Tema 1 Introducción a Dispositivos Móviles

1.1 Introducción

Los dispositivos de comunicación móvil tienen unos 20 años. La tecnología de comunicación ha evolucionado gracias al uso de tecnologías inalámbricas. Desde el uso para comunicación de voz hasta el uso multiplataforma actual.

Cronología:

- **1860:** James Clerk formulo las ecuaciones de propagación de ondas electromagnéticas.

Gracias a los avances de la tecnología, el Mark Weiser en 1991 formulo el concepto de computación ubicua, que pone al alcance de las personas el uso de la tecnología en la vida cotidiana. Para lograr este objetivo, se necesitan cumplir 4 objetivos:

1. **Descentralización:** se pasa de un modelo centralizado en grandes equipos a un modelo ligero tipo cliente/servidor.
2. **Diversificación:** adaptar los servicios para diferentes dispositivos con funcionalidades diferentes. Gracias a esta diversificación se crearon protocolos nuevos de comunicación: HTTP, FTP, SMTP, etc. Y además se crearon nuevos entornos de comunicación: Sistemas remotos, plataformas virtuales.
3. **Conectividad:** la creación de un conjunto de estándares de comunicación que permiten la conexión de diferentes dispositivos mediante conectividades adaptadas para cada dispositivo. Este es el punto de partida para el desarrollo de estándares como:
    - **WAP** (Wireless Access Protocol): Protocolo de acceso inalámbrico.
    - **UMTS** (Universal Mobile Telecommunications System): Sistema universal de comunicación móvil.
    - **Bluetooth:** Protocolo de comunicación inalámbrico mediante radiofrecuencia.

Simplicidad: se centra en el diseño de dispositivos con interfaces de usuario intuitivas. Apuesta por la integración de software y hardware para lograr mejor cobertura de las necesidades al usuario final. Haciendo el acceso fácil y rápido.

Common Language Specification (CLS) Especificación del Lenguaje Común es un conjunto de reglas que han de seguir las definiciones de tipos que se hagan usando un determinado lenguaje gestionado si se desea que sean accesibles desde cualquier otro lenguaje gestionado
Define un conjunto de características en las que se deben basar los lenguajes de programación para determinar si el tipo de dato cumple con la especificación. Si las reglas se cumplen, se marca como *CLS-compliant.* Esto significa que todos los lenguajes .NET son capaces de interactuar con él.

- **Finalidades :**
  Independencia del lenguaje .
  Integración entre Lenguajes.
  Apertura a nuevos lenguajes.

SDK -> kit de desarrollo de software.
IIS -> (internet information server) Servidor web de Microsoft que se ejecuta sobre plataforma Windows.

MSIL -> Lenguaje intermedio de Microsoft

Que es .net?

Plataforma de desarrollo que se compone de:

– Entorno de ejecución (Runtime)
– Biblioteca de clases (Class Library)
– Lenguajes de Programación.
– Compiladores
– Herramientas de desarrollo (IDE y Herramientas)

¿Qué no es .NET?

- **No es un Sistema Operativo.**
- **No es un Leng.Programación**
- **No es un entorno de desarrollo**
- **No es un servidor de aplicaciones**
- **No es un paquete de software**

Características .NET

- **Plataforma de ejecución intermedia.**
- **Es orientada a objetos (lenguajes, class library, entorno)**
- **Multilenguaje.**

Características de C#

- **Sencillez**
- **Novedad**
- **Orientación a objetos**
- **Orientación a componentes**
- **Gestión automática de memoria**
- **Seguridad de tipos**
- **Sistema de tipos unificado**
- **Extensibilidad de operadores**
- **Gestión dinámica de memoria**

Identificadores

- **Sólo se pueden utilizar: letras mayúsculas, minúsculas, dígitos y carácter _.**
- **No puede comenzar por dígito.**
- **No puede contener blancos.**
- **No puede ser igual a una palabra reservada.**
- **Sensible a mayúsculas y minúsculas**

Tipo de datos primitivos

Int, long, float, doublé, decimal, string, char y bool

Instrucciones

Instrucción: es un comando que realiza una acción. Las sentencias se encuentras dentro de métodos. Un programa está formado por un método llamado Main.

- ***Instrucción nula:*** es una instrucción que no realiza nada en absoluto. Su sintaxis consiste en escribir un simple punto y coma para representarla.
- **Instrucción foreach**

foreach (<tipoElemento> <elemento> in <colección>)
<instrucciones>

- tipoElemento: Indica el tipo de datos de los elementos de la colección.
- elemento: Nombre de variable de tipoElemento que almacena en cada iteración el valor del elemento actual de la colección.
- colección: objeto capaz de almacenar una colección de elementos (array, listas, etc)

4.1 Operadores sobre tipos

<expresión> is <nombreTipo>

Se evalúa la expresión. Si el resultado de la evaluación corresponde al tipo indicado, devuelve true. En caso contrario devuelve false.

sizeof(<nombreTipo>)

Solo puede usarse en código no seguro (No plataformado C++) y solo se puede aplicar sobre nombres de tipos cuyos objetos se almacenan en pila.

