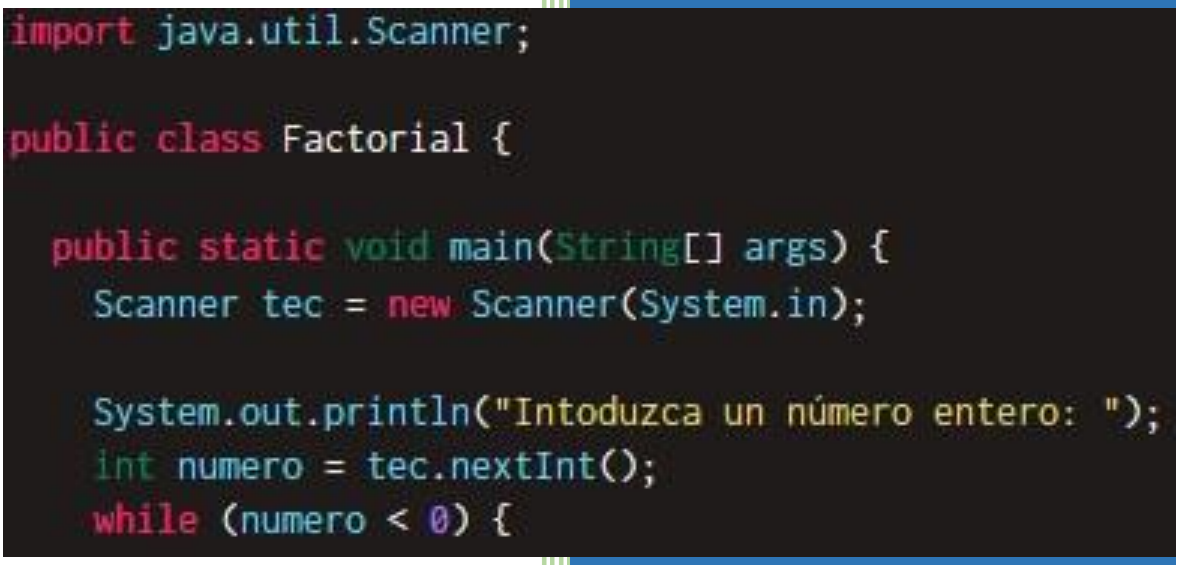
Noviembre 2020

ENTORNOS DE DESARROLLO INTEGRADOS: HISTORIA Y CARACTERÍSTICAS



```
import java.util.Scanner;

public class Factorial {

  public static void main(String[] args) {
    Scanner tec = new Scanner(System.in);

    System.out.println("Intoduzca un número entero: ");
    int numero = tec.nextInt();
    while (numero < 0) {
```

Rodríguez Jácome, David

Contornos de Desarrollo

1º curso DAW

ÍNDICE

Introducción: concepto de entorno de desarrollo integrado p. 2

1. Reseña histórica. Primeros IDEs hasta la actualidad p. 2
2. Principales funciones de los entornos de desarrollo p. 4
3. Tipos de entorno de desarrollo p. 6
4. Estructura de los entornos de desarrollo p. 6
5. Principales entornos de desarrollo p. 6
   5.1. NetBeans p. 6
   5.2. Eclipse p. 7
   5.3. Visual Studio p. 7

Introducción: concepto de entorno de desarrollo integrado

La definición más sencilla de un IDE (*Integrated Development Environment*) o entorno de desarrollo integrado es la de un programa informático que nos ayuda a programar aplicaciones, y para ello el IDE integra una serie de herramientas que se ocupan de ciertas tareas del desarrollo de software de forma automática, evitando que las asuma el programador y facilitándole un trabajo de creación que sería muy difícil sin el mencionado IDE.

