JAVA. Introducere

În prezent, avem diferite sisteme de operare precum Windows, macOS și Linux. Software-urile dezvoltate pentru fiecare dintre aceste sisteme sunt dependente de platformă, ceea ce înseamnă că programele create pentru macOS nu vor funcționa pe Windows și invers. Acest lucru se datorează modului diferit în care fișierele sunt executate pe diverse sisteme de operare.

Limbajul de programare Java oferă o soluție pentru această problemă. Java permite dezvoltarea de programe portabile, adică programe care pot rula pe diferite sisteme de operare fără a necesita modificări.

Principiul de bază al Java este WORA, care înseamnă „Write Once, Run Anywhere" („Scrie o dată, rulează oriunde"). Aceasta înseamnă că, odată ce ai scris codul aplicației, îl poți utiliza pe diverse dispozitive cu diferite sisteme de operare.

COMENTARII în JAVA

În Java există trei feluri de comentarii: două din ele sunt single line și multi-lines. Al treilea tip de comentariu se folosește pentru documentație. Acest tip de comentariu slujește pentru crearea fișierului HTML unde este înscrisă documentația asupra codului/aplicației/programului.
Comentariul pentru documentație începe cu simbolurile /** și se termină cu */.

În general comentariile pot fi folosite pentru a explica Codul scris în limbajul de programare Java. Cu alte cuvinte comentariile fac codul să fie mult mai readable.

Comentariile single-line încep cu 2 slash-uri.

// *Acesta este un comentariu*

Orice text scris după ce adăugăm două slash-uri va fi ignorat de Java, cu alte cuvinte nu va fi executat.

Comentariile single-line în Java pot fi inițalizate atât la începutul oricărui rând, cât și la sfârșitul oricărei linii de cod scrise.

Comentariile multi-line încep cu /* și se termină cu */

Orice text scris între /* și */ va fi ignorat de Java.

Diferența cea mai simplă dintre single și multi-line comments este faptul că dacă user-ul inițializează un comentariu pe o singură linie cu dublu // slash, atunci, la trecerea într-un rând nou, comentariul va deveni disabled și pentru a mai adăuga un text ce va fi ignorat de Java, va trebui să inițializeze din nou comentariul single-line.
Cât privește comentariile multi-line, atunci odată inițializate, acestea la trecerea într-un rând nou, nu se vor dezactiva, deci tot codul ce va urma după deschiderea multi-line comment cu /* va fi ignorat de Java până în momentul când nu se va închide cu */

Diferența dintre codul mașinii și bytecode

Diferența principală între codul mașinii și bytecode este că codul mașinii este un set de instrucțiuni în limbajul binar care poate fi

executat direct de procesor. Bytecode-ul, pe de altă parte, este un cod intermediar generat prin compilarea unui cod sursă și poate fi executat de o mașină virtuală. Un program de calculator este o colecție de instrucțiuni care efectuează o anumită sarcină. Un software special, cum ar fi compilatorii sau interpreții, transformă programul într-un cod de mașină care poate fi citit de mașină.

Ce este Codul mașinii?

Codul mașinii este un set de instrucțiuni pe care procesorul le poate executa direct. De exemplu, dacă scrii un program în C, compilatorul va transforma codul sursă în codul mașinii, pe care procesorul îl înțelege și îl execută. Aceste limbaje au o sintaxă similară cu cea a limbii engleze și este mai ușor de citit și de înțeles de către programator. Cu toate acestea, aceste programe nu sunt înțelese de un computer. Prin urmare, programul sau codul sursă este convertit în codul de mașină inteligibil pentru mașină. Un compilator sau un interpret efectuează această conversie.

Exemplu: scrii un program în **C** pentru a adăuga două numere. Compilatorul transformă acest cod în codul mașinii specific procesorului tău, iar procesorul îl execută.

Un compilator convertește întregul cod sursă într-un cod de mașină echivalent dintr-o dată. Un interpret convertește linia de cod sursă pe linie în codul echivalent al mașinii. Prin urmare, un limbaj bazat pe compilatoare este mai rapid decât un limbaj bazat pe interpreți. În cele din urmă, CPU poate executa direct codul mașinii pentru a efectua sarcina definită în program.

Ce este Bytecode?

Bytecode este creat după compilarea codului sursă. Bytecode-ul este un cod intermediar creat după compilarea codului sursă. Acest cod nu este executat direct de procesor, ci de o mașină virtuală (VM). Programele Java folosesc în principal bytecodes. La compilarea unui cod sursă Java, compilatorul Java convertește codul sursă într-un bytecode. Mai mult, acest bytecode este executabil de Java Virtual Machine (JVM). JVM convertește octetul în codul mașinii. Orice computer cu un JVM poate executa octetul. Cu alte cuvinte, orice platformă care constă dintr-un JVM poate executa un Bytecode Java.

