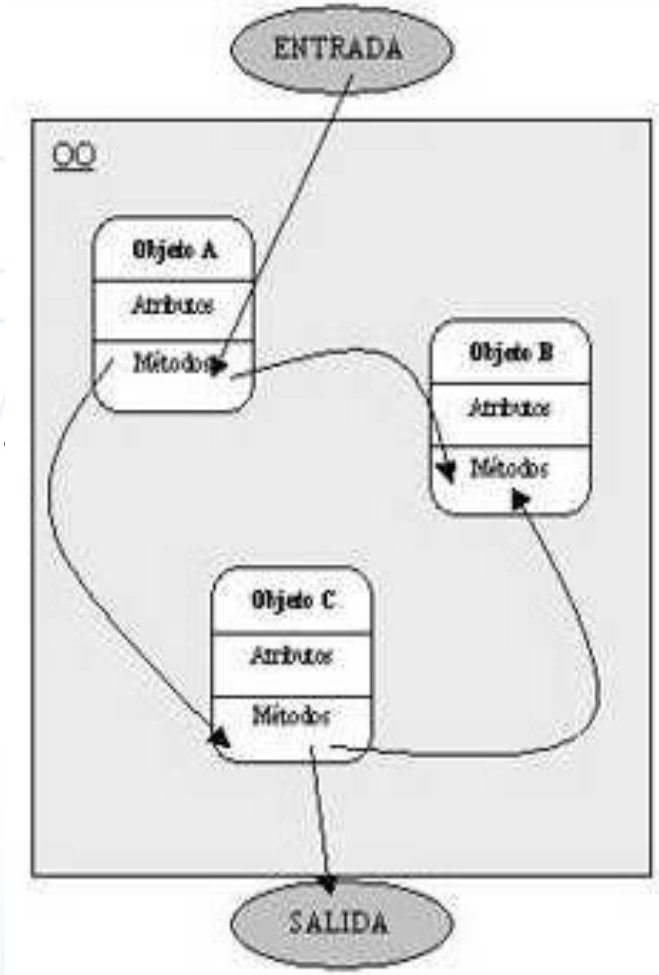
PROGRAMACIÓN II

1. INTRODUCCIÓN A LA PROGRAMACIÓN VISUAL Y DE EVENTOS

1.1. Paradigmas de programación.
1.2. Paradigma orientado a eventos: definición, estructura, características y tipos de eventos.
- 1.3. Paradigma de la Programación Visual.
- 1.4. Lenguajes de programación visual y orientada a eventos.