1. Reseña histórica. Primeros IDEs.

Para comprender los IDEs debemos remontarnos a los años 50 del siglo XX, con la creación de los primeros ordenadores. Estas máquinas usaban un código binario para funcionar, y dicho código tan sólo se componía de unos y ceros para formas instrucciones en interminables secuencias de códigos numéricos que indicaban a los circuitos del ordenador los estados correspondientes a cada operación, lo que se denominó lenguaje máquina. Posteriormente se desarrolló otro lenguaje llamado Ensamblador, y que consistía en una serie de códigos que eran más fáciles de recordar por los usuarios del ordenador y que correspondían a secuencias de instrucciones en código binario ejecutables por la máquina. Para establecer esta correspondencia se desarrolló el primer compilador[1] llamado Sistema A-0 (Math Matic) y el B-0, que fueron desarrollados por Grace Hopper en 1952, y consistían en un traductor que pasaba las correspondencias de lenguaje Ensamblador a lenguaje Máquina. Posteriormente Hopper desarrolló FLOW-MATIC en 1955, siendo el primer lenguaje de alto nivel orientado al ámbito de la gestión, y en 1960 desarrollaría la primera versión del lenguaje COBOL. En 1954 también surgió el lenguaje de alto nivel FORTRAN con un compilador desarrollado por IBM y que estaba enfocado a cálculo numérico y computación científica. Poco después, en 1960, surgió una versión de FORTRAN extendido llamada ALTAC.

Estos lenguajes anteriormente mencionados tenían la desventaja de que necesitaban conocer la arquitectura de la máquina para funcionar, y no fue hasta la creación del lenguaje de alto nivel COBOL cuando se desarrolló el primer lenguaje multiplataforma que poseía una mayor independencia de la máquina.

[1] Cabe destacar que hay quien considera que el primer compilador fue el lenguaje de alto nivel FORTRAN desarrollado por IBM, dado que este tenía una versión completa de un compilador, mientras que el A-0 de Hopper funcionaba más como un cargador o enlazador.

En los años siguientes surgieron otros lenguajes de programación y métodos de compilación, pero no fue hasta 1975 cuando surgieron los primeros entornos de desarrollo integrados que ofrecían una interfaz de usuario gráfica. Fue en este año cuando se desarrolló Maestro I por parte de la empresa alemana Softlab Munich, siendo el primer entorno de desarrollo integrado. Consistía en un paquete de entrada de datos clave (tarjeta perforada o teclado) a disco, un emulador de la terminal IBM 3270 con *ISPF* o *Interactive System Productivity Facility* (interfaz de terminal), funciones de programación y editor de texto, y el lenguaje *COBOL*.

Posteriormente, en 1986 vio la luz Turbo Pascal lanzado por la empresa Borland, siendo el más potente y revolucionario de su época. Funcionaba en MS-DOS, CP/M, CP/M 86, Macintosh y Windows y usaba el lenguaje Pascal. Podía programarse orientado a objetos y poseía una herramienta integrada llama Turbo Profiler que permitía optimizar el código, y su velocidad de compilación era muy rápida. Borland lanzaría 9 años más tarde una evolución de Turbo Pascal llamada Delphi para el sistema Windows y que usaba una variante del lenguaje Pascal llamada Object Pascal. Unos años después, en 1989, Hewlett Packard desarrolló y lanzó un año después HP Softbench, uno de los primeros entornos de desarrollo integrado que basado en UNIX y X Window.

Hacia 1991 Alan Cooper desarrolló para Microsoft el IDE Visual Basic, derivado del lenguaje BASIC y creado como un lenguaje de programación dirigido por eventos y pensado para usar con un entorno de desarrollo simplificando la programación. Su última versión, la 6.0, integraba el paradigma de desarrollo RAD (*Rapid Application Development*), donde primero se desarrollaban las interfaces de forma rápida y luego se consensuaban por el usuario, para posteriormente empezar a crear la base de datos y el código. Estas interfaces de creaban a partir de componentes que ofrecía la propia herramienta además de poder usar componentes de terceros, ganando funcionalidad y potencia.

