



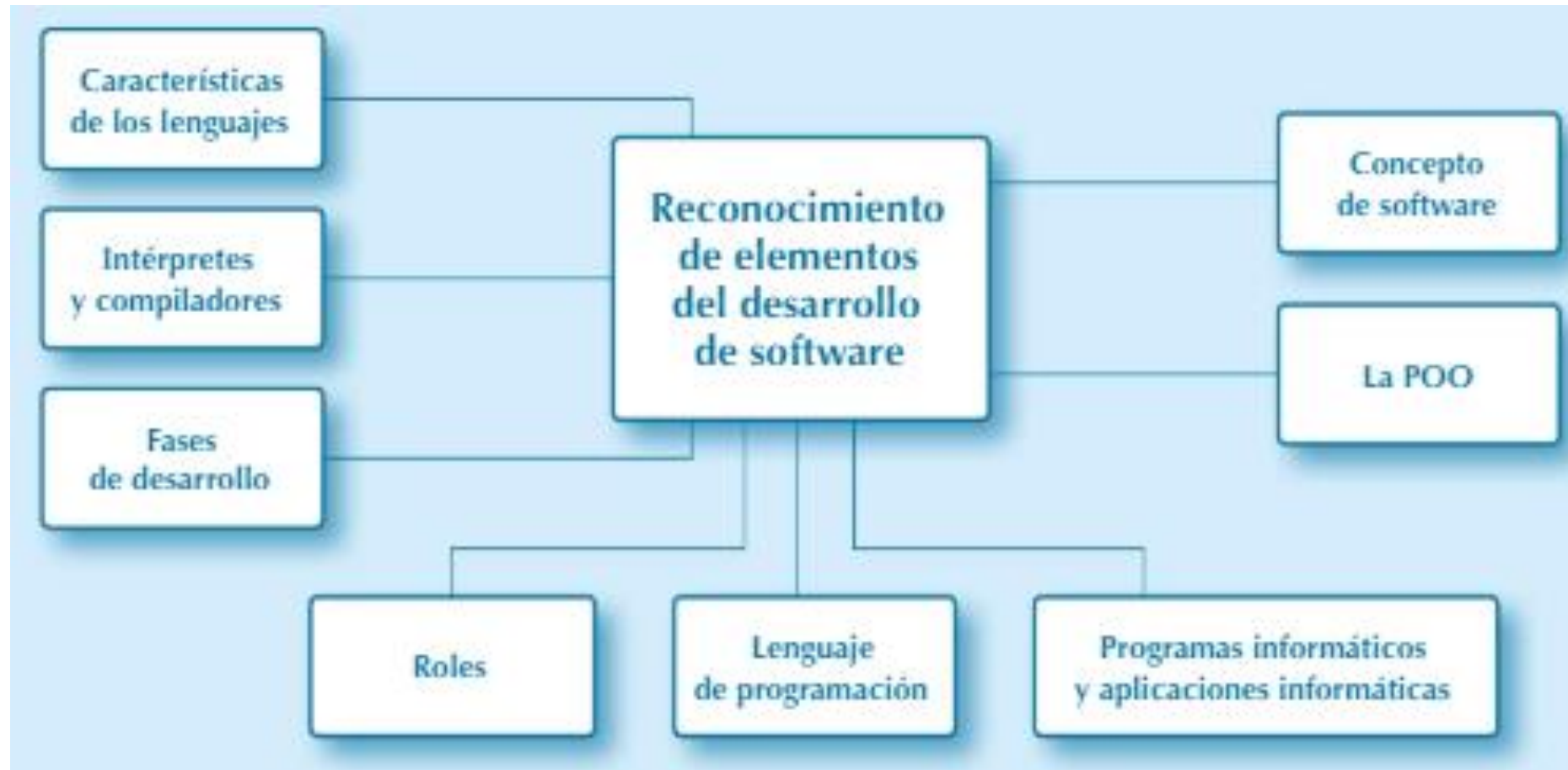
UNIDAD 1. RECONOCIMIENTO DEL DESARROLLO DE ELEMENTOS DE SOFTWARE

VICENT MARTÍ

OBJETIVOS

- Comprender los principios básicos del desarrollo del software.
- Conocer las principales características de los lenguajes de programación.
- Conocer las fases de desarrollo de las aplicaciones.
- Reconocer las diferencias entre compiladores, intérpretes y máquinas virtuales.

MAPA CONCEPTUAL



CONTENIDO

1. Introducción
2. Programas informáticos y aplicaciones informáticas
3. Lenguajes de programación
4. Desarrollo de una Aplicación
5. El paradigma de la Programación Orientada a Objetos

1. INTRODUCCIÓN

Todo sistema informático está formado por una **parte física** con elementos electromecánicos que se denomina **HARDWARE**, y una **parte lógica** con elementos como los datos, programas y su documentación que se denomina **SOFTWARE**.

Puede definirse software como “el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación” (definición extraída del estándar 729 de IEEE).

El concepto de software fue utilizado por Charles Babbage (1791-1871), cuando trabajaba en su máquina diferencial, utilizaba series de instrucciones que se leían desde la memoria principal del sistema y a las que denominaba software.

En la historia más reciente los lenguajes de programación se ha caracterizado por como ha evolucionado y se observan sus avances agrupados en lo que se llaman generaciones, así se distinguen hasta 5 generaciones de lenguajes de programación.

GENERACIONES DE LENGUAJES DE PROGRAMACIÓN

Primera Generación: A principios de los años 50. Los primeros ordenadores se programaban directamente en código máquina (binario).

Segunda generación: Lenguajes del tipo ensamblador, simplifican la escritura de las instrucciones y son más legibles que sus antecesores.

Tercera Generación: Son lenguajes de alto nivel, aumentan el nivel de abstracción. Son usados en ámbitos computacionales y de gestión empresarial, donde mejoran el rendimiento de la programación. Surgen a partir de 1957 con el Fortran, Cobol (1960) y Pascal (1970), Smalltalk, C y posteriormente C++, C#, Java y PHP, entre otros, estos proporcionan bibliotecas. Surgen nuevos paradigmas programación modular y estructurada, POO.