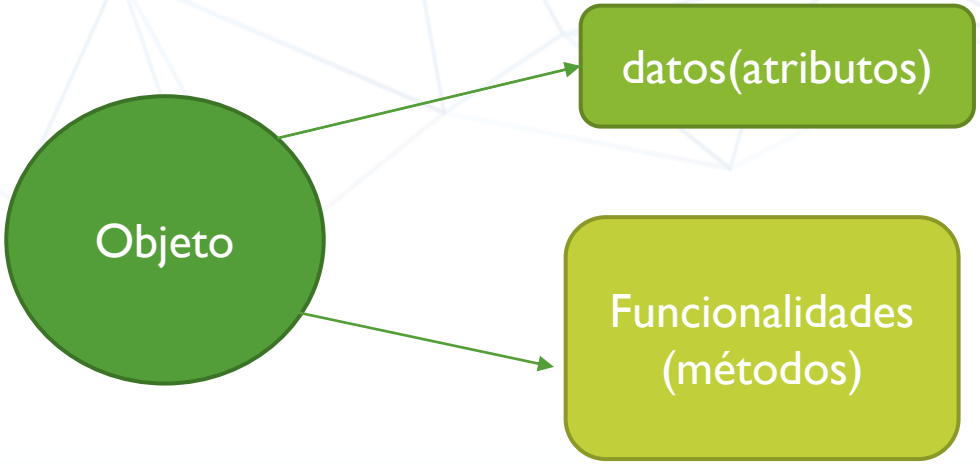
De dónde venimos……

- El paradigma de programación más utilizado en el mundo es el **Orientado a Objetos**, porque cada uno de los elementos que se necesitan representar en un sistema, por ejemplo: los productos, clientes, vendedores, etc. pueden ser representados como un objeto. Y estos objetos tienen sus propios datos y sus propias funcionalidades.
- Ejemplos:
  - Un producto tiene nombre, precio, puede ser comprado
  - Un cliente tiene nombre, ruc, puede comprar producto
  - Un carrito de compras tiene productos, cliente puede ser enviado a pagar.



Programación Orientada a Objetos

- Cada elemento en los que vamos dividiendo el sistema es un objeto, y ***los objetos tienen datos y funcionalidad.***
- Los datos al llevarlo a POO se denominan **atributos** y las funcionalidades se denominan **métodos**.



1.1. Paradigmas de programación.

"Un paradigma de programación indica un método de realizar cómputos y la manera en que se deben estructurar y organizar las tareas que debe llevar a cabo un programa"

- Los paradigmas fundamentales están basados en diferentes modelos de cómputo y por lo tanto afectan a las construcciones más básicas de un programa .
- Se asocian a un determinado estilo de programación
- Los lenguajes de programación suelen implementar, de manera parcial, a menudo, varios paradigmas.

Tipos De Paradigmas De Programación

- Dos grandes grupos:
  - Imperativos (programación procedural, modular y estructurada.).
  - Declarativos (programación lógica y funcional)
- Orientada Objetos (el más utilizado)
- Orientada a Eventos (Dirigida por eventos)
- Programación Reactiva

- Un paradigma de programación está sujeto en el tiempo por el uso y la aceptación, por el surgimiento de nuevos paradigmas que aportan nuevas y mejores soluciones.

Paradigmas Imperativo

- Describe cómo debe realizarse, no el porqué.
- Un cómputo consiste en una serie de sentencias, ejecutadas según un control de flujo explícito, que modifican el estado del programa.
- La sentencia principal es la asignación. a=objeto.metodo();
- Las variables pueden ser modificadas, y representan el estado del programa.
- Asociados al paradigma imperativo se encuentran los paradigmas procedural, modular, y la programación estructurada.
- Los lenguajes representativos sería FORTRAN-77, junto con COBOL, BASIC, PASCAL, C, ADA.
- También lo implementan Java, C++, C#, Eiffel, Python, ..

Paradigmas Declarativo

- Describe que se debe hacer, sin explicitar el cómo.
- No existe un orden de evaluación prefijado.
- No existe sentencia de asignación.
- Las variables son nombres asociados a definiciones, y una vez instanciadas son inmutables.
- El control de flujo suele estar asociado a la composición funcional, la recursividad y/o técnicas de reescritura y unificación.
- Las principales variantes son los paradigmas funcional, lógico, la programación reactiva y los lenguajes descriptivos (html,sql).
- Lenguajes de ejemplos:
- Programación Lógica: **Prolog**
- Programación funcional: ML, **Lisp, Scala, Java, Kotlin**

Diferencia entre imperativo y declarativo

- En la programación imperativa se describe paso a paso un conjunto de instrucciones, es decir, un algoritmo en el que se describen los pasos necesarios para solucionar el problema.
- En la programación declarativa las sentencias que se utilizan lo que hacen es describir el problema que se quiere solucionar; se programa diciendo lo que se quiere resolver a nivel de usuario, pero no las instrucciones necesarias para solucionarlo.

