Conceptos básicos Java

Introducción

Java es un lenguaje de programación de alto nivel y de propósito general desarrollado en 1995 por James Gosling y otros ingenieros de Sun Microsystems. Sun Microsystems fue adquirida por Oracle Corporation en 2010, y Java es actualmente propiedad de Oracle.

Java fue diseñado con el objetivo de ser fácil de usar y ofrecer una plataforma para la programación en red. Una de las características más importantes de Java es que es independiente de la plataforma, lo que significa que un programa escrito en Java puede ejecutarse en cualquier dispositivo que tenga una máquina virtual de Java instalada. Esto es posible gracias a que Java utiliza un bytecode que es interpretado por la máquina virtual, lo que permite a los programas Java ser ejecutados en cualquier plataforma sin requerir recompilación.

Hoy por hoy, Java es por excelencia uno de los lenguajes más empleados para el desarrollo BackEnd. Esto se debe a las siguientes características:

- Robustez
- Flexibilidad
- Versátil
- Multiplataforma
- Orientado a objetos
- Extensa comunidad y soporte

Veremos a continuación, el por qué de cada uno de estos detalles.

Robustez: Un programa es robusto si se mantiene funcional aún en circunstancias que no fueron previstas. Un programa que genere un error no recuperable en tiempo de ejecución tan pronto como el usuario ingrese inadvertidamente un comando incorrecto no será robusto.

Esta tecnología implementa todo un sistema de gestión de errores y excepciones mediante las palabras reservadas throw, try, catch, entre otros mecanismos. Permite un correcto manejo de errores, haciendo que Java, sea una de las tecnologías predilectas para gestiones bancarias, entre otros ejemplos de robustez.

Flexibilidad: Dentro de la industria informática, se define a la flexibilidad, como la capacidad de un desarrollo, a modificarse, y ser funcional en prestaciones más complejas. Dentro del paquete de frameworks disponibles para Java, hay inmensas posibilidades de agregar prestaciones constantemente a lo que sea que se esté desarrollando. Supongamos el desarrollo de un servidor con Spring Boot, al cual le agregamos toda una etapa de autenticación con Spring Security.

Multiplataforma: También dentro de las posibilidades de Java se encuentran tanto el desarrollo de aplicaciones móviles, como las que van a correr en un navegador web, o un servidor. Como también en sistemas ya no tan usados, pensados para generar programas que despliega una ventana en el sistema operativo.

Compilado e interpretado, orígenes de Java: Java tiene la capacidad de correr en cualquier entorno, sin importar su estructura (Mac, Windows, Linux, etc.) Esto es lo que se conoce como lenguaje multiplataforma. Los detalles técnicos respecto de esto, serán enunciados en breve.

Gran parte de la solidez y facilidad que exhibe la tecnología viene de la mano de su origen. Java está basado en C y C++, por lo que además de la simpleza que otorgan estos lenguajes, si se tiene experiencia en alguno de ellos, Java resultará muy familiar.

Los lenguajes previamente mencionados, son lenguajes compilados. Se debe saber que en sistemas informáticos, encontramos estos dos procesos para lograr que el código que desarrollamos, pueda llegar a ejecutarse. La diferencia reside en qué el compilador es un software encargado de tomar el código fuente escrito en algún lenguaje de programación detectando posibles errores y posteriormente, traducir este programa íntegramente a código de máquina para luego guardarlo en un archivo ejecutable.

Mientras qué un lenguaje interpretado, no atraviesa por este proceso, es tomado directamente por quien lo ejecuta, sin procesos intermedios.

Luego de enunciar lo que conocemos como proceso de compilación y de interpretación de nuestro código, podemos establecer las siguientes comparaciones:

Código interpretado:
- Ejecución lento
- Multiplataforma
- Código fuente liberado

Código compilado:
- Ejecución rápida
- Entorno simple
- Código fuente protegido

Por las características enunciadas previamente, respecto de lenguajes compilados e interpretados, y lo respectivo acerca de Java, podemos pensar a esta tecnología como una especie de híbrido, entre ambos procesos.