Cuarta generación: Herramientas que permiten construir aplicaciones combinando piezas prefabricadas. Facilitan acceso a base de datos, capacidades gráficas, generación de código, etc. Algunos ejemplos son: 4GL (Informix) y PL/SQL (Oracle).

Quinta generación: en ocasiones se llama así a los lenguajes de inteligencia artificial. Ejemplos: Lisp, Prolog y OPS5. Algunos autores incluyen aquí la POO, con entornos gráficos de desarrollo: como las .NET, Eclipse, Netbeans, etc.

CARACTERÍSTICAS DEL SW

El software tiene características propias que lo definen:

- **Es lógico, no físico. Es intangible.**
- **Se desarrolla, no se fabrica.**
- **Se puede copiar y tiene las características del original.**
- **En ocasiones puede construirse a medida de las necesidades**

EJERCICIO 1



Escribe un documento y súbelo al aula virtual (AV)

- Importancia de Alan Turing en la Segunda Guerra Mundial.
- Define el firmware.

2. PROGRAMAS INFORMÁTICOS Y APLICACIONES INFORMÁTICAS

- Un programa es una serie de órdenes o instrucciones secuenciadas u ordenadas con una finalidad concreta y que realizan una función determinada.
- Vemos el ejemplo clásico de Hola mundo en lenguaje C:





















```
/* Programa holamundo.c */  
Esta línea no realiza ninguna función solo dice cuál es el nombre del pro-  
grama.  
#include <stdio.h>  
Esta línea es necesaria si va a sacarse algo por pantalla.  
main ()  
Esta línea indica que esto es lo primero que va ejecutar el programa (lo  
contenido entre { y } ).{  
    printf ("Hola Mundo");  
    Esta línea muestra las palabras Hola mundo por pantalla.  
}
```

APLICACIONES INFORMÁTICAS

- Existen en el mercado muchas aplicaciones informáticas y cada una tiene su uso y finalidad.
- Suelen estar formadas por varios programas que suelen compartir librerías y datos.
- Si cada programa se ejecuta de manera independiente se llama suite o paquete integrado (ejemplo Open Office)
 - Programas de contabilidad
 - Programas de diseño gráfico
 - Procesadores de texto
 - Programa de facturación
 - Multimedia
 - Bases de datos
 - Hojas de cálculo
 - Presentaciones
 - Correo electrónico

3. LENGUAJES DE PROGRAMACIÓN

- Todos los programas se desarrollan en algún lenguaje de programación.
- Actualmente, nadie programa directamente en código máquina, ya que produciría muchos errores.
- Los LP son por tanto lenguajes artificiales creados para que, al traducirse a código máquina, cada una de las instrucciones de lugar a una o varias instrucciones máquina..
- Utilizan una sintaxis y un conjunto de normas y palabras reservadas.

1	Java		11	MATLAB	
2	C		12	R	
3	Python		13	Perl	
4	C++		14	Assembly Language	
5	Visual Basic .NET		15	Swift	
6	Javascript		16	Go	
7	C#		17	Delphi/Object Pascal	
8	PHP		18	Ruby	
9	SQL		19	PL/SQL	
10	Objective-C		20	Visual Basic	

ELEMENTOS DE LOS LP

Los LP utilizan:

- **Identificadores:** Nombres simbólicos que se da a ciertos elementos de la programación como variables, tipos, funciones, etc. Algunas son parte del léxico que utiliza el lenguaje para sus instrucciones, son palabras reservadas propias de la sintaxis que utiliza.
- **Tipos de datos:** como los numéricos (enteros y reales), caracteres y cadenas, booleanos; más complejos como estructuras estáticas como vectores, registros, tablas y estructuras dinámicas como árboles, grafos, etc.
- **Constantes:** no cambian su valor durante la ejecución del programa.
- **Variables:** identifican zonas de memoria que contiene un dato, que se puede modificar durante la ejecución del programa.
- **Operadores:** son las operaciones que hacemos con los datos. Aritméticos (+, -, *, /) lógicos (and, or, not), relacionales (>, <, >=, <=, <>), asignación (:=), de cadenas (&), etc.
- **Expresiones:** combinación de operandos (variables, constantes, otras expresiones) y operadores.
- **Instrucciones:** primitivas como entradas, asignación y salida; otras de control como las secuencia, alternativas e iterativas.
- **Comentarios:** texto para documentar los programas, no se ejecuta.

EJERCICIO 2



- Destacan, según el índice de Programación Tiobe, que es uno de los más fiables: Java (15%), C (14%), C++ (8,8%), y Python (8,1%), se ha incluido el % de implantación en abril 2019. Resaltar C y C++, y su resurgimiento gracias a dispositivos IoT (Internet of things) y el crecimiento imparable de Python.

Escribe un documento y súbelo al aula virtual (AV)

- Con una tabla los índices Tiobe a la última actualización.
- Analiza el ranking de los 5 primeros.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según como se desarrolla

Distinguimos entre software a medidas y estándar

Software a medida

Necesita de un tiempo de desarrollo.

Se adapta a las necesidades específicas de la empresa.

No suele ser trasladable a otras empresas.

Suele contener errores y se necesita de una etapa de mantenimiento.

Más costoso que el software estándar.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según como se desarrolla (continua)

Software estándar

Se compra ya hecho.

Suele tener muchos menos errores que el software a medida.

Suele ser más barato que el software a medida.

Generalmente tiene funciones que la empresa no usará y también carecerá de otras opciones.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según proximidad al lenguaje Máquina

Lenguaje de bajo nivel. (Lenguaje en código máquina)

- Específico de cada tipo de máquina, que no necesita ser traducido, la máquina lo entiende directamente. Combinaciones de unos y ceros.

Lenguaje de nivel medio. (Ensamblador)

- Cada instrucción equivale a una instrucción en el lenguaje máquina, utiliza para su escritura palabras nemotécnicas en lugar de cadenas de bits.
- Los programas hechos en lenguaje ensamblador son generalmente rápidos y consumen pocos recursos del sistema, pero son menos portables entre distintas máquinas que sus sucesores, los de alto nivel.