Fuente: https://www.docsity.com/es/paradigmas-de-programacion-2/7114608/

Programación Funcional

- Basado en los modelos de cómputo cálculo lambda (Lisp, Scheme) y lógica combinatoria (familia ML, Haskell).
- Con este paradigma las funciones serán tratadas como ciudadanos de primera clase. Las funciones podrán ser asignadas a variables además podrán ser utilizadas como entrada y salida de otras funciones.
- A las funciones que puedan tomar funciones como parámetros y devolver funciones como resultado serán conocidas como función de orden superior.
- Los lenguajes funcionales priorizan el uso de recursividad y aplicación de funciones de orden superior para resolver problemas que en otros lenguajes se resolverían mediante estructuras de control (por ejemplo, ciclos).
- Entre los lenguajes de programación funcional más importantes se encuentran los siguientes: LISP, ML, Haskell, Scala, R

Programación Lógica

- Basado en la lógica de predicados de primer orden
- Los programas se componen de hechos, predicados y relaciones.
- La ejecución consiste en la resolución de un problema de decisión, los resultados se obtienen mediante la instanciación de las variables libres.
- La mayoría de los lenguajes de programación lógica se basan en la teoría lógica de primer orden, aunque también incorporan algunos comportamientos de orden superior como la lógica difusa.
- Lenguaje representativo: PROLOG, F-Prolog

Paradigma Orientado a Objetos

- Se construyen modelos de objetos que representan elementos (objetos) del problema a resolver, que tienen características y funciones.
- La programación orientada a objetos disminuye los errores y promociona la reutilización del código.
- Se basa en objetos que se crean a partir de clases.
- Las principales características son: Abstracción, Encapsulamiento, Herencia y Polimorfismo.
- El modo de organización del programa: En clases (datos + operaciones sobre datos).
- El concepto de ejecución de programa: Paso de mensajes
- Ejemplos de lenguajes de programación orientados a objetos serían Java, Python, C#.

Paradigma Orientado A Eventos

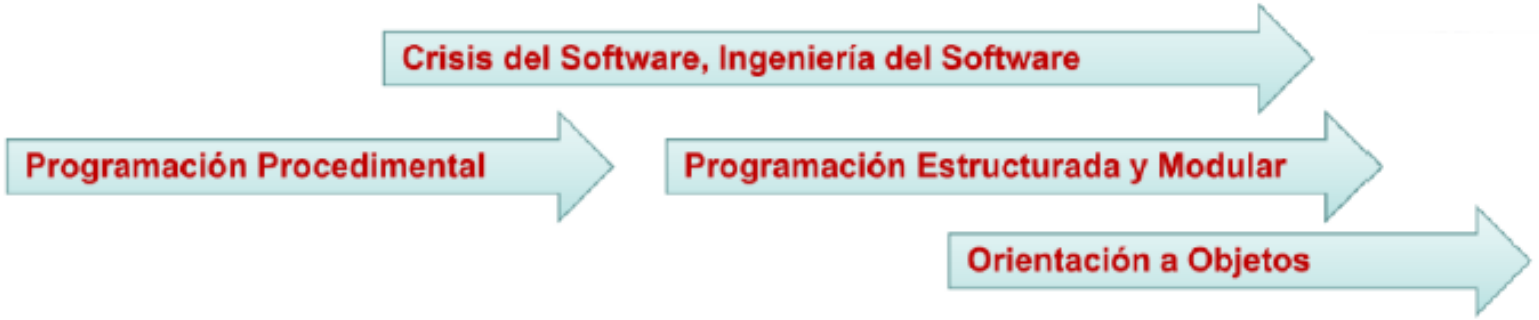
- El flujo del programa es guiado por los eventos que suceden sobre su interfaz gráfico, señales de sensores o mensajes de otros programas o hilos. [Adaptado de Wikipedia]
- Los programas atienden a sucesos (eventos) que ocurren, y dependiendo de cuales sean, se ejecutan diferentes funciones.
- Ejemplo: aplicación con una interface gráfica -GUI.
  - No existe un único flujo de ejecución.
  - Se especifican los eventos a los que debe responder el programa.
  - Se especifican las acciones a realizar cuando ocurren los eventos.
  - Funcionamiento asíncrono.