Retomando con ideas previas, Java es un lenguaje multiplataforma, caracterizado por correr de manera rápida y flexible, lo cual plantea una disyuntiva a partir de lo enunciado antes respecto de los lenguajes compilados e interpretados. Esta combinación de características se logra gracias a la Java Virtual Machine (máquina virtual de Java) (JVM)

Máquina Virtual Java

Una máquina virtual, no es otra cosa que un software que simula una computadora, pudiendo ejecutar programas dentro de la misma, como dentro de cualquier otro sistema. Java compila su código a un código intermedio, llamado ByteCode, disponible en archivos .class. Luego, la JVM

traduce este archivo a código de máquina adaptado a cualquier plataforma. En Windows se obtiene un archivo .jar, equivalente a los de extension .exe

Todo lo enunciado hasta el momento nos da a suponer que debe existir una JVM para cada plataforma donde se quiera correr código Java. Y esto es así, al punto de que múltiples sistemas operativos ya la tienen incorporada en su entorno.

Más elementos de Java

Emparentados estrechamente con la JVM aparece el JDK. Se traduce directamente al español como kit de desarrollo de Java y se encarga de contener las herramientas y librerías necesarias para crear y ejecutar aplicaciones de Java.

Si no se tiene instalado ningún JDK no se podrá hacer ninguna aplicación. Por otro lado, también emparentado estrechamente con la JVM aparece el JRE, que se traduce como entorno de ejecución de Java. Podemos definirlo como un conjunto de utilidades, que permite la ejecución de programas en Java. Dicho de otra manera, es la JVM junto con las librerías o API de Java.

Incluye una gran conjunto de herramientas, siendo las más destacables:

● Javac: es el compilador de Java y es el que se encarga de traducir el código fuente que escribimos en Java a ByteCode.
● Java: Es el lanzador de aplicaciones en Java, ejecuta el código ByteCode a partir de los archivos .class.
● AppletViewer: herramienta que permite visualizar applets (Aplicación Java) a correr en un navegador web, sin la necesidad de hacerlo en un navegador web.
● Javadoc: Permite la generación de documentación automática en formato HTML, a partir de comentarios establecidos en el código de la aplicación.
● JDB: Debugger (eliminación de bugs), se emplea para depurar y mejorar el código de una aplicación.
● Javap: Herramienta para desensamblar archivos .class.
● Jar: Herramienta empleada para crear y gestionar archivos de empaquetado JAR.

Anteriormente se hizo mención de las características del lenguaje:

● Robustez
● Versatilidad
● Multiplataforma
● Sólida comunidad
● Orientación a objetos, etc…

¿Por qué aprender Java?

Además de ser un lenguaje base, por lo que aprenderlo nos ayudará a comprender otras tecnologías, se emplea en:

- Aplicaciones de escritorio
- Sistemas informáticos
- Servicios Web (APIs)
- Páginas Web
- Videojuegos
- Aplicaciones de tipo mobile.

Frameworks en Java

Así como en muchas otras tecnologías disponibles hoy en día, gran parte de los atributos adjudicados a Java, se dan a partir de múltiples frameworks (FW).
Durante este curso se abordarán diferentes elementos que integran el desarrollo, haciendo especial detalle en los FW:

- Maven: se encarga de la gestión e incorporación de software externo a través de dependencias.
- Hibernate: Se encarga de mapear (replicar) atributos de clases de objetos en bases de datos del tipo relacional. Esto se administra a través de archivos declarativos del tipo XML, o anotaciones dentro del mismo código Java.
- Spring/Spring Boot: es un conjunto de módulos, qué permiten automatizar ciertas tareas dentro del desarrollo con JAva, entre ellas están:
  - Core container: inyección de dependencias e inversión de control.
  - Web: automatiza la creación de contenedores web
  - Acceso a datos: a partir de abstracciones sobre JDBC
  - AOP: Programación orientada a aspectos Spring Boot, se encarga de la etapa de configuración inicial y preparación de las aplicaciones para su producción.Esto se logra a partir de contenedores de aplicaciones integrados e iniciadores. Como plataforma de uso es conocida y recomendada Spring Initialzr.
- JUnit: Contextualizado dentro del testeo de software, es de pensar qué JUnit, es un FW para desarrollo de pruebas unitarias. También permite la implementación de pruebas de regresión junto con otras herramientas tanto para testing, como para exhibición de datos obtenidos luego de las pruebas