Lenguaje de alto nivel

- Independencia de la máquina, pudiendo utilizar un mismo programa en diferentes equipos con la única condición de disponer de un programa traductor o compilador, que es suministrado por el fabricante.
- Aproximarse al lenguaje natural, sustituye nemotécnicos por sentencias más comprensibles: if_then_else, while, etc.
- Incluir librerías del lenguaje, se pueden utilizar siempre que se quiera sin necesidad de programarlas cada vez. En ocasiones ofrece frameworks para una programación más eficiente.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según funcionalidad

Nos referimos al uso principal o destino para el que se concibió el lenguaje, algunos ya están en desuso:

- **Lenguajes para la educación**, como: Logo, Pilot y Eiffel.
- **De propósito general**. En diversos campos, como: Pascal, Modula, C y C++.
- **Lenguajes de inteligencia artificial**. Programas que emulan comportamientos inteligentes. Algoritmos de juegos (ajedrez, etc.), programas para comprender el lenguaje natural, robótica, etc. Como: Lisp, Prolog,
- **Orientados a la gestión**. Las aplicaciones de nóminas, gestión comercial, etc. Se caracterizan por un gran volumen de datos a manejar y por unas determinadas estructuras, como: Cobol y PL/SQL, ERP: ABAP (SAP), C/AL (NAVISION-Microsoft).
- **Científicos**. Pocas operaciones entrada/salida y cálculos complejos: Fortran.
- **Estadísticos**. Con muchas funciones para minería y bigdata, como: R y Python.
- **Orientados a Internet**. Transversales a la máquina y al sistema operativo, como: HTML, JAVA, C# y PHP.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según paradigma de Programación

En general, la mayoría de paradigmas son variantes de dos tipos: los **imperativos resuelven el CÓMO**, mientras que los **declarativos especifican el QUÉ**. A continuación, se describen algunos de los paradigmas de programación.

- **Programación imperativa o por procedimientos:** Se basa en dar instrucciones al ordenador de cómo hacer las cosas en forma de algoritmos, en lugar de describir el problema o la solución. Es la forma de programación más usada y la más antigua. Ejemplos: C, BASIC o Pascal. Un caso específico son los **lenguajes de programación estructurados**, como Pascal, surgen en la década de los setenta. Coincide su difusión con la falta de fiabilidad del software en general y de los programas más complejos. Sus técnicas utilizan el concepto de TOP-DOWN, se va de lo más general a lo más particular, dividiendo el problema en otros más pequeños y sencillos. Se basan en **Teorema de estructuras**, que dice: **Dado un programa P propio (un punto de entrada y uno de salida y al menos un camino entre ambos) y no estructurado, podemos obtener otro programa P1, pero construido siguiendo la programación estructurada que combine sólo tres estructuras lógicas o de control.**
 - Secuencia: ejecución de una instrucción tras otra.
 - Selección o alternativa: ejecuta unas instrucciones u otras, según una condición.
 - Iteración o bucle: ejecución de unas instrucciones mientras cumpla una condición.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según paradigma de Programación (continúa)

- **Programación orientada a objetos (POO)** : La POO eleva el nivel de abstracción pero no deja de ser una programación imperativa. Encapsula elementos denominados objetos que incluyen tanto variables (datos) como funciones (métodos). El lenguaje SIMULA fue el primero en agrupar datos y procedimientos en una sola entidad. En una aplicación OO todo son objetos, el cómputo avanza por interacción entre ellos. Estas interacciones se realizan por medio de peticiones o mensajes. Ofrece características como el encapsulado, polimorfismo o la herencia. Ejemplos: C++, C#, Java o Python.
- **Programación dinámica**: Consiste en trocear los problemas en partes pequeñas para analizarlos y resolverlos de forma lo más cercana al óptimo, sin usar métodos recursivos. Este paradigma está basado en el modo de realizar los algoritmos y se puede usar con cualquier lenguaje imperativo. Richard Bellman concluye que si las soluciones parciales son óptimas, la secuencia de todas ellas también lo será.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según paradigma de Programación (continúa)

Programación dirigida por eventos: Es un paradigma en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, ya sean definidos o provocados por el usuario.

Programación declarativa: Está basada en describir el problema declarando propiedades y reglas que deben cumplirse, en lugar de instrucciones. La solución es obtenida sin especificar exactamente cómo encontrarla, se le indica a la computadora qué es lo que se desea.

Programación funcional: Basada en la definición los predicados, de corte más matemático. Ej: Scheme (una variante de Lisp). Se emplea en Inteligencia Artificial.

Programación lógica: Utiliza relaciones lógicas, está representado por Prolog.

Programación con restricciones: Similar a la lógica usando ecuaciones. Casi todos los lenguajes son variantes del Prolog.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según paradigma de Programación (continúa)

Programación multiparadigma: Es el uso de dos o más paradigmas dentro de un programa. Por ejemplo, Python que es orientado a objetos, reflexivo, imperativo y funcional. Otros como: JAVA, C++ y PHP combinan el paradigma imperativo con OO. (reflexivo: meta-sistema conectado a si mismo, un objeto en si mismo).

Programación reactiva: Se basa en la declaración de una serie de objetos emisores de eventos asíncronos y otra serie de objetos que se "suscriben" a los primeros, es decir, quedan a la escucha de la emisión de eventos de estos y "reaccionan" a los valores que reciben. Es muy común usar la librería Rx de Microsoft (Reactive Extensions), disponible para múltiples lenguajes.

Lenguaje específico del dominio o DSL: Desarrollados para resolver un problema específico. El más representativo sería SQL, para el manejo de las bases de datos, es de tipo declarativo, pero los hay imperativos, como el Logo.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