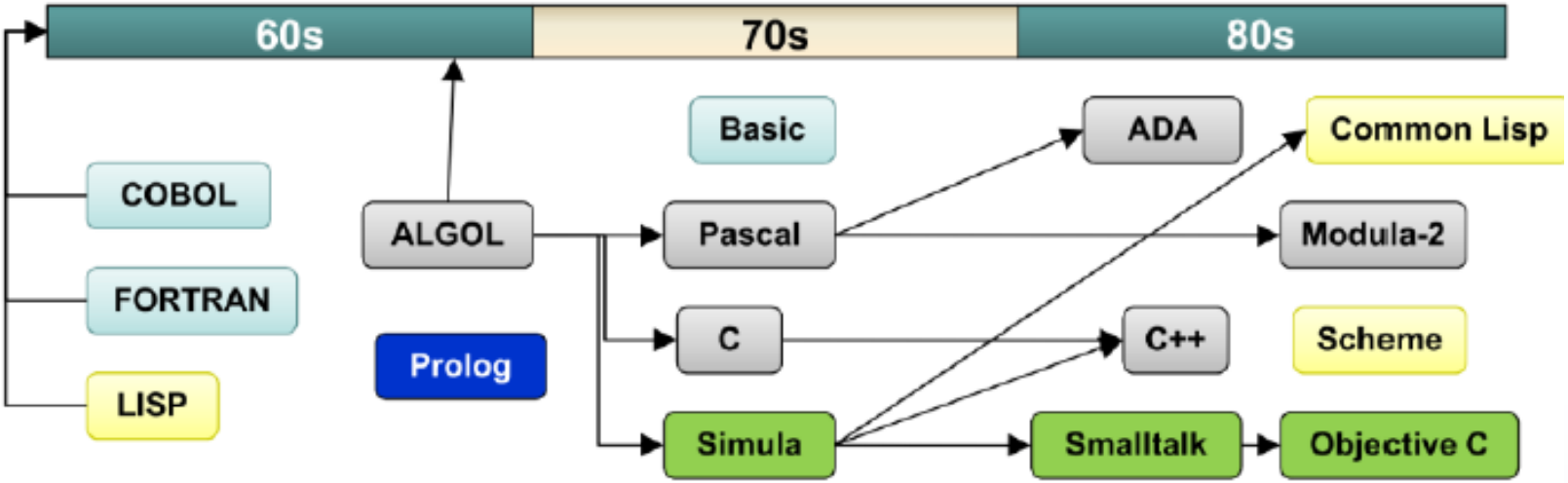
Programación reactiva

- Este paradigma se basa en escuchar lo que emite un evento o cambios en el flujo de datos, en donde los objetos reaccionan a los valores que reciben de dicho cambio.
- Este tipo de programación se mantiene observando cambios en un flujo de datos. Ejemplo una transmisión de video, un chat. Cuando los datos cambian hay una reacción. Se reaccionar a los cambios en los flujos de datos. Ejemplo si baja la velocidad de conexión durante una transmisión de video se podría reaccionar enviando una calidad de imagen menor.
- Las librerías más conocidas son Project Reactor, y RxJava.
- React/Angular usan RxJs para hacer uso de la programación reactiva.

