## Universidad Autónoma de Ciudad Juárez

## División Multidisciplinaria Ciudad Universitaria

## 

## Cuadro comprativo de lenguajes de Programacion Orientada a Objetos

## Programacion II

## Docente: Alan Ponce

## 

## 

## Alumno: Diego Almanza Beyless

## Licenciatura en Ingeniería de Software

## 08 de Marzo de 2019

**Procedural programing**

Procedural Programming which at times has been referred to as inline programming takes a more top down approach to programming. Object-oriented Programming uses classes and objects, Procedural Programming takes on applications by solving problems from the top of the code down to the bottom.

This happens when a program starts with a problem and then breaks that problem down into smaller subproblems or sub-procedures. These sub-procedures are continually broken down in the process called functional decomposition until the sub-procedure is simple enough to be solved.

The issue that is obvious in Procedural Programming is that if an edit is needed to the program, the developer must edit every line of code that corresponds to the original change in the code. An example would be if at the beginning of a program a variable was set to equal the value of 1. If other sub-procedures of the program rely on that variable equaling 1 to function properly they will also need to be edited. As more and more changes may be needed to the code, it becomes increasingly difficult to locate and edit all related elements in the program.

**Fortran**

Fortran is a general-purpose, [compiled](https://en.wikipedia.org/wiki/Compiled_language) [imperative](https://en.wikipedia.org/wiki/Imperative_programming) [programming language](https://en.wikipedia.org/wiki/Programming_language) that is especially suited to [numeric computation](https://en.wikipedia.org/wiki/Numerical_analysis) and [scientific computing](https://en.wikipedia.org/wiki/Computational_science).

Originally developed by [IBM](https://en.wikipedia.org/wiki/IBM) in the 1950s for scientific and engineering applications, FORTRAN came to dominate this area of programming early on and has been in continuous use for over half a century in computationally intensive areas such as [numerical weather prediction](https://en.wikipedia.org/wiki/Numerical_weather_prediction), [finite element analysis](https://en.wikipedia.org/wiki/Finite_element_method), [computational fluid dynamics](https://en.wikipedia.org/wiki/Computational_fluid_dynamics), [computational physics](https://en.wikipedia.org/wiki/Computational_physics), [crystallography](https://en.wikipedia.org/wiki/Crystallography) and [computational chemistry](https://en.wikipedia.org/wiki/Computational_chemistry). It is a popular language for [high-performance computing](https://en.wikipedia.org/wiki/High-performance_computing) and is used for programs that benchmark and rank the world's [fastest supercomputers](https://en.wikipedia.org/wiki/TOP500).

Ejemplo:

program helloworld  
 print \*, "Hello world!"  
end program helloworld

**ALGOL**

A contemporary of the other early 1950’s programming languages FORTRAN, Lisp, and COBOL. It became the standard way of describing algorithms in academia for 30 years, meaning it influenced many other languages’ syntaxes, including C.

Example:

BEGIN DISPLAY("HELLO WORLD!") END.

**COBOL**

COBOL is a compiled English-like computer programming language designed for business use. It is imperative, procedural and, since 2002, object-oriented. COBOL is primarily used in business, finance, and administrative systems for companies and governments. COBOL is still widely used in legacy applications deployed on mainframe computers, such as large-scale batch and transaction processing jobs. But due to its declining popularity and the retirement of experienced COBOL programmers, programs are being migrated to new platforms, rewritten in modern languages or replaced with software packages. Most programming in COBOL is now purely to maintain existing applications.

Example:

IDENTIFICATION DIVISION.  
 PROGRAM-ID. hello-world.  
 PROCEDURE DIVISION.  
 DISPLAY "Hello, world!"

**BASIC**

BASIC (Beginner's All-purpose Symbolic Instruction Code) is a family of [general-purpose](https://en.wikipedia.org/wiki/General-purpose_programming_language), [high-level programming languages](https://en.wikipedia.org/wiki/High-level_programming_language) whose design philosophy emphasizes ease of use. In 1964, [John G. Kemeny](https://en.wikipedia.org/wiki/John_G._Kemeny) and [Thomas E. Kurtz](https://en.wikipedia.org/wiki/Thomas_E._Kurtz) designed the original BASIC language at [Dartmouth College](https://en.wikipedia.org/wiki/Dartmouth_College). They wanted to enable students in fields other than science and mathematics to use computers. At the time, nearly all use of computers required writing custom software, which was something only [scientists](https://en.wikipedia.org/wiki/Scientist)and [mathematicians](https://en.wikipedia.org/wiki/Mathematician) tended to learn.