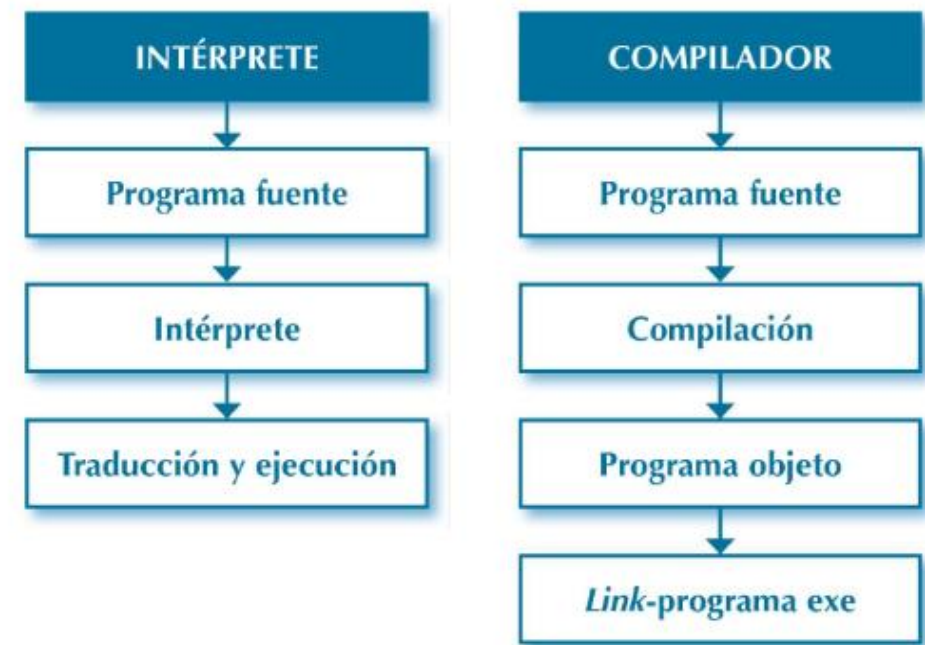
Según traducción a código máquina (continúa)

- Con una visión clásica tenemos, por un lado, los lenguajes de programación **compilados**, que utilizan compiladores para traducir el código fuente de un lenguaje de programación al código máquina del sistema, y por otro lado, los **interpretados**, que realizan la traducción a medida que se van ejecutando. También existen lenguajes **virtuales**, más portables que los compilados pues se generan tras la compilación a un código intermedio bytecode. Este código a su vez es interpretado por una máquina virtual instalada en cualquier equipo, como en Java.
 - **Código fuente:** Programas escritos en un determinado lenguaje de programación
 - **Código objeto:** Código que resulta de la compilación del código fuente.
 - **Código ejecutable:** Código obtenido del proceso de enlazar todos los archivos de código objeto con un programa llamado enlazador.
- Pero con una visión más actual, los términos de lenguaje interpretado y compilado no es tan clara, en las implementaciones más modernas de un lenguaje de programación, se ofrecen ambas opciones. Al clasificarlos nos referiremos a su práctica más usual.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según traducción a código máquina (continúa)

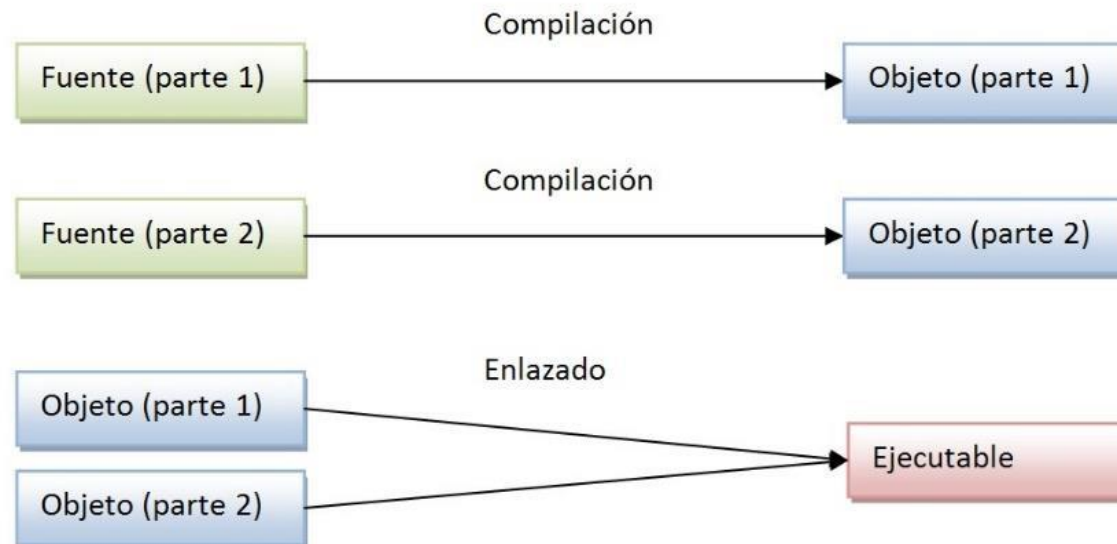
- Con una visión clásica tenemos, por un lado, los lenguajes de programación **compilados**, que utilizan compiladores para traducir el código fuente de un lenguaje de programación al código máquina del sistema, y por otro lado, los **interpretados**, que realizan la traducción a medida que se van ejecutando. También existen lenguajes **virtuales**, más portables que los compilados pues se generan tras la compilación a un código intermedio bytecode. Este código a su vez es interpretado por una máquina virtual instalada en cualquier equipo.



TIPOS DE LENGUAJES DE PROGRAMACIÓN

Según traducción a código máquina (continúa)

- **Código fuente:** Programas escritos en un determinado lenguaje de programación
- **Código objeto:** Código que resulta de la compilación del código fuente.
- **Código ejecutable:** Código obtenido del proceso de enlazar todos los archivos de código objeto con un programa llamado enlazador.



TIPOS DE LENGUAJES DE PROGRAMACIÓN

Interpretados (sentencia traducida, sentencia ejecutada)

- Sintaxis más elaborada que la de los ensambladores.
- La velocidad de traducción depende de la complejidad de la sintaxis.
- Sintaxis sencillas como en BASIC. Suelen ser más lentos que los compilados.
- Uno de los principales problemas es que una sentencia que se ejecuta varias veces tendrá que traducirse cada vez.
- Son más flexibles para reemplazar partes del programa. Ofrecen un entorno no dependiente de la máquina, pasa a depender del intérprete (máquina virtual). Un ejemplo es el bytecode, que es un compilado comprimido y optimizado del código fuente (genera un código intermedio), como ocurre en las versiones iniciales de máquinas virtuales que ejecutan Java. Se les acusaba de ser lentos.
- Una opción para mejorar la velocidad del bytecode es el código fuente en un árbol de sintaxis abstracta optimizado (AST), que se complementa con la compilación justo a tiempo (JIT), donde la compilación es dinámica del código intermedio a código máquina nativo en tiempo de ejecución. Se usa en Java, Python, .NET.
- En la actualidad, uno de los entornos más comunes de uso de los intérpretes es en los navegadores web, debido a que tienen de ejecutarse independientemente de la plataforma.