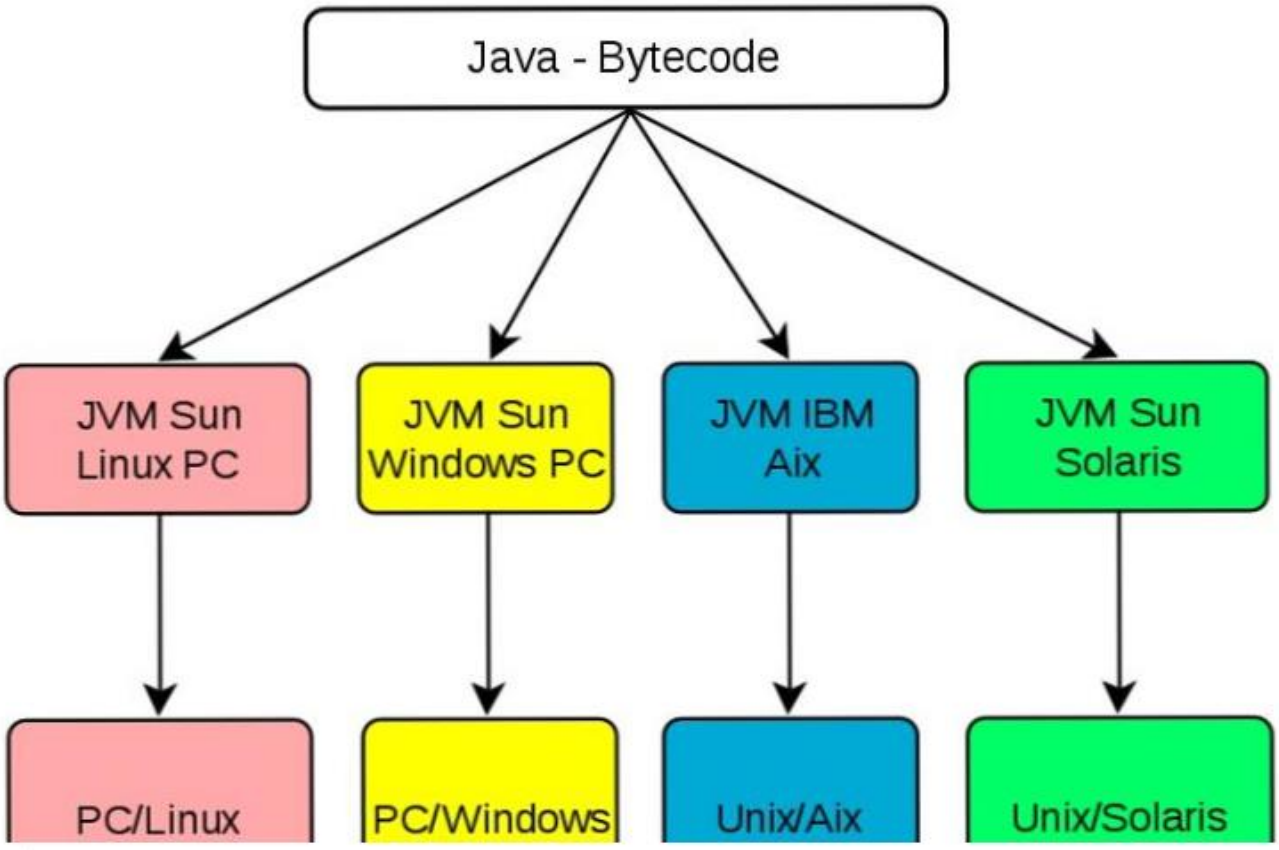
Exemplu: scrii un program în Java. Compilatorul Java transformă codul sursă în bytecode. Acest bytecode este executat de Java Virtual Machine (JVM), care la rândul său îl transformă în codul mașinii specific platformei pe care rulează.

Concluzie

Procesorul execută direct codul mașinii. Bytecode-ul, însă, este executat de o mașină virtuală, care îl transformă apoi în codul mașinii.

- **Codul Mașinii:** Executat direct de CPU.
- **Bytecode:** Executat de o mașină virtuală (de exemplu, JVM pentru Java).

În concluzie, alegerea între codul mașinii și bytecode depinde de nevoile proiectului și de prioritățile de performanță versus portabilitate. Înțelegerea acestor concepte ajută programatorii să dezvolte software mai eficient și versatil, adaptat nevoilor specifice ale fiecărui context de utilizare.



Componentele Java

În procesul studierii limbajului de programare Java o să vă întