JAVA- UNIDAD 1
CONCEPTOS BÁSICOS

En esta unidad abordamos la definición de **Java** como lenguaje de programación de propósito general, basado en el paradigma orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuese posible.

Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como **WORA**, o "write once, run anywhere"), lo que quiere decir que el código que se ejecuta en una plataforma no tiene que ser recompilado para correr en otra.

El lenguaje de programación Java fue originalmente desarrollado por ***James Gosling*** **de Sun Microsystems**, la cual fue adquirida por la compañía Oracle y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems.

Las aplicaciones de Java son generalmente compiladas a ***Bytecode*** (clase Java) que se ejecuta en una ***Máquina Virtual Java*** **(JVM)** que es una máquina de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial *bytecode*.

La JVM es una de las piezas fundamentales de la plataforma Java. Básicamente se sitúa en un nivel superior al hardware del sistema sobre el que se pretende ejecutar la aplicación, y este actúa como un puente que entiende tanto el *bytecode* como el sistema sobre el que se pretende ejecutar. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una máquina virtual Java en concreto, siendo ésta la que en última instancia convierte de código *bytecode* a código nativo del dispositivo final.

JAVA- UNIDAD 2
CONTROL DE FLUJO DE PROGRAMAS

En esta unidad conocimos las **Entradas y Salidas** por consola que nos permiten manejar el flujo de datos que necesitamos tanto salidas como entradas, para ello manejamos la comunicación por consola entre el usuario y el programa. De esta manera podemos leer datos y mantenerlos almacenados dentro de nuestro programa.

Además estudiamos los **Operadores** que son símbolos que tienen una función predefinida (suma, resta, multiplicación, entre otros) y que reciben argumentos de manera fija. Estos operadores se utilizan en la elaboración de una ecuación, permiten combinar diferentes términos entre sí y establecen una lógica entre los términos de una expresión.

Por otra parte tratamos las estructuras **Condicionales** cuya función es comparar una variable con otra, con el fin de que el resultado de esta comparación decida un curso de acción dentro del programa. Es muy importante destacar que la comparación se puede hacer con otra variable o con una constante.

También revisamos los **Ciclos Repetitivos o Bucles**, que permiten repetir una operación o secuencia de operaciones en función de ciertas condiciones. Son segmentos de un algoritmo cuyas instrucciones se repiten un número determinado de veces, mientras se cumplan unas condiciones específicas. Dentro de los ciclos podrían usarse contadores o acumuladores que regulan y se aseguran de que el ciclo llegue a su fin.

Y finalmente definimos los aspectos básicos acerca del **Manejo de Excepciones** que se refiere a un evento que ocurre durante la ejecución del programa e interrumpe el flujo natural de las sentencias, es decir, es algo que altera la ejecución normal.

Muchas clases de errores pueden generar excepciones desde formatos de un valor hasta errores en la sintaxis de la programación, por ejemplo, al acceder a un elemento de un arreglo. Con el manejo de excepciones es posible diferenciar los tipos de errores, y dependiendo de esto ejecutaremos sentencias, de esta manera hacemos todas las excepciones juntas y podremos pensar en la manera más adecuada para tratar los errores.

JAVA- UNIDAD 3
PROGRAMACIÓN ORIENTADA A OBJETOS

En esta unidad tratamos la **Programación Orientada a Objetos** que se refiere a la contextualización del dominio del problema en términos de los objetos que interactúan, que se modifican y que responden a las acciones, construyendo un modelo que simula el comportamiento del mundo real. Utiliza objetos en sus interacciones para diseñar aplicaciones y programas informáticos. Este paradigma está basado en varias técnicas, incluyendo la abstracción, la herencia y el polimorfismo.

La **Abstracción** hace referencia al proceso de aislar un elemento de su contexto o del resto de los elementos que lo acompañan. Es decir, tratar únicamente los conceptos del dominio del problema, y no tomar decisiones con respecto al diseño o la implementación. La abstracción es el proceso de centrarse en lo que es y lo que se hace, y no en cómo se hace.

También conocimos **UML "Unified Modeling Language"** que es el lenguaje de modelado de sistemas de software. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. El UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

En cuanto a la **Herencia** es el mecanismo empleado para obtener atributos y operaciones utilizando la estructura de generalización. En ella se comparten atributos y operaciones entre clases, tomando como base una relación jerárquica. Y el **Polimorfismo** hace referencia a la capacidad de un método para tener muchas formas, aunque sea sintácticamente igual.