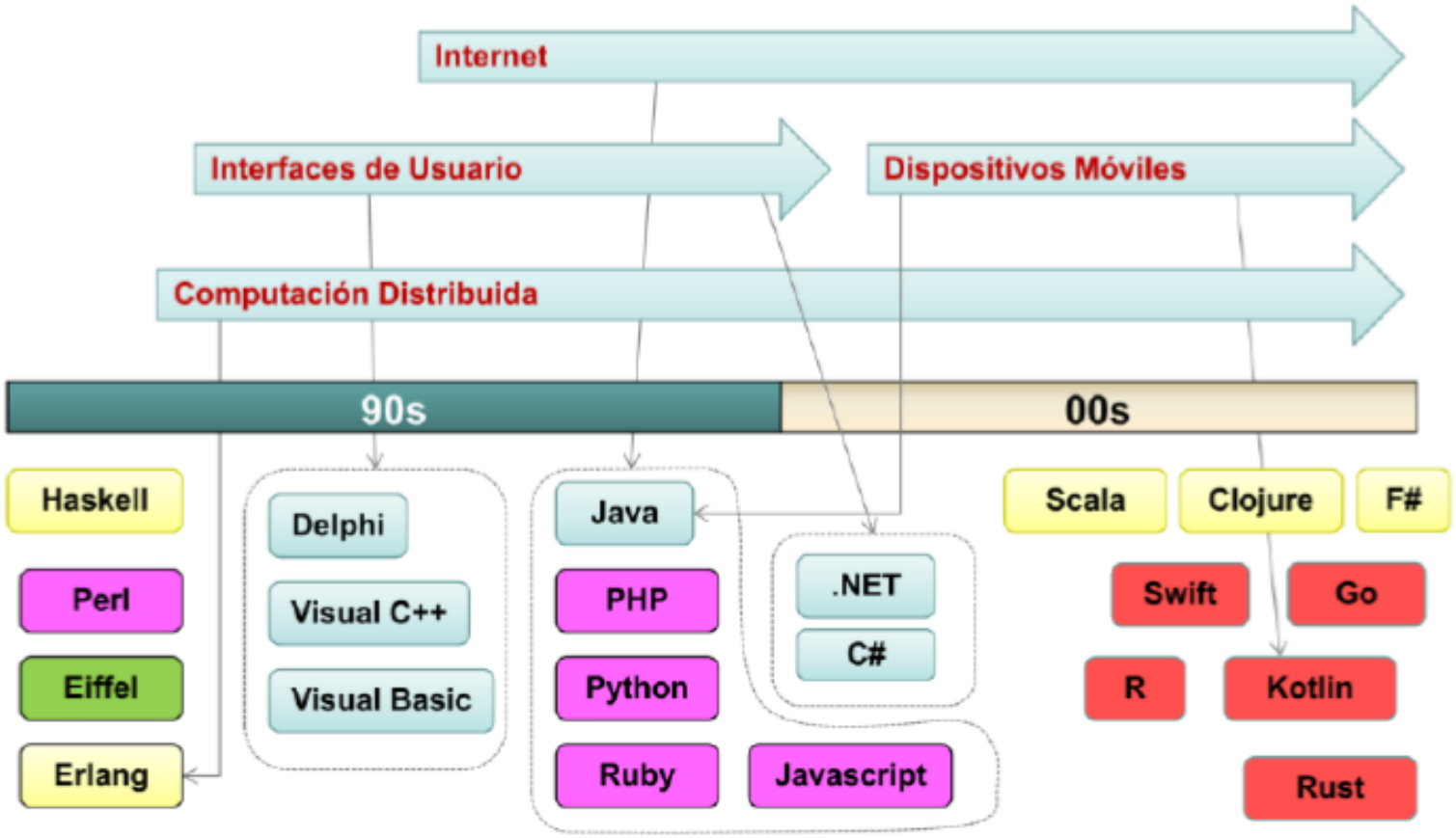
Fuente: https://profile.es/blog/que-son-los-paradigmas-de-programacion/