TIPOS DE LENGUAJES DE PROGRAMACIÓN

Compilados

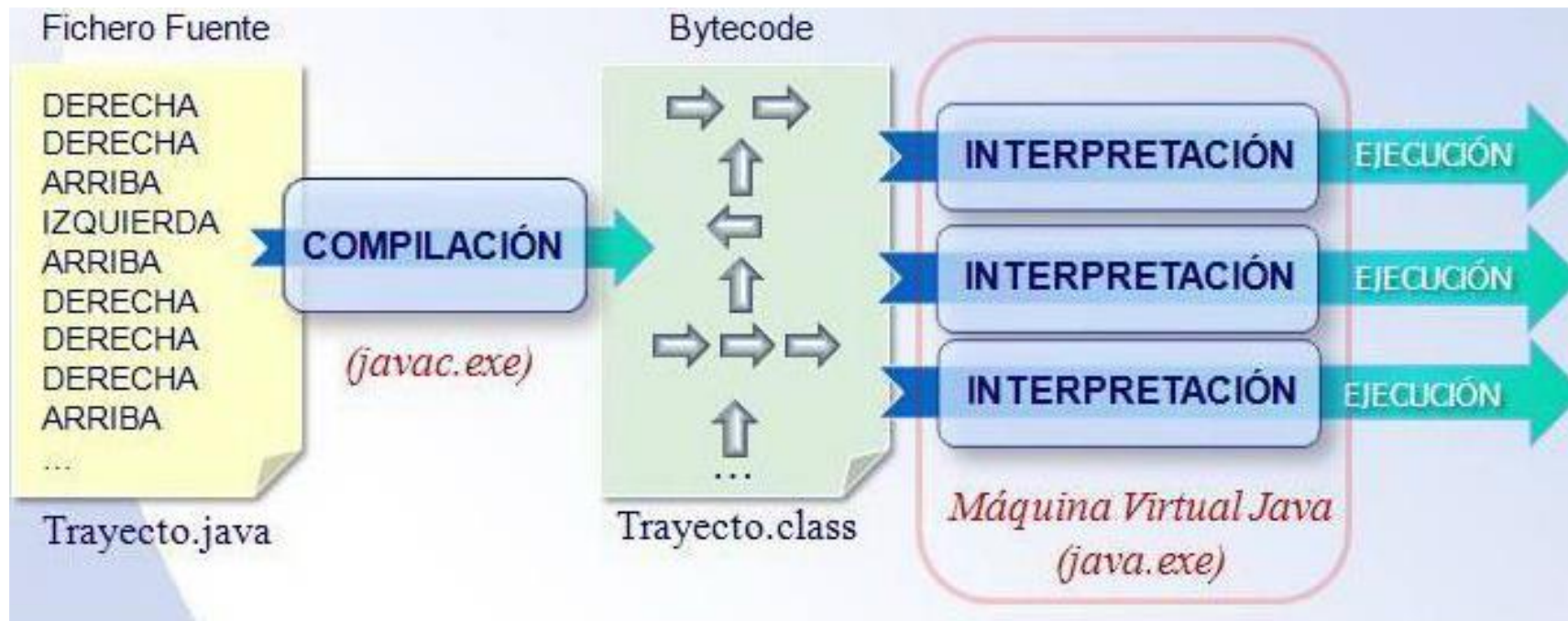
Los programas compilados requieren el uso de un compilador que convierte el código fuente a código máquina, para ello pasa por varias fases:



- **Analizador léxico:** Lee el fuente, carácter a carácter generando una secuencia de unidades léxicas llamadas “tokens” que va introduciendo en la tabla de símbolos. Los tokens son de identificadores, constantes, operadores, etc. Se verifica que se escriben correctamente. Ejemplos de errores léxicos en Pascal: escribir TPROGRAM en lugar de PROGRAM, Identificador de variable que empiece con un número: 1abc
- **Analizador Sintáctico:** A partir de la lista de “tokens” se construyen agrupaciones jerárquicas y se determina si se cumplen las restricciones sintácticas de la gramática del lenguaje. Paréntesis mal balanceados, dejarse “puntos y coma”, un “else” sin “then”
- **Analizador Semántico:** Se realizan diversas comprobaciones a partir de las estructuras jerárquicas de la fase anterior como: Comprobación de tipos de datos, el índice de un array esté en concordancia con la declaración, ejemplo si es bidimensional tendremos que poner dos índices.
- **Código intermedio:** Consiste en un código abstracto independiente de la máquina. A partir de este código se permite la portabilidad a otras máquinas. Este concepto de código intermedio tiene grandes similitudes con los bytecodes generados por JAVA.
- **Optimizador de código:** Permite mejorar el rendimiento de los procesos. En lugar de traducir $A := 2 * 3 - 1$ se traduce directamente $A := 5$
- **Generador de código:** Se genera código objeto final aprovechando las prestaciones de una máquina concreta.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

COMPILADOR + INTERPRETE = JAVA



CARACTERÍSTICAS DE LOS LENGUAJES DE PROGRAMACIÓN

En la evolución de los lenguajes de programación algunos han triunfado debido a que aportaban algunas características que mejoraban a sus antecesores, veamos algunos de los más destacados.

Fortran: FORmula TRANslation, significa Sistema de Traducción de Fórmulas matemáticas, el más viejo de los lenguajes de “alto-nivel”. Diseñado en IBM durante los años 50, se hizo muy popular en los años 60. FORTRAN 66 fue el primer lenguaje en estandarizarse. Surgieron después otros como FORTRAN77 y FORTRAN90. Orientado a cálculos matemáticos. Las sucesivas versiones han ido paliando su principal inconveniente, poca capacidad de representación de estructuras de datos.

Cobol: Se trata de un lenguaje de 3ª generación, imperativo y orientado a la gestión. Aunque nace en 1960, la primera versión estandarizada es de 1968 (COBOL ANSI) con gran implantación en entornos empresariales, en 2002 se añadió OO y la última es de 2014. Actualmente tiene integración con Internet.

Pascal: Desarrollado a principios de los 70 por Niklaus Wirth. Con estructuras de datos y fuertemente tipado, todas las variables deben declararse previamente. Muy apto para la programación estructurada, especialmente en ambientes académicos. Evolucionó a Turbo Pascal, Objet Pascal y Delphi.

