

ARRAYS

Un array es una estructura de datos que nos permite almacenar una gran cantidad de datos de un mismo tipo.

- No se puede cambiar el tamaño de un arreglo una ves asignado el tamaño
- Cuando se declara un arreglo se inicializan con los valores por defecto

ARRAYS

```
int[] x;
int []x;
int x [];
int[][] x;
int x[][];
```

ARRAYS

```
int[] testScores = new int[3]; // solo define el tipo y cantidad

int[] testScores = new int[] {4,7,2}; // define el tipo y asigna

int[] testScores = {4,7,2}; // define el tipo y asigna los valores
```

ARRAYS

```
public class MyClass {

   public static void main(){

    int[] numbers = new int[4];
    numbers[0] = 10;
    numbers[1] = 0;
    numbers[2] = 150;
    numbers[3] = 400;

    int total = numbers.length;
  }

}
```

BLOQUES DE INICIALIZACIÓN

Es un bloque de código es ejecutado una ves que se crea una instancia y es ejecutado justo despues de llamar el super()

- Se ejecutan en el orden en el que estan definidos
- Se ejecutan cada ves que se crea una instancia de la clase

BLOQUES DE INICIALIZACIÓN

```
public class MyClass{

  public MyClass(){
    super();
    System.out.println("Mi Constructor");
  }

  {
    System.out.println("bloque de inicializacion 1");
  }

  {
    System.out.println("bloque de inicializacion 2");
  }

  public static void main(String[] args) {
    System.out.println("main");
    new MyClass();
  }
}
```

BLOQUES DE INICIALIZACIÓN ESTATICOS

Es un bloque que se ejecuta solo la primera ves que se inicializa la clase

- Se ejecutan en el orden en el que estan definidos
- Se ejecutan solo una ves que se crea una instancia de la clase

BLOQUES DE INICIALIZACIÓN

```
public class MyClass{

  public MyClass(){
    super();
    System.out.println("Mi Constructor");
  }

  static{
    System.out.println("bloque de inicializacion 1");
  }

  static{
    System.out.println("bloque de inicializacion 2");
  }

  public static void main(String[] args) {
    System.out.println("main");
    new MyClass();
  }
}
```

JAVA BEAN

Es una clase simple que solo contiene atributos privados y proporciona métodos para acceder a los datos

JAVA BEAN

```
public class Person{

  private String name;

  public Person(){

  }

  public String getName(){
    return name;
  }

  public void setName(String name){
    this.name = name;
  }

}
```