Example:

10 PRINT "Hello, World!"  
20 END

**Pascal**

Pascal is a procedural programming language that supports structured programming and data structures to encourage good programming practices. Pascal was originally developed in 1970 by Niklaus Wirth and is named after the famous French mathematician Blaise Pascal.

Example:

program HelloWorld(output);  
begin  
 Write('Hello, world!')  
end.

**C**

C is a [general-purpose](https://en.wikipedia.org/wiki/General-purpose_language), [imperative](https://en.wikipedia.org/wiki/Imperative_programming) computer [programming language](https://en.wikipedia.org/wiki/Programming_language), supporting [structured programming](https://en.wikipedia.org/wiki/Structured_programming), [lexical variable scope](https://en.wikipedia.org/wiki/Lexical_variable_scope) and [recursion](https://en.wikipedia.org/wiki/Recursion_(computer_science)), while a [static type system](https://en.wikipedia.org/wiki/Static_type_system) prevents many unintended operations. By design, C provides constructs that map efficiently to typical [machine instructions](https://en.wikipedia.org/wiki/Machine_code), and therefore it has found lasting use in applications that had formerly been coded in [assembly language](https://en.wikipedia.org/wiki/Assembly_language), including [operating systems](https://en.wikipedia.org/wiki/Operating_system), as well as various [application software](https://en.wikipedia.org/wiki/Application_software) for computers ranging from [supercomputers](https://en.wikipedia.org/wiki/Supercomputer) to [embedded systems](https://en.wikipedia.org/wiki/Embedded_system).

Example:

#include <stdio.h>  
  
int main(void)  
{  
 printf("hello, world\n");  
}

**Object Oriented Programing**

**C++**

C++ is a [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). It has [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) and [generic](https://en.wikipedia.org/wiki/Generic_programming) programming features, while also providing facilities for [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) [memory](https://en.wikipedia.org/wiki/Memory_(computing)) manipulation.

It was designed with a bias toward [system programming](https://en.wikipedia.org/wiki/System_programming) and [embedded](https://en.wikipedia.org/wiki/Embedded_software), resource-constrained and large systems, with [performance](https://en.wikipedia.org/wiki/Performance_(software)), efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including [desktop applications](https://en.wikipedia.org/wiki/Application_software), servers (e.g. [e-commerce](https://en.wikipedia.org/wiki/E-commerce), [Web search](https://en.wikipedia.org/wiki/Web_search_engine) or [SQL](https://en.wikipedia.org/wiki/SQL) servers), and performance-critical applications (e.g. [telephone switches](https://en.wikipedia.org/wiki/Telephone_switches) or [space probes](https://en.wikipedia.org/wiki/Space_probes)). C++ is a [compiled language](https://en.wikipedia.org/wiki/Compiled_language), with implementations of it available on many platforms. Many vendors provide C++ compilers, including the [Free Software Foundation](https://en.wikipedia.org/wiki/Free_Software_Foundation), [Microsoft](https://en.wikipedia.org/wiki/Microsoft), [Intel](https://en.wikipedia.org/wiki/Intel), and [IBM](https://en.wikipedia.org/wiki/IBM).

In 1985, the first edition of [*The C++ Programming Language*](https://en.wikipedia.org/wiki/The_C%2B%2B_Programming_Language) was released, which became the definitive reference for the language, as there was not yet an official standard.The first commercial implementation of C++ was released in October of the same year.

In 1989, C++ 2.0 was released, followed by the updated second edition of *The C++ Programming Language* in 1991. New features in 2.0 included multiple inheritance, abstract classes, static member functions, [const member functions](https://en.wikipedia.org/wiki/Const_correctness), and protected members. In 1990, *The Annotated C++ Reference Manual* was published. This work became the basis for the future standard. Later feature additions included [templates](https://en.wikipedia.org/wiki/Template_(programming)), [exceptions](https://en.wikipedia.org/wiki/Exception_handling), [namespaces](https://en.wikipedia.org/wiki/Namespaces), new [casts](https://en.wikipedia.org/wiki/Cast_(computer_science)), and a [boolean type](https://en.wikipedia.org/wiki/Boolean_datatype).