Linea del Tiempo





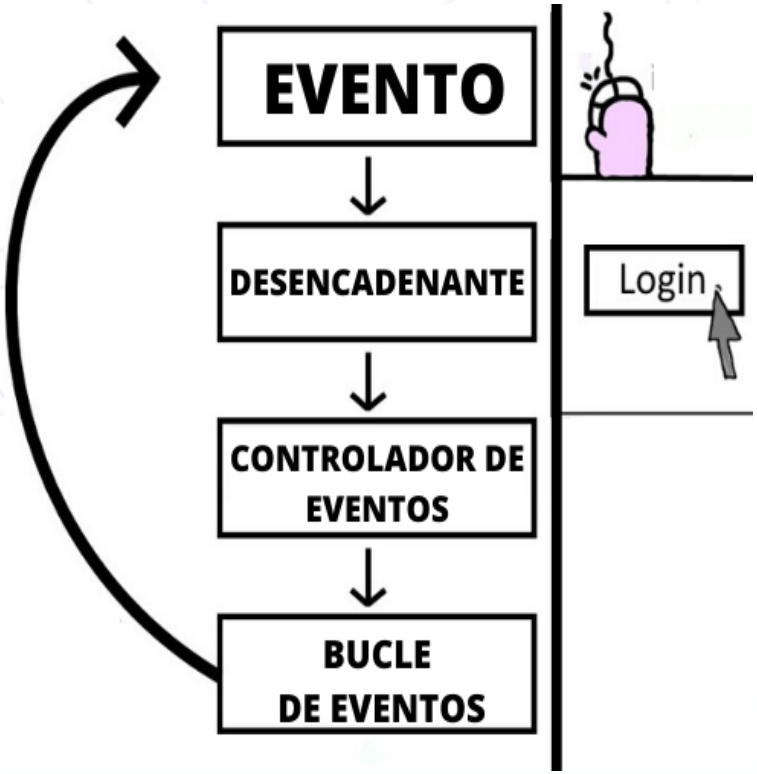
Linea del Tiempo



Programación Orientada a Eventos

- Es un paradigma de programación en el que la estructura y la ejecución de los programas van determinados por los sucesos o acciones que ocurren en el sistema.

- En la POE la línea de ejecución del programa no está determinada de antemano, no se conoce cuál de las líneas de código se ejecutarán, no es secuencial ya que depende de un evento.



Fuente: https://www.lifeder.com/programacion-orientada-a-eventos/

Programación Orientada a Eventos

- Un evento es una acción que es reconocida por un objeto y que normalmente es provocada por un usuario al interactuar con la interfaz gráfica del programa (clic en un botón, pulsación de una tecla, etc).

- Algunos objetos tienen predefinidos un conjunto de eventos, por lo tanto, una aplicación ejecuta funciones para tratar los eventos. De ahí la analogía con la programación orientada a objetos.

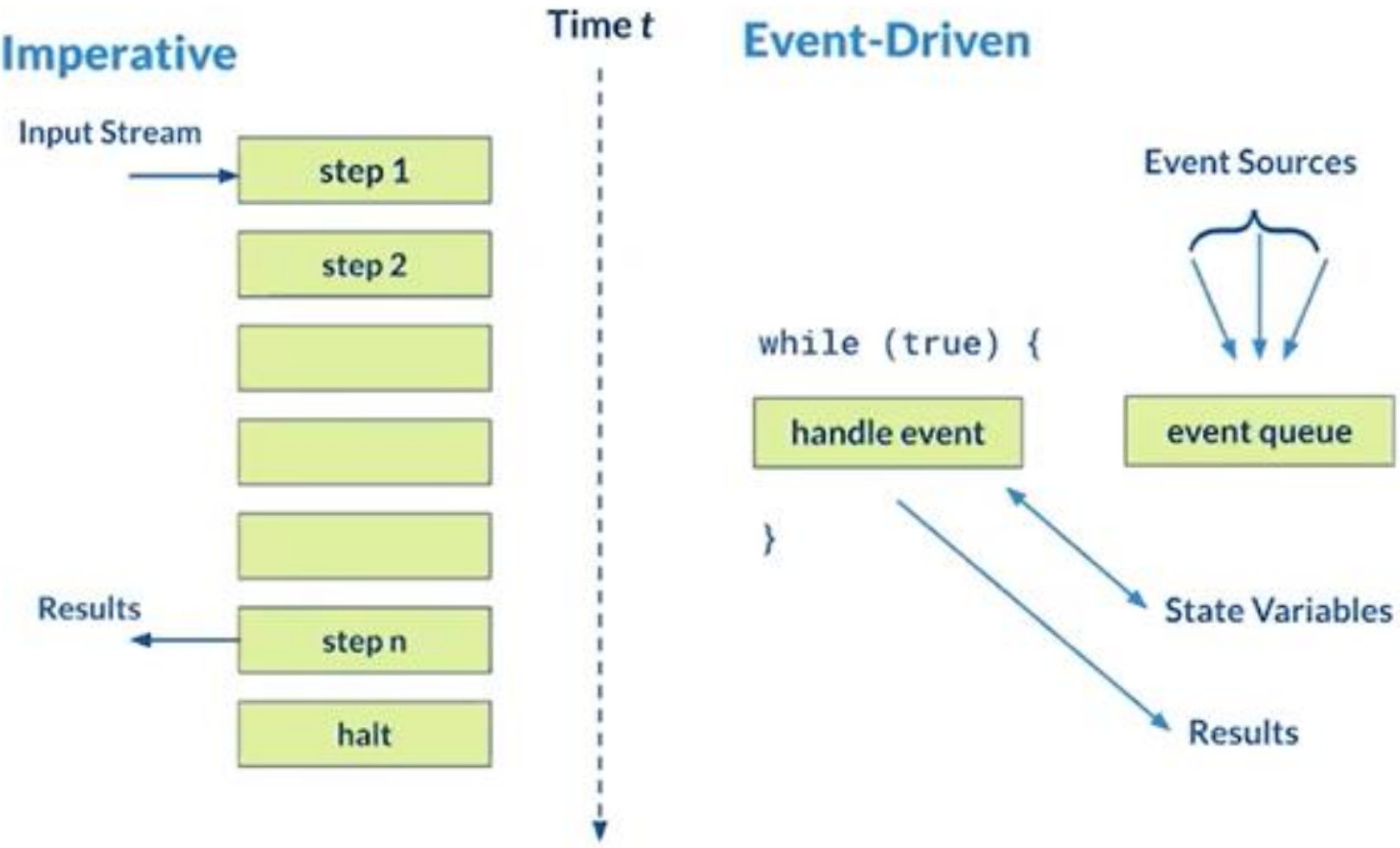
Programación Orientada a Eventos – Tipos de Eventos

- Externos: Producidos por el usuario (generalmente a través de los periféricos).
- Ejemplos:
  - pulsaciones de teclado o ratón.
- Internos: Producidos por el sistema o la aplicación.
- Ejemplos:
  - vencimiento de un temporizador.
  - datos en líneas de comunicaciones.
  - sensores

Programa dirigido por eventos

- El flujo de control del programa depende de la ocurrencia de eventos externos o internos.
- El típico programa dirigido por eventos permanece en un estado en el que escucha para o espera eventos, selecciona a qué eventos responder a continuación, responde a ellos y luego regresa a su estado de escucha.
- A diferencia de los programas secuenciales, los programas controlados por eventos deben funcionar correctamente en un entorno en el que los estímulos externos, dinámicos e inesperados provienen de fuentes como los usuarios, hardware u otros procesos.
- Generalmente los programas corren indefinidamente integrando manejadores de eventos (event handlers)

Cómo Funciona



Eventos y event handler

- Los programas dirigidos a eventos se mantienen en estado de reposo, hasta que una acción dispara un **event handler**. Este se encarga de procesar el evento en cuestión. Algunos eventos comunes son:
  - Clics, presionar teclas (ya sea en un mouse, teclado, pantalla, etc.)
  - Sensores (de temperatura, movimiento, etc.)
  - Mensajes
  - Triggers (disparadores, en sql)
  - Solicitudes HTTP (web)

Event handler: un ejemplo en Java.

- Añadimos un event listener con addMouseListener
- Ejemplo en Java
- Se utiliza las interfaces de escucha de eventos (ActionListener, MouseListener, WindowListener, KeyListener, etc.)

```
public class MyApplication extends
JPanel implements MouseListener{

  public MyApplication() {
     ...
     addMouseListener(this)
     ...
  }

}
```

Event handler: un ejemplo en JavaScript.