4.2 Operadores de comprensión

- **(<tipoDestino>) <expresion>**

  Convierte la expresión al tipo especificado(Cast).

- **<expresion> as <tipoDestino>**
  Devuelve el resultado de convertir la expresión al tipo indicado como destino. Las diferencias entre as y **()** ←operador de molde.
  - as solo se aplica a tipos de referencia (tipos no primitivos) y que tengan conversión explicita.
  - Los programadores pueden sobrecargar el operador (), pero no podrán sobrecargar el as.

Metodo Conjunto de instrucciones a las que se les da un determinado nombre de tal manera que sea posible ejecutarlas en cualquier momento sin tenerlas que reescribir sino usando sólo su nombre.
- Cuerpo: Contiene las instrucciones que se ejecutan cuando se realiza la llamada al método.
- Nombre: Identificador para la nomenclatura del método.

Llamadas a metodos

<objeto>.<nombreMétodo>(<listaDeArgumentos>)
- objeto: indica la variable de tipo clase en la que se encuentra el método al que se quiere acceder.
- nombreMétodo: identificador del método que se quiere acceder.
- listaDeArgumentos: Lista de parámetros necesarios para el método.

Los Tipos de parámetros en los metodos pueden ser:

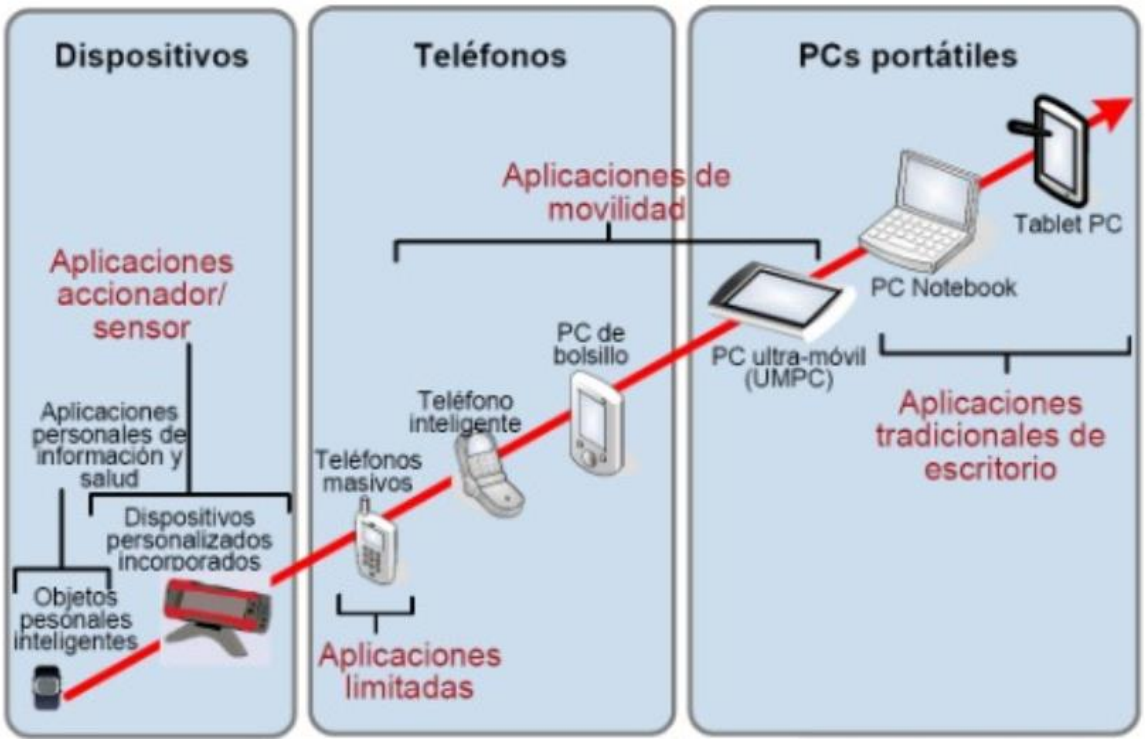
De entrada: Se copia un valor del parámetro y su valor no se ve afectado en el método.
De entrada/salida (Por referencia): Su valor se puede modificar dentro del método. Se emplea la palabra reservada ref.
De salida (Por referencia): Es obligatorio modificar el valor del parámetro de salida, además el valor anterior del parámetro no puede ser utilizado dentro del método. Se emplea la palabra reservada out (Método y llamada)

POO en C#

La programación orientada a objetos permite cambiar el paradigma de programación.
- Cambia la filosofía respecto a la programación tradicional en la que el foco de desarrollo era la función.

Control de acceso

- **Public**: Accesible por exterior e interior de la clase.
- **Private**: Accesible por interior de la clase.
- **Protected:** Accesible por interior de la clase y los hijos.


Dispositivos
Teléfonos
PCs portátiles
Aplicaciones de movilidad
Aplicaciones accionador/ sensor
Aplicaciones personales de información y salud
Dispositivos personalizados incorporados
Objetos personales inteligentes
Teléfonos masivos
Teléfono inteligente
PC de bolsillo
PC ultra-móvil (UMPC)
PC Notebook
Tablet PC
Aplicaciones tradicionales de escritorio
Aplicaciones limitadas