Smalltalk: Nace en 1970 y es un referente como lenguaje de POO, tiene tipado dinámico, una misma variable puede ser de distintos tipos. Un sistema Smalltalk está compuesto por: una máquina virtual, un archivo llamado "Imagen", que contiene a todos los objetos del sistema, un lenguaje de programación (**Smalltalk**), una enorme biblioteca de "objetos reusables" y un entorno de desarrollo. Todo en Smalltalk son objetos. No existen estructuras de control implementadas al margen de la jerarquía de clases.

CARACTERÍSTICAS DE LOS LENGUAJES DE PROGRAMACIÓN

C: Es un lenguaje de programación, desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell, como evolución del anterior lenguaje B. En 1983 se creó un comité que definió el estándar ANSI de C, llamado como C90. Este estándar incluye tanto la definición del lenguaje como una enorme biblioteca de funciones para entrada/salida, tratamiento de textos, matemáticas, etc. Posteriormente han surgido más versiones como "C99" y en 2011 la última C11. C sirvió como base para otros lenguajes como C++ y C#.

C está orientado a la programación de sistemas, sus características son:

- Proporciona las construcciones de control de flujo para programas estructurados. No obstante, su alta flexibilidad no induce a una programación disciplinada.
- Combina elementos de lenguajes de alto nivel con ensamblador.
- Incluye el concepto de puntero.
- Los argumentos de las funciones se transfieren por su valor.
- Las funciones pueden ser llamadas recursivamente.
- Es relativamente portable, se puede compilar para cualquier máquina.

```
using namespace std;
int main(int argc, char *argv[]) {
    std::cout << "Hola mundo" << endl;
    return 0;
}
```

CARACTERÍSTICAS DE LOS LENGUAJES DE PROGRAMACIÓN

C++: Fue diseñado en 1979, por Bjarne Stroustrup, como una extensión de C al que se le han añadido propiedades de OO, es multiparadigma. De este lenguaje surge C# en torno al año 2000 como una versión mejorada de C++ para la plataforma .NET de Microsoft. Las últimas versiones de C++ son de 2014 y 2017, y se espera una nueva para 2020.

- El código escrito en C es compatible en C++.
- Conserva la capacidad de C para manejar de forma eficiente los objetos fundamentales de la máquina.
- Soporta la creación de clases.
- Herencia (simple y múltiple)
- Polimorfismo (sobrecarga y ligadura dinámica).
- Genericidad.

CARACTERÍSTICAS DE LOS LENGUAJES DE PROGRAMACIÓN

Java: Desarrollado a principios de los 90 en Sun Microsystems con la finalidad de diseñar un lenguaje universal cuyos programas pudieran ejecutarse sobre cualquier arquitectura y cualquier sistema operativo. Ideal por tanto para programar por y para Internet. Para ejecutar los programas en Java se deben efectuar dos operaciones:

- 1.- Compilamos mediante javac.exe, por ejemplo, obteniendo los llamados bytecodes.
- 2.- Interpretamos/ejecutamos estos bytecodes desde cualquier plataforma con el intérprete JIT de los navegadores.

Sus principales características:

- Es OO, basado en la sintaxis de C++ pero a diferencia de este último, en Java no existen funciones libres, todo se hace a través de las clases, elimina el uso de punteros, e incorpora la recolección automática de memoria. No contempla la herencia múltiple, ni la sobrecarga de operadores.
- Es más sencillo de manejar que C++ para los programadores, en lo referente al tipo string.
- Pensado para trabajar con redes y protocolos TCP/IP, HTTP, FTP, etc.
- Es muy portable y ofrece múltiples aspectos de seguridad y multihilos.

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



CARACTERÍSTICAS DE LOS LENGUAJES DE PROGRAMACIÓN

PHP: Procesador de Hipertexto. Creado por Rasmus Lerdorf en 1995. Lenguaje diseñado específicamente para el trabajo con la web, cuyos programas se incluyen dentro de páginas html. Mientras que HTML ha sido diseñado para generar páginas estáticas, PHP lo ha sido para páginas dinámicas.

- Es multiplataforma. Buena integración con Mysql y otros servicios como ProFTPd, etc.
- PHP es server-side: Se interpreta en el servidor. Puede ser usado en la mayoría de los servidores web y sistemas operativos sin ningún costo.
- Permite técnicas Orientada a objetos.
- Es un lenguaje interpretado, no es necesario definir variables (tipado débil).
- Existen muchos frameworks basados en PHP, que permite trabajar con patrones MODELO-VISTA-CONTROLADOR (MVC).

```
<?php echo '<p>Hola Mundo</p>'; ?>
```



CARACTERÍSTICAS DE LOS LENGUAJES DE PROGRAMACIÓN

JavaScript: Es un lenguaje utilizado en programación web y los frameworks basados en él como Angular, React, etc.

- Se parece a Java, C, C++ por tanto fácil de aprender para los programadores que los usan.
- Lenguaje de scripting
- Se ejecuta en el lado cliente.
- Es seguro y fiable.
- Su código es visible y cualquiera puede leerlo.
- Hay muchas librerías basadas en él, como AngularJS, ReactJS, JQuery, Foundation JS, Backbone.js, etc.

```
<script type="text/javascript">  
    alert("Hola Mundo!");  
</script>
```

CARACTERÍSTICAS DE LOS LENGUAJES DE PROGRAMACIÓN

Python: Creado por Guido Van Rossum, debe su nombre a la afición de Guido a los Monty Python. Ha logrado posicionarse en pocos años dentro de los primeros lenguajes y sigue en alza. Se trata de un lenguaje que sirve para muchas de las tecnologías actuales: Bigdata, Realidad Virtual, ciberseguridad, robótica, etc. Entre sus características destacan:

- Permite una gran portabilidad, si se evita usar librerías particulares.
- Es un lenguaje interpretado, que utiliza un bytecode (similar a Java).
- Permite incluir código en C, e incrustarse en C y C++.
- Es un lenguaje orientado a objetos.
- Sus bibliotecas están organizadas en módulos, lo que permite que sea fácil de manejar y aprender.

```
# Hola mundo en Python
print ("¡Hola mundo!")
```