System.out.println (sistema de salida):

Para dar nuestros primeros pasos en Java, utilizaremos recurrentemente, este recurso. Básicamente es una función qué nos permite mostrar o imprimir en consola, mensajes y variables.
Para emplearla, no necesitamos más qué la siguiente línea:

```
System.out.println("Hola Mundo!");
```

Lo qué nos permitirá mostrar en consola, el mensaje "Hola Mundo!".

Por otro lado, a través del operador “+”, podemos concatenar texto, con variables. Supongamos una suma.

```
Integer a = 7, b = 8;
Integer resultado = a + b;
System.out.println(“El resultado de la suma es: ” + resultado);
```

Lo qué mostrará por consola: “El resultado de la suma es: 15”

Clase Scanner (sistema de entrada):

Así como la función antes vista, nos permite mostrar resultados de lo qué nuestro programa procesa. A través de esta clase, podremos ingresar información al mismo.
Más adelante detallaremos con mayor profundidad, el mecanismo para crear objetos, qué es una clase, etc. Por el momento solo diremos qué para poder hacer uso de esta funcionalidad, deberemos crear un objeto de la clase Scanner. Previamente, habremos importado la misma.

```
import java.util.Scanner;  //Importación de la clase

Scanner sc = new Scanner(System.in); //Creación del objeto
```

Dicha clase, cuenta con una serie de métodos para su utilización:

Método	**Explicación**
boolean nextBoolean()	Lee valores lógicos booleanos introducidos por el usuario.
byte nextByte()	Lee valores byte introducidos por el usuario.
double nextDouble()	Lee valores double introducidos por el usuario.
float nextFloat()	Lee valores float introducidos por el usuario.
int nextInt()	Lee valores introducidos por el usuario.
String nextLine()	Lee valores String introducidos por el usuario.
long nextLong()	Lee valores long introducidos por el usuario.
short nextShort()	Lee valores short introducidos por el usuario.

Método Main: Main, en inglés significa principal. Toda aplicación Java, necesita de su método main. En programación, esto se denomina punto de entrada al programa, ya qué es de donde se

estarán ejecutando todas las operaciones o llamados a operaciones, qué nuestro programa necesite. Dicho esto, debemos suponer qué para poder aplicar lo desarrollado hasta acá. Nuestro código deberá de estar dentro de un método main.

```
// Programa Java para leer datos de varios tipos usando la clase Scanner

import java.util.Scanner;

public class ScannerDemo
{
   public static void main(String[] args)
   {
      // Declarar el objeto e inicializar con
      // el objeto de entrada estándar predefinido

      Scanner sc = new Scanner(System.in);

      // entrada de una cadena
      String name = sc.nextLine();

      // entrada de un carácter
      char gender = sc.next().charAt(0);

      // Entrada de datos numéricos
      // byte, short y float
      int age = sc.nextInt();
      long mobileNo = sc.nextLong();
      double average = sc.nextDouble();

      // Imprima los valores para verificar si la entrada
      // fue obtenida correctamente.
      System.out.println("Nombre: "+name);
      System.out.println("Género: "+gender);
      System.out.println("Edad: "+age);
      System.out.println("Teléfono: "+mobileNo);
      System.out.println("Promedio: "+average);
   }
}
```

El porqué y cómo de la utilización de variables, se desarrollará con mayor detalle en la clase siguiente.

Lectura requerida:

Lectura ampliatoria:

https://www.oracle.com/ar/java/

Enlaces:
VSC: https://code.visualstudio.com/
Eclipse: https://www.eclipse.org/downloads/
IntelliJ: https://www.jetbrains.com/es-es/idea/
JDK: https://www.oracle.com/java/technologies/downloads/