Microsoft también lanzaría en 1994 *Visual C++*, un IDE para para programar usando los lenguajes C, C++ y C++/CLI para el sistema Windows incluyendo bibliotecas de Windows, bibliotecas MFC y añade la posibilidad de añadir nuevas bibliotecas como DirectX, wzWidgets o SDL. Integra también el entorno de desarrollo para Framework .NET, su propio compilador y otras herramientas como IntelliSense, TeamFoundation Server, Debug, una herramienta para autogestionar la memoria, etc.

Existen otros muchos entornos como Visual IntelliJ IDEA, Borland JBuilder, JDeveloper, KDevelop, Visual C++, etc., que fueron surgiendo a lo largo de los años. Se han dado casos de IDEs que han sido liberados por sus propietarios y adoptados por comunidades de desarrolladores para continuar con un soporte de software libre, como es el caso de Eclipse. Hoy día hay infinidad de entornos de desarrollo integrados y algunos de épocas pasadas siguen siendo usados en estos días. Como ejemplos de los IDEs actuales más populares tenemos NetBeans, Eclipse y Visual Studio.

Cabe mencionar que últimamente se está extendiendo el uso de entornos de desarrollo online que, si bien no son tan potentes como los IDEs tradicionales, poseen otras ventajas como el trabajo colaborativo, poder trabajar desde cualquier lugar y desde cualquier plataforma. Ejemplos de estos IDEs son Codeanywhere o Cloud9.

2. Principales funciones de los entornos de desarrollo.

De entre todas las funciones disponibles las más importantes son:

- Edición del código fuente:

Realiza operaciones como marcar errores sintácticos, sugerencia y completado de instrucciones, uso de colores por tipo de instrucción, inserción automática de comillas, paréntesis, corchetes y espaciados, y tabulación de líneas.

- Análisis de código:

Hay dos tipos de analizador: el de tipo estático, que analiza el léxico, la sintaxis y la semántica del código que se está redactando para detectar fallos; el de tipo dinámico, que valora el comportamiento del código al mismo tiempo que se está ejecutando y obtiene información útil como tiempo de ejecución, espacio ocupado en la memoria, etc.

- Depuración de código:

El depurador se usa para eliminar errores en el código de los programas. Para ello el código se ejecuta en un simulador de instrucciones que permite detenerse cuando se verifican ciertas circunstancias establecidas por el programador, lo que hace que el programador pueda localizar el error y arreglarlo.

- Generación de código ejecutable y ejecución mediante herramientas adecuadas:

Son las herramientas que se usan para traducir el código fuente a código ejecutable, y se usará una u otra dependiendo del tipo de lenguaje usado.

- Pruebas:

Esta herramienta permite la comprobación de estabilidad del código de los programas mediante pruebas aisladas de parte del código o todo el programa o programas enteros en pruebas unitarias.

- Control de versiones:

Es una herramienta que facilita la gestión de las versiones de un programa o producto a lo largo de su desarrollo.

- Construcción de interfaces gráficas de usuario:

Esta función permite realizar interfaces que permitan la creación de entornos visuales sencillos de usar por parte del usuario para usar cómodamente el programa y comunicarse con el sistema operativo mediante imágenes gráficas.

- Generación de documentación:

Es una herramienta que crea un archivo con la documentación del código basándose en el código fuente.

- Refactorización del código:

Esta función permite alterar la estructura interna el código sin modificar su comportamiento exterior, con el objetivo de mejor la facilidad de comprensión de este, eliminar código inservible o facilitar su mantenimiento en el futuro.

3. Tipos de entorno de desarrollo.

Los tipos fundamentales de IDEs son: de código libre y gratuitos, y propietarios. Como ejemplo de código libre y gratuitos tenemos entornos de software como Eclipse o NetBeans; de tipo propietario que permiten su uso mediante un tipo de pago como XCode[2], IntelliJ IDEA, IBM RAD o Codeanywhere.

Cabe mencionar que hay quienes consideran que se deben incluir otros dos tipos adicionales: los de desarrollo específico y los multiplataforma. Citamos de ejemplos IDEs de desarrollo específico como el anteriormente mencionado XCode (destinado a programar aplicaciones para el sistema iOS de Apple) o Android Studio (orientado a la creación de aplicaciones para el sistema Android), o IDEs multiplataforma como Xamarin, que se puede usar para crear aplicaciones para iOS, Android y Windows.