After the 2.0 update, C++ evolved relatively slowly until, in 2011, the [C++11](https://en.wikipedia.org/wiki/C%2B%2B11) standard was released, adding numerous new features, enlarging the standard library further, and providing more facilities to C++ programmers. After a minor [C++14](https://en.wikipedia.org/wiki/C%2B%2B14) update released in December 2014, various new additions were introduced in [C++17](https://en.wikipedia.org/wiki/C%2B%2B17), and further changes planned for 2020.

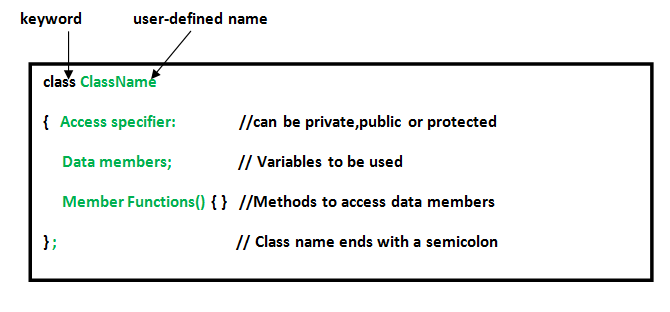
As of 2017, C++ remains the third most popular programming language, behind [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) and C.

On January 3, 2018, Stroustrup was announced as the 2018 winner of the [Charles Stark Draper Prize](https://en.wikipedia.org/wiki/Charles_Stark_Draper_Prize) for Engineering, "for conceptualizing and developing the C++ programming language".

**Class in C++**

The building block of C++ that leads to Object Oriented programming is a Class. It is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.

****

# **Inheritance in C++**

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important feature of Object Oriented Programming.

Sub Class: The class that inherits properties from another class is called Subclass or Derived Class.

Super Class:The class whose properties are inherited by subclass is called Base Class or Super class.

Implementing inheritance in C++:

// C++ program to demonstrate implementation

// of Inheritance

#include <bits/stdc++.h>

using namespace std;

//Base class

class Parent{

public:

int id\_p;

};

// Sub class inheriting from Base Class(Parent)

class Child : public Parent{

public:

int id\_c;

};

//main function

int main() {

Child obj1;

// An object of class child has all data members

// and member functions of class parent

obj1.id\_c = 7;

obj1.id\_p = 91;

cout << "Child id is " << obj1.id\_c << endl;

cout << "Parent id is " << obj1.id\_p << endl;

return 0;

}

# **Polymorphism in C++**

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

Implementing

// C++ program for function overloading

#include <bits/stdc++.h>

using namespace std;

class Geeks {

public:

// function with 1 int parameter

void func(int x) {

cout << "value of x is " << x << endl;

}

// function with same name but 1 double parameter

void func(double x) {

cout << "value of x is " << x << endl;

}

// function with same name and 2 int parameters

void func(int x, int y) {

cout << "value of x and y is " << x << ", " << y << endl;

}

};

int main() {

Geeks obj1;

// Which function is called will depend on the parameters passed

// The first 'func' is called

obj1.func(7);

// The second 'func' is called

obj1.func(9.132);

// The third 'func' is called

obj1.func(85,64);

return 0;

}

**Encapsulation in C++**

In normal terms Encapsulationis defined as wrapping up of data and information under a single unit. In Object Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulates them.