EJERCICIO 3



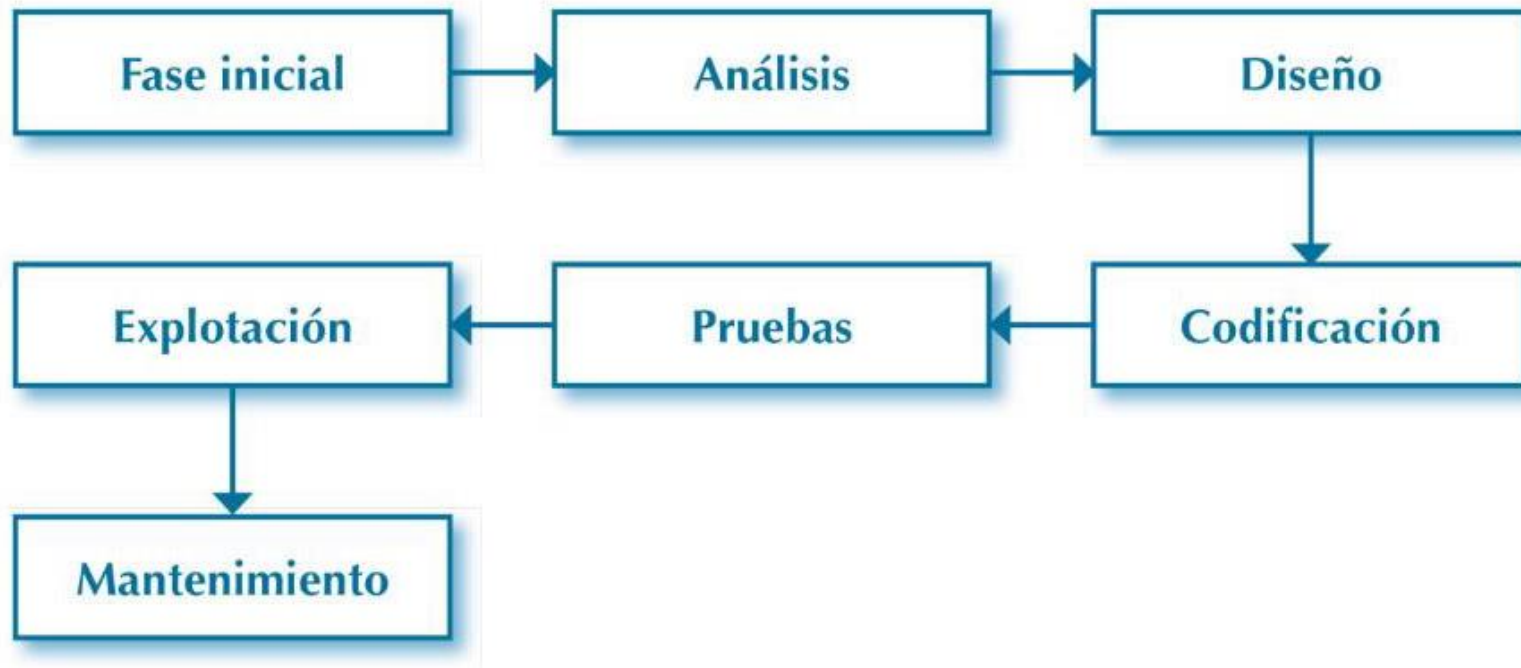
- Hemos estudiado las principales características de los lenguajes de programación más usados.

Escribe un documento y súbelo al aula virtual (AV)

- Con una tabla comparando sus características. Debes incluir otros 3 lenguajes más, sobre los que debes buscar sus características en Internet.
- Razona las ventajas/desventajas de los compiladores e intérpretes.

4. FASES DEL DESARROLLO DE UNA APLICACIÓN

Existen muchos paradigmas de desarrollo pero todos tienen en común las siguientes etapas, desde que se produce la necesidad de crear un software hasta que se finaliza y está listo para ser usado por un usuario, independientemente de su ciclo de vida. Las diferentes fases no siempre tienen un límite claro y en ocasiones se solapan.



FASE INICIAL

- En esta fase se planifica el proyecto, se hacen estimaciones, se evalúa la rentabilidad, etc.
- Se establecen las bases de como va a desarrollarse el resto de fases.
- Se requiere experiencia por lo complicado que supone planificar con datos económicos, soluciones a problemas y sus costes.
- Se realiza documentación que debe ser aceptada y acordada con la dirección de la empresa y los responsables del proyecto.

ANÁLISIS

- En esta fase se ANALIZA el problema y establece el producto a desarrollar
- Existe gran comunicación entre el cliente y el analista por conocer todas las necesidades que precisa la aplicación.
- Desarrollo de prototipos para saber con más precisión sus requerimientos.
- Es importante que haya una comunicación bilateral, aunque el cliente puede pretender que la comunicación sea unilateral, es necesario un contraste y consenso por ambas partes para llegar a definir los requisitos verdaderos del software.
- Informe ERS (Especificación de Requisitos Software)
- La documentación generada puede tener carácter contractual.

DISEÑO

- En esta fase se alcanza con mayor precisión una solución optima de la aplicación, teniendo en cuenta:
 - **Recursos físicos** (tipo de ordenador, periféricos, comunicaciones, etc...)
 - **Recursos lógicos** (sistema operativo, programas de utilidad, bases de datos, etc...)
- Se define por lo tanto el entorno que requerirá el sistema, aunque también se puede establecer en sentido contrario, es decir, diseñar el sistema en función de los recursos de los que se dispone.
- Se genera documentación técnica, que realizan los analistas con la supervisión del jefe de proyecto, como los diagramas de casos de uso y de secuencia para definir la funcionalidad del sistema.
- Se especificará también el formato de la información de entrada y salida, las estructuras de datos y la división modular. Con todo esto se obtiene el denominado **cuaderno de carga**.

CODIFICACIÓN E IMPLEMENTACIÓN

- Consiste en traducir los resultados obtenidos a un determinado lenguaje de programación, teniendo en cuenta las especificaciones obtenidas en el cuaderno de carga.
- Se genera la documentación detallada del código, parte incluida en el propio fuente, pero también en documentación indicando para cada función su parámetros de entrada y salida, propósito, módulos o librerías donde se encuentra, autor, fecha de creación y revisiones, etc.
- Se deben de realizar las pruebas necesarias para comprobar la calidad y estabilidad del programa.
- No se está exento de necesitar un reanálisis o rediseño al encontrar un problema al programar el software.
- La documentación debe facilitar el posterior mantenimiento del código.