También estudiamos **Métodos Especiales** que se utilizan para controlar los cambios en una variable, este método es ampliamente conocido como método Setter. A menudo, un Setter se acompaña de un *Getter* (también conocido como un descriptor de acceso), el cual devuelve el valor de la variable de un atributo que tiene acceso privado. El método Setter es el más utilizado en la programación orientada a objetos, de acuerdo con el principio de encapsulación. Es decir, los atributos privados de una clase solo deben accederlos a través de los *Getters* y *Setters*.

Y con respecto a las **Clases Especiales**, tratamos la clase abstracta que es una clase que no se va a instanciar, y que sirve como estructura para subclases. En esta clase pueden existir métodos abstractos y concretos, los cuales en general definen el comportamiento generalizado de objetos.

JAVA- UNIDAD 4
BÚSQUEDA, ORDENAMIENTO Y PATRONES DE DISEÑO

Comenzamos en esta unidad recordando el concepto de **Estructura de Datos** se puede definir como una representación organizada de un conjunto de datos agrupados en una entidad u objeto para facilitar su relación y manipulación. Las estructuras de datos se pueden clasificar de la siguiente manera:

- **Estáticas**: este tipo de estructura de datos especifica su número de elementos en tiempo de compilación y no puede ser modificado en tiempo de ejecución. Las estructuras que concuerdan con este tipo son los arreglos unidimensionales y bidimensionales o matrices.
- **Dinámicas**: por el contrario de las estáticas, estas estructuras no tienen un tamaño especificado, es decir, el número de elementos que puede contener es variable y puede ser editado en tiempo de ejecución. Las estructuras que trabajamos en este curso fueron listas enlazadas, pila y cola.

Luego estudiamos Los algoritmos de búsqueda que son aquellos que están diseñados para localizar un elemento con ciertas propiedades dentro de una estructura de datos. Nos centramos en la búsqueda secuencial y la búsqueda binaria.

- La **búsqueda secuencial:** se utiliza cuando el arreglo no está ordenado o no puede ser ordenado previamente. Consiste en buscar el elemento comparándolo secuencialmente (de ahí su nombre) con cada elemento del arreglo hasta encontrarlo, o hasta que se llegue al final. La existencia se puede asegurar cuando el elemento es localizado, pero no podemos asegurar la no existencia hasta no haber analizado todos los elementos del arreglo.
- La **búsqueda binaria:** se utiliza cuando el arreglo en el que queremos determinar la existencia de un elemento está previamente ordenado. Este algoritmo reduce el tiempo de búsqueda considerablemente, ya que disminuye de forma significativa el número de iteraciones necesarias.

En cuanto a los algoritmos de ordenamiento son desarrollados con el propósito de organizar o dar un formato a las estructuras de datos. Los métodos descritos son:

- El **método burbuja:** que se basa en comparar pares de elementos adyacentes, intercambiándolos dependiendo de la condición estipulada para el ordenamiento.
- El **método de ordenamiento por selección:** consiste en ir comparando cada uno de los elementos de la lista con los demás, e ir guardando en cada comparación el índice del menor de sus elementos (para el caso de un ordenamiento descendente) y del mayor elemento (para el caso de ordenamiento ascendente), con el fin de realizar el respectivo cambio al final de todo el recorrido.

- El **método de inserción:** consiste en ir comparando el elemento a ordenar con los elementos anteriores a él, luego si la condición de ordenamiento (ya sea de forma ascendente o descendente) se cumple, el elemento se mueve a la posición superior.

Finalmente, conocimos los **Patrones de Diseño** (design patterns) que son soluciones simples a problemas específicos y comunes del diseño orientado a objetos. En otras palabras, son soluciones exitosas a problemas recurrentes. En el caso de los patrones de diseño, estos son técnicas o mecanismos definidos que permiten reutilizar soluciones en el contexto informático, con los siguientes objetivos:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Proveer soluciones a problemas recurrentes.
- Formalizar un vocabulario común entre diseñadores, y estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores, condensando conocimiento ya existente.

Los patrones de diseño de software se pueden clasificar en:

- **Patrones de creación:** proporcionan ayuda a la hora de crear objetos y son utilizados generalmente cuando esta creación requiere tomar decisiones. Estos patrones ayudan a estructurar y encapsular dichas decisiones, y la toma de decisiones puede ser dinámica.
- **Patrones estructurales:** están relacionados con cómo las clases y los objetos se combinan para dar lugar a estructuras más complejas.
- **Patrones de comportamiento:** están relacionados con algoritmos y asignación de responsabilidades a los objetos. Los patrones de comportamiento describen no solamente patrones de objetos o clases, sino también patrones de comunicación entre ellos.