Implementing

// c++ program to explain Encapsulation

#include<iostream>

using namespace std;

class Encapsulation{

private:

// data hidden from outside world

int x;

public:

// function to set value of

// variable x

void set(int a)

{

x =a;

}

// function to return value of

// variable x

int get(){

return x;}

};

// main function

int main(){

Encapsulation obj;

obj.set(5);

cout<<obj.get();

return 0;

}

**JAVA**

Java is a set of [computer software](https://en.wikipedia.org/wiki/Computer_software) and specifications developed by [James Gosling](https://en.wikipedia.org/wiki/James_Gosling) at [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems), which was later acquired by the [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation), that provides a system for developing [application software](https://en.wikipedia.org/wiki/Application_software) and deploying it in a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) computing environment. Java is used in a wide variety of [computing platforms](https://en.wikipedia.org/wiki/Computing_platform) from [embedded devices](https://en.wikipedia.org/wiki/Embedded_device) and [mobile phones](https://en.wikipedia.org/wiki/Mobile_phone) to [enterprise servers](https://en.wikipedia.org/wiki/Enterprise_server)and [supercomputers](https://en.wikipedia.org/wiki/Supercomputer). [Java applets](https://en.wikipedia.org/wiki/Java_applet), which are less common than standalone Java applications, were commonly run in secure, [sandboxed](https://en.wikipedia.org/wiki/Sandbox_(computer_security)) environments to provide many features of native applications through being embedded in [HTML](https://en.wikipedia.org/wiki/HTML) pages. It's still possible to run Java in web browsers after most of them having dropped support for Java's VM.

The Java language has undergone several changes since the release of JDK ([Java Development Kit](https://en.wikipedia.org/wiki/Java_Development_Kit)) 1.0 on January 23, 1996, as well as numerous additions of classes and packages to the standard [library](https://en.wikipedia.org/wiki/Library_(computer_science)). Since J2SE 1.4 the [Java Community Process](https://en.wikipedia.org/wiki/Java_Community_Process) (JCP) has governed the evolution of the Java Language. The JCP uses *Java Specification Requests*(JSRs) to propose and specify additions and changes to the Java platform. The *Java Language Specification* (JLS) specifies the language; changes to the JLS are managed under JSR 901.

Sun released *JDK 1.1* on February 19, 1997. Major additions included an extensive retooling of the [AWT](https://en.wikipedia.org/wiki/Abstract_Window_Toolkit) event model, [inner classes](https://en.wikipedia.org/wiki/Inner_class) added to the language, [JavaBeans](https://en.wikipedia.org/wiki/JavaBean) and [JDBC](https://en.wikipedia.org/wiki/Java_Database_Connectivity).

*J2SE 1.2* (December 8, 1998) – Codename *Playground*. This and subsequent releases through J2SE 5.0 were rebranded *Java 2* and the version name "J2SE" ([Java 2 Platform, Standard Edition](https://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition)) replaced JDK to distinguish the base platform from J2EE ([Java 2 Platform, Enterprise Edition](https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition)) and J2ME ([Java 2 Platform, Micro Edition](https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition)). Major additions included [reflection](https://en.wikipedia.org/wiki/Reflection_(computer_science)), a collections framework, [Java IDL](https://en.wikipedia.org/wiki/Java_Interface_Definition_Language) (an [interface description language](https://en.wikipedia.org/wiki/Interface_description_language) implementation for [CORBA](https://en.wikipedia.org/wiki/CORBA) interoperability), and the integration of the [Swing](https://en.wikipedia.org/wiki/Swing_(Java)) graphical API into the core classes. A [Java Plug-in](https://en.wikipedia.org/wiki/Java_Plug-in) was released, and Sun's JVM was equipped with a JIT compiler for the first time.

*J2SE 1.3* (May 8, 2000) – Codename *Kestrel*. Notable changes included the bundling of the [HotSpot](https://en.wikipedia.org/wiki/HotSpot) JVM (the HotSpot JVM was first released in April, 1999 for the J2SE 1.2 JVM), [JavaSound](https://en.wikipedia.org/w/index.php?title=JavaSound&action=edit&redlink=1), Java Naming and Directory Interface (JNDI) and [Java Platform Debugger Architecture](https://en.wikipedia.org/wiki/Java_Platform_Debugger_Architecture) (JPDA).

*J2SE 1.4* (February 6, 2002) – Codename *Merlin*. This became the first release of the Java platform developed under the Java Community Process as JSR 59.Major changes included [regular expressions](https://en.wikipedia.org/wiki/Regular_expressions) modeled after [Perl](https://en.wikipedia.org/wiki/Perl), exception chaining, an integrated [XML](https://en.wikipedia.org/wiki/XML) parser and [XSLT](https://en.wikipedia.org/wiki/XSLT) processor ([JAXP](https://en.wikipedia.org/wiki/Java_API_for_XML_Processing)), and [Java Web Start](https://en.wikipedia.org/wiki/Java_Web_Start).