PRUEBAS

- Comprobar la calidad y estabilidad del programa. Por una parte, la codificación tiene que ser exitosa y el software no debe contener errores, y por otra parte, el software hace lo que debe hacer.
- En general, las pruebas las realiza, personal diferente al que codificó la aplicación, con una amplia experiencia en programación, personas capaces de saber en qué condiciones un software puede fallar de antemano sin un análisis previo.
 - **Pruebas unitarias:** Sirven para comprobar que cada módulo realice bien su tarea.
 - **Pruebas de interconexión:** Sirven para comprobar en el programa el buen funcionamiento en conjunto de todos sus módulos.
 - **Pruebas de integración:** Sirven para comprobar el funcionamiento correcto del conjunto de programas que forman la aplicación. (el funcionamiento de todo el sistema).
- La documentación que se genera la podemos establecer en dos bloques:
 - **Funcionales**, con pruebas que valida junto con el cliente y se contrastan los requerimientos que se contrataron. Se anotan las posibles incidencias por posibles discrepancias.
 - **Estructurales**, se detallan los resultados de las pruebas más técnicas, con cargas de datos reales sometiendo a todo el sistema a estrés.

EXPLOTACIÓN

- Debe generarse la documentación como:
 - Documentación para el usuario
 - Documentación técnica, incluido el manual de instalación.
- En esta fase se realiza la implantación de la aplicación en el sistema o sistemas físicos donde van a funcionar habitualmente y su puesta en marcha para comprobar el buen funcionamiento.
- Actividades a tener en cuenta o realizar:
 - *Instalación del/los programa/s.*
 - *Pruebas de aceptación al nuevo sistema.*
 - *Conversión de la información del antiguo sistema al nuevo (si hay una aplicación antigua)*
 - *Eliminación del sistema anterior.*
- Al final de esta fase se debe de completar la información al usuario respecto al nuevo sistema y su uso.

MANTENIMIENTO

- Esta es la fase que completa el ciclo de vida y en ella nos encargaremos de solventar los posibles errores o deficiencias de la aplicación. Existe la posibilidad de que ciertas aplicaciones necesiten reiniciar el ciclo de vida.
- Es un proceso continuo que permite adaptar y evolucionar las aplicaciones.
- Se requiere utilizar la documentación que ya debe existir y es necesario ir actualizándola.
- **Mantenimiento correctivo:** Consiste en corregir errores no detectados en pruebas anteriores y que aparezcan con el uso normal de la aplicación.
- **Mantenimiento adaptativo:** Consiste en modificar el programa a causa de cambio de entorno gráfico y lógico en el que estén implantados.
- **Mantenimiento perfectivo:** Consiste en una mejora sustancial de la aplicación al recibir por parte de los usuarios propuestas sobre nuevas posibilidades y modificaciones de las existentes.
- Los tipos de mantenimiento adaptativo y perfectivo reinician el ciclo de vida, debiendo proceder de nuevo al desarrollo de cada una de sus fases para obtener un nuevo producto.

ROLES QUE INTERACTÚAN EN EL DESARROLLO

Arquitecto de software: Decide como se va a realizar el proyecto. Tiene un conocimiento profundo de las tecnologías, los frameworks, librerías, etc. Decide los recursos a utilizar.

Jefe de proyecto: Dirige el proyecto, puede ser un analista con experiencia, gestiona el equipo de trabajo y los plazos. Mantiene una relación fluida con el cliente.

Analista de sistemas: Su objetivo consiste en realizar un estudio del sistema para establecer los requerimientos para que se garantice las expectativas del cliente determinando el comportamiento del sistema.

Diseñador de software: Realiza, en función del análisis de un software, el diseño de la solución que hay que desarrollar.

Analista programador: Se suele llamar “desarrollador”, domina una visión más amplia de la programación, aportando una visión general del proyecto más detallada, diseñando una solución más amigable para la codificación y participando activamente en ella.

Programador: Se encarga de manera exclusiva de crear el resultado del estudio realizado por analistas y diseñadores. Escribe el código fuente del software.

EJERCICIO 4



Investiga sobre los siguientes ciclos de vida: modelo Evolutivo y modelo en Espiral.

- Fases
- Ventajas
- Inconvenientes

Escribe un documento y súbelo al aula virtual (AV)

5. EL PARADIGMA DE LA PROGRAMACIÓN OO

La diferencia fundamental respecto a la programación tradicional estructurada es que en lugar de aplicar sobre una serie de datos unas instrucciones, se abstrae el programa a una serie de objetos que intercambian mensajes entre sí. Analistas y programadores piensan y describen las cosas y el entorno en términos de objetos, atributos y métodos.

Sus principios fundamentales son:

Abstracción se centra en las características esenciales de un objeto, según la perspectiva del observador.

Encapsulación esconde los detalles de implementación de un objeto.

Modularidad es la propiedad de un sistema que se descompone en un conjunto de módulos acoplados y relacionados entre sí.

Jerarquía es una ordenación de abstracciones.

Clasificación es asignar de forma precisa un objeto a una clase de objetos.

Concurrencia permite que diferentes objetos actúen al mismo tiempo.

Persistencia es la propiedad de un objeto a través de la cuál su existencia trasciende tiempo y/o espacio.

OBJETO

Definición conceptual

Un objeto es una entidad que contiene los atributos (datos) que caracterizan su estado y los métodos (acciones) que pueden modificar ese estado. Un objeto es una entidad que posee:

Estado: contiene la información que puede almacenar el objeto. Puede variar con el tiempo.

Comportamiento: conjunto de acciones que puede realizar.

Identidad: propiedad que hace un objeto distinguible del resto.

Alternativamente se define como una entidad con dos componentes:

Parte estática: atributos (datos).

Parte dinámica: procedimientos o funciones que manipulan esos datos (acciones).

Un objeto no funciona de forma independiente en un programa, sino que tiene que comunicarse con el resto de los objetos que componen el sistema, para ello se usan los mensajes.