Aquí vemos un ejemplo en JavaScript. Nota el uso de **addEventListener**

```
<!-- HTML -->

<button onclick="myScript">Click here</button>

// JAVASCRIPT

object.addEventListener("click", myScript);
```

Programación Visual

- En la programación visual, los elementos del lenguaje de programación están disponibles en forma de bloques diseñados de manera gráfica, por lo que también se la llama programación gráfica. Se caracteriza por:
  - Visual: los elementos se arrastran y sueltan en el flujo del programa para integrarlos.
  - Orientado a acontecimientos: cada paso del programa comienza cuando ocurre un acontecimiento previamente definido.
  - Imperativo: la programación consiste en una secuencia de comandos.
  - Orientado a objetos: hay objetos individuales con tareas definidas asignadas.

Programación Visual y Orientada a Eventos

- Los programas basados en eventos incluyen programas con interfaces gráficas de usuario, sistemas operativos, dispositivos controladores, software de sistema de control y videojuegos, por nombrar algunos.

- La programación orientada a eventos es un paradigma mientras que la programación visual presenta elementos o herramientas visuales que permiten arrastrar y soltar, generándose muchas veces código automático, y a partir de estos controles u objetos se agregan funciones o métodos para programar los eventos a los cuales están relacionados.

Detección de eventos

- En contraposición al modelo clásico, la programación orientada a eventos permite interactuar con el usuario en cualquier momento de la ejecución. Esto se consigue debido a que los programas creados bajo esta arquitectura se componen por un bucle exterior permanente encargado de recoger los eventos, y distintos procesos que se encargan de tratarlos. Habitualmente, este bucle externo permanece oculto al programador que simplemente se encarga de tratar los eventos, aunque en algunos entornos de desarrollo (IDE) será necesaria su construcción.

- Ejemplo de programa orientado a eventos en pseudo lenguaje:

```
While (true){
    Switch (event){
        case mouse_button_down:
        case mouse_click:
        case keypressed:
        case Else:
    }
}
```

Desventajas

- La programación orientada a eventos supone una complicación añadida con respecto a otros paradigmas de programación, debido a que el flujo de ejecución del software escapa al control del programador.
- En cierta manera podríamos decir que en la programación clásica el flujo estaba en poder del programador y era este quien decidía el orden de ejecución de los procesos, mientras que en programación orientada a eventos, es el usuario el que controla el flujo y decide.

- Pongamos como ejemplo de la problemática existente, un menú con dos botones, botón 1 y botón 2. Cuando el usuario pulsa botón 1, el programa se encarga de recoger ciertos parámetros que están almacenados en un archivo y calcular algunas variables. Cuando el usuario pulsa el botón 2, se le muestran al usuario por pantalla dichas variables. Es sencillo darse cuenta de que la naturaleza indeterminada de las acciones del usuario y las características de este paradigma pueden fácilmente desembocar en el error fatal de que se pulse el botón 2 sin previamente haber sido pulsado el botón 1. Aunque esto no pasa si se tienen en cuenta las propiedades de dichos botones, haciendo inaccesible la pulsación sobre el botón 2 hasta que previamente se haya pulsado el botón 1.

Lenguajes

- Algunos lenguajes que implementan este paradigma son:

- Java
- JavaScript
- C#
- Librerías/Frameworks de GUI
- JavaFX, React.js, PyQT
- Sobretodo, cualquier lenguaje orientado a objetos.

Tipos de códigos comunes para manejo de eventos (event handlers)

- La programación orientada a eventos es un subtipo de programación paralela y, en consecuencia, es rápida, eficiente y concurrente. El código del programa ejecutado por el evento se ejecuta de forma asíncrona con el programa principal; esto sucede a menudo como un hilo o una tarea.
- Los tipos de códigos utilizados para control de eventos incluyen:
  - Call back (Ejemplo javascript)
  - Event handler (Ejemplo c#)
  - Listener (Ejemplo java)
  - Event listener (Ejemplo javascript)
  - Exit routine (Ejemplo c, visual basic)
  - Event notification function (servicios, java c#)