*J2SE 5.0* (September 30, 2004) – Codename *Tiger*. It was originally numbered 1.5, which is still used as the internal version number. Developed under JSR 176, Tiger added several significant new language features including the [for-each loop](https://en.wikipedia.org/wiki/For-each_loop), [generics](https://en.wikipedia.org/wiki/Generic_programming), [autoboxing](https://en.wikipedia.org/wiki/Autoboxing) and [var-args](https://en.wikipedia.org/wiki/Variadic_function).

*Java SE 6* (December 11, 2006) – Codename *Mustang*. It was bundled with a database manager and facilitates the use of scripting languages with the JVM (such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript)using [Mozilla](https://en.wikipedia.org/wiki/Mozilla)'s Rhino engine). As of this version, Sun replaced the name "J2SE" with *Java SE* and dropped the ".0" from the version number.Other major changes include support for pluggable [annotations](https://en.wikipedia.org/wiki/Java_annotation) ([JSR 269](https://en.wikipedia.org/w/index.php?title=JSR_269&action=edit&redlink=1)), many [GUI](https://en.wikipedia.org/wiki/GUI) improvements, including native UI enhancements to support the look and feel of [Windows Vista](https://en.wikipedia.org/wiki/Windows_Vista), and improvements to the Java Platform Debugger Architecture (JPDA) & JVM Tool Interface for better monitoring and troubleshooting.

*Java SE 7* (July 28, 2011) – Codename *Dolphin*. This version developed under JSR 336.It added many small language changes including strings in switch, try-with-resources and [type inference](https://en.wikipedia.org/wiki/Type_inference) for generic instance creation. The JVM was extended with support for dynamic languages, while the class library was extended among others with a join/fork framework, an improved new file I/O library and support for new network protocols such as SCTP. Java 7 Update 76 was released in January 2015, with expiration date April 14, 2015.

In June 2016, after the last public update of Java 7, "[remotely exploitable](https://en.wikipedia.org/wiki/Exploit_(computer_security))" security bugs in Java 6, 7, and 8 were announced.

*Java SE 8* (March 18, 2014) – Notable changes include language-level support for [lambda expressions](https://en.wikipedia.org/wiki/Lambda_(programming)) ([closures](https://en.wikipedia.org/wiki/Closure_(computer_science))) and default methods, the Project Nashorn JavaScript runtime, a new Date and Time API inspired by Joda Time, and the removal of PermGen. This version is not officially supported on the Windows XP platform.However, due to the end of Java 7's lifecycle it is the recommended version for XP users. Previously, only an unofficial manual installation method had been described for Windows XP SP3. It refers to JDK8, the developing platform for Java that also includes a fully functioning [Java Runtime Environment](https://en.wikipedia.org/wiki/Java_Runtime_Environment). Java 8 is supported on Windows Server 2008 R2 SP1, Windows Vista SP2 and Windows 7 SP1, Ubuntu 12.04 LTS and higher (and some other OSes).

*Java SE 9* and *10* had higher system requirements, i.e. Windows 7 or Server 2012 (and web browser minimum certified is upped to [Internet Explorer 11](https://en.wikipedia.org/wiki/Internet_Explorer_11) or other web browsers), and Oracle dropped [32-bit](https://en.wikipedia.org/wiki/32-bit) compatibility for all platforms, i.e. only Oracle's "[64-bit](https://en.wikipedia.org/wiki/64-bit_computing) Java virtual machines (JVMs) are certified".