4. Estructura de los entornos de desarrollo.

Las funciones de un IDE vienen determinadas por una serie de componentes: el editor de textos, que nos brinda ayuda a la hora de escribir el código; el compilador o intérprete, que transforma el código fuente de un lenguaje a código máquina; el depurador, que detecta y elimina errores en los programas; el generador automático de herramientas, usado para la visualización de componentes visuales, asistentes y utilidades de gestión; y la interfaz gráfica, que permite programar en varios lenguajes de programación con un mismo IDE.

5. Principales entornos de desarrollo.

5.1. NetBeans:

Es un IDE que está escrito en Java, lo que lo hace compatible con gran cantidad de plataformas. Fue lanzado en 1996 por Sun Microsystems (ahora propiedad de Apache Foundation) de forma libre y sin restricciones, gozando de una enorme popularidad por parte de los usuarios. Su funcionamiento se basa en el desarrollo de aplicaciones a partir de *plugins* (complementos o módulos[3]) que realizan funciones como soporte de Java, edición o soporte para el sistema de control de versiones. Por defecto tiene todos los módulos de Java incluidos. Además, tiene su propio compilador, e incluye una

[2] XCode se usa para para la creación de aplicaciones para iOS y su uso es gratuito; sin embargo, si quieres incluir tu aplicación a la tienda de aplicaciones de Apple, App Store, debes pagar $100 al mes.
[3] Archivo Java que contiene clases de Java escritas para interactuar con las API de NetBeans y un archivo que lo identifica como módulo.

herramienta para crear interfaces de usuario llamada *Matisse*, que permite realizar aplicaciones basadas en la librería Swing y un editor para programar JavaScript, Ajax y CSS. Se usa principalmente para desarrollar aplicaciones en Java, aunque también permite programar C, C/++, HTML5, Python y PHP.

5.2. Eclipse.

Otro entorno de desarrollo integrado de código abierto con licencia EPL muy popular. Fue originalmente desarrollado por IBM como reemplazo de su anterior desarrollo Visual Age, aunque ahora tiene soporte mediante la comunidad Fundación Eclipse. Este IDE integra un editor de texto con un analizador sintáctico, un depurador y un compilador. Se basa en una plataforma llamada RCP (*Rcih Cliente Platform*), y este IDE también funciona a base de módulos funcionales; sin embargo, la diferencia radica en que estos módulos se proporcionan en la RCP y es el usuario el que elige qué funcionalidades va agregando según necesite, a diferencia de otros entornos de desarrollo integrados monolíticos donde todas las funcionalidades están incluidas en el IDE, las necesite el usuario o no. Su SDK incluye herramientas de Java, además de técnicas de refactorización y análisis de código. Permite programar otros lenguajes como Ada, C/C++, JavaScript o PHP.

Entre otras características también destaca que permite compilar en tiempo real, tiene pruebas unitarias con JUnit, posee control de versiones con CVS, integración con Ant, refactorización y asistentes de ayuda a creación de proyectos.

5.3. Visual Studio.

Este es un entorno de desarrollo propietario desarrollado por Microsoft para la plataforma .NET y lanzado en 1997. Este IDE permite crear aplicaciones y servicios web compatibles con la plataforma .NET que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, embebidos, etc. Visual Studio permite compilar el código fuente en un código intermedio denominado CIL (*Common Intermediate Language*) similar al Bytecode de Java. Visual Studio también incluye el entorno de ejecución CLR (*Common Languge Runtime*) y el compilador JIT (*Just-In-Time*) integrados en .NET para traducir el código fuente a máquina que se ejecuta en la plataforma del cliente, consiguiendo la independencia de la plataforma respecto de la máquina. Soporta los lenguajes C++, C#, Visual Basic, .NET, F#, Java, Phyton, Ruby y PHP, y entornos de desarrollo web como ASP.NET MVC, Django y Windows Azure.