*Java SE 11* was released September 2018, the first [LTS](https://en.wikipedia.org/wiki/Long-term_support) release since the [rapid release](https://en.wikipedia.org/wiki/Release_early,_release_often) model was adopted starting with version 9. For the first time, OpenJDK 11 represents the complete source code for the Java platform under the GNU General Public License, and while Oracle still dual-licenses it with an optional proprietary license, there are no code differences nor modules unique to the proprietary-licensed version. Java 11 features include two new garbage collector implementations, Flight Recorder to debug deep issues, a new HTTP client including [WebSocket](https://en.wikipedia.org/wiki/WebSocket) support.[[44]](https://en.wikipedia.org/wiki/Java_(software_platform)#cite_note-44)

In addition to language changes, significant changes have been made to the Java class library over the years, which has grown from a few hundred classes in JDK 1.0 to over three thousand in J2SE 5.0. Entire new APIs, such as [Swing](https://en.wikipedia.org/wiki/Swing_(Java)) and [Java 2D](https://en.wikipedia.org/wiki/Java_2D), have evolved, and many of the original JDK 1.0 classes and methods have been deprecated.

**Classes in JAVA**

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.

Implementing

// Class Declaration

public class Dog{

// Instance Variables

String name;

String breed;

int age;

String color;

// Constructor Declaration of Class

public Dog(String name, String breed,

int age, String color){

this.name = name;

this.breed = breed;

this.age = age;

this.color = color;

}

// method 1

public String getName(){

return name;

}

// method 2

public String getBreed(){

return breed;

}

// method 3

public int getAge(){

return age;

}

// method 4

public String getColor(){

return color;

}

@Override

public String toString()

{

return("Hi my name is "+ this.getName()+

".\nMy breed,age and color are " +

this.getBreed()+"," + this.getAge()+

","+ this.getColor());

}

public static void main(String[] args)

{

Dog tuffy = new Dog("tuffy","papillon", 5, "white");

System.out.println(tuffy.toString());

}

}

# **Inheritance in Java**

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.

Implementing

//Java program to illustrate the

// concept of inheritance

// base class

class Bicycle {

// the Bicycle class has two fields

public int gear;

public int speed;

// the Bicycle class has one constructor

public Bicycle(int gear, int speed{

this.gear = gear;

this.speed = speed;

}

// the Bicycle class has three methods

public void applyBrake(int decrement){

speed -= decrement;

}

public void speedUp(int increment){

speed += increment;

}

// toString() method to print info of Bicycle

public String toString()

{

return("No of gears are "+gear+"\n"+ "speed of bicycle is "+speed);

}

}

// derived class

class MountainBike extends Bicycle {

// the MountainBike subclass adds one more field

public int seatHeight;

// the MountainBike subclass has one constructor

public MountainBike(int gear,int speed, int startHeight){

// invoking base-class(Bicycle) constructor

super(gear, speed);

seatHeight = startHeight;

}

// the MountainBike subclass adds one more method

public void setHeight(int newValue){

seatHeight = newValue;

}

// overriding toString() method

// of Bicycle to print more info

@Override

public String toString() {

return (super.toString()+

"\nseat height is "+seatHeight);

}

}

// driver class

public class Test {

public static void main(String args[]){

MountainBike mb = new MountainBike(3, 100, 25);

System.out.println(mb.toString());

}

}

## **Java Polymorphism**

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

Implementing:

class Animal {

public void animalSound() {

System.out.println("The animal makes a sound");

}

}

class Pig extends Animal {

public void animalSound() {

System.out.println("The pig says: wee wee");

}

}

class Dog extends Animal {

public void animalSound() {

System.out.println("The dog says: bow wow");

}

}

## **Encapsulation**

The meaning of Encapsulation, is to make sure that "sensitive" data is hidden from users.

Implementing:

public class Person {

private String name; // private = restricted access

// Getter

public String getName() {

return name;

}// Setter

public void setName(String newName) {

this.name = newName; }}

**Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

## **Python Classes**

Python is an object oriented programming language. Almost everything in Python is an object, with its properties and methods. A Class is like an object constructor, or a "blueprint" for creating objects.

Implementing

class MyClass:

x = 5

## **Python Inheritance**

Inheritance allows us to define a class that inherits all the methods and properties from another class. Parent class is the class being inherited from, also called base class. Child class is the class that inherits from another class, also called derived class.

Implementing

class Person:

def \_\_init\_\_(self, fname, lname):

self.firstname = fname

self.lastname = lname

def printname(self):

print(self.firstname, self.lastname)

#Use the Person class to create an object, and then execute the printname method:

x = Person("John", "Doe")

x.printname()

**Polymorphism**

Polymorphism is based on the greek words Poly (many) and morphism (forms). We will create a structure that can take or use many forms of objects.

Implementing

We create two classes: Bear and Dog, both can make a distinct sound. We then make two instances and call their action using the same method.

**class** Bear(object):  
**def** sound(self):  
**print** "Groarrr"  
   
**class** Dog(object):  
**def** sound(self):  
**print** "Woof woof!"  
   
**def** makeSound(animalType):  
animalType.sound()  
   
bearObj = Bear()  
dogObj = Dog()  
   
makeSound(bearObj)  
makeSound(dogObj)

**Encapsulation**

In an object oriented python program, you can *restrict access* to methods and variables. This can prevent the data from being modified by accident and is known as *encapsulation*.

**class** Car:  
   
**def** \_\_init\_\_(self):  
self.\_\_updateSoftware()  
   
**def** drive(self):  
**print** 'driving'  
   
**def** \_\_updateSoftware(self):  
**print** 'updating software'  
   
redcar = Car()  
redcar.drive()  
*#redcar.\_\_updateSoftware() not accesible from object.*

**C#**

C# is a general object-oriented programming (OOP) language for networking and Web development. C# is specified as a common language infrastructure (CLI) language.

In January 1999, Dutch software engineer Anders Hejlsberg formed a team to develop C# as a complement to Microsoft’s NET framework. Initially, C# was developed as C-Like Object Oriented Language (Cool). The actual name was changed to avert potential trademark issues. In January 2000, NET was released as C#. Its NET framework promotes multiple Web technologies.

# **C# | Class and Object**

Class and Object are the basic concepts of Object Oriented Programming which revolve around the real-life entities. A class is a user-defined blueprint or prototype from which objects are created. Basically, a class combines the fields and methods(member function which defines actions) into a single unit. In C#, classes support the polymorphism, inheritance and also provide the concept of derived classes and base classes.

// declaring public class  
public class Geeks  
{  
  
 // field variable  
 public int a, b;  
  
 // member function or method  
 public void display()  
 {  
 Console.WriteLine(“Class & Objects in C#”);  
 }  
}

# 

# **C# | Inheritance**

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in C# by which one class is allowed to inherit the features(fields and methods) of another class.

using System;  
  
namespace InheritanceApplication {  
 class Shape {  
 public void setWidth(int w) {  
 width = w;  
 }  
 public void setHeight(int h) {  
 height = h;  
 }  
 protected int width;  
 protected int height;  
 }  
  
 // Derived class  
 class Rectangle: Shape {  
 public int getArea() {   
 return (width \* height);   
 }  
 }  
 class RectangleTester {  
 static void Main(string[] args) {  
 Rectangle Rect = new Rectangle();  
  
 Rect.setWidth(5);  
 Rect.setHeight(7);  
  
 // Print the area of the object.  
 Console.WriteLine("Total area: {0}", Rect.getArea());  
 Console.ReadKey();  
 }  
 }  
}

**polymorphism**

The word polymorphism means having many forms. In object-oriented programming paradigm, polymorphism is often expressed as 'one interface, multiple functions'.

using System;  
  
namespace PolymorphismApplication {  
 class Printdata {  
 void print(int i) {  
 Console.WriteLine("Printing int: {0}", i );  
 }  
 void print(double f) {  
 Console.WriteLine("Printing float: {0}" , f);  
 }  
 void print(string s) {  
 Console.WriteLine("Printing string: {0}", s);  
 }  
 static void Main(string[] args) {  
 Printdata p = new Printdata();  
   
 // Call print to print integer  
 p.print(5);  
   
 // Call print to print float  
 p.print(500.263);  
   
 // Call print to print string  
 p.print("Hello C++");  
 Console.ReadKey();  
 }  
 }  
}

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **C# | Encapsulation**

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In a different way, encapsulation is a protective shield that prevents the data from being accessed by the code outside this shield.

using System;  
  
namespace RectangleApplication {  
 class Rectangle {  
 //member variables  
 public double length;  
 public double width;  
   
 public double GetArea() {  
 return length \* width;  
 }  
 public void Display() {  
 Console.WriteLine("Length: {0}", length);  
 Console.WriteLine("Width: {0}", width);  
 Console.WriteLine("Area: {0}", GetArea());  
 }  
 }//end class Rectangle  
   
 class ExecuteRectangle {  
 static void Main(string[] args) {  
 Rectangle r = new Rectangle();  
 r.length = 4.5;  
 r.width = 3.5;  
 r.Display();  
 Console.ReadLine();  
 }  
 }  
}

|  |  |
| --- | --- |
| Paradigma Procedural | Paradigma Orientada a Objetos |
| Object-oriented Programming uses classes and objects. | Procedural Programming takes on applications by solving problems from the top of the code down to the bottom. |
| When developing with Object-oriented Programming a developer does not simply sit down and begin to write out code on a page that comes to their mind. The application must be planned out in a series of steps and diagrams. | When developing with Procedural Programming a developer may take a much different approach to designing an application. Procedural Programming takes a more top down approach to writing an application and while a developer who uses Object-oriented Programming to create applications would think of planning out the program with re-usable classes, a developer who uses Procedural Programming might plan out the program without the idea of recycling code. |
| When developing in Object-oriented Programming you begin with diagramming and planning out the program. Classes are constructed and data fields are associated to help the developer in the planning process. As the classes are built, functions and methods can be introduced to help in the planning of the program. | When developing in Procedural Programming you start first with the main functions or as some call it the common procedures. This main function is the main goal of the system, or what it is supposed to do for the person using it. |
| When developing in Object-oriented Programming there are many tools available that will aid the programmer while developing. Many languages have software built to help aid developers while coding in their language. | Unlike Object-oriented Programming, Procedural Programming can use a much wider array of programs to develop with. The reason for this is because most code in Procedural Programming can be written in anything from Microsoft word to Adobe Dreamweaver. |
| The obvious advantages of using Object-oriented Programming is its reliability and sustainability across different platforms. Developers who have experience programming in JAVA would also be qualified to develop in C# or JQuery which have very similar syntax. The benefit to employers looking for developers with a special area of expertise is that they are able to train with ease developers who have Object-oriented Programming experience vs. the ones who don’t. | One glaring advantage of Procedural Programming is the ability to jump right into coding a program without the need to create any objects or classes. Many programmers prefer this method because they want to be able to write small amounts of code in a short period of time without the need to plan it out. Examples could include simple forms or html pages with basic data displayed. |
| One disadvantage of Object-oriented Programming is the difficulty of understanding how objects, classes, methods, actions etc relate to each other. Especially for those who are used to a Procedural Programming approach to programming this may be very difficult. | A major disadvantage of using Procedural Programming as a method of programming is the inability to reuse code throughout the program. Having to rewrite the same type of code many times throughout a program can add to the development cost and time of a project. |

# Conclusiones

There are many advantages to programming in both Object-oriented Programming and Procedural Programming languages. Many argue the superiority of each method, but both serve their purpose and both have their advantages and disadvantages.

**Bibliografía**  
(s.f.). Obtenido de https://en.wikipedia.org/wiki/C\_(programming\_language)  
.techopedia. (s.f.).

Obtenido de <https://www.techopedia.com/definition/26272/c-sharp>

ELIASON, K. (1 de Agosto de 2013). neonbrand.com. Obtenido de <https://neonbrand.com/website-design/procedural-programming-vs-object-oriented-programming-a-review/>

excelwithbusiness.com. (s.f.). Obtenido de <https://excelwithbusiness.com/blog/say-hello-world-in-28-different-programming-languages/?fbclid=IwAR2DWkyM0FdIEybCNblSfnenYPEbfz5HmThDxdfLtX3CMFNlvhcIa-rcVes>

geeksforgeeks. (s.f.). Obtenido de <https://www.geeksforgeeks.org/c-classes-and-objects/>

python.org. (s.f.). Obtenido de <https://www.python.org/doc/essays/blurb/>

pythonspot. (s.f.). Obtenido de <https://pythonspot.com/polymorphism/>

techopedia.com. (s.f.). Obtenido de <https://www.techopedia.com/definition/3940/pascal>

w3schools. (s.f.). Obtenido de <https://www.w3schools.com/java/java_polymorphism.asp>

Wikipedia. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Java\_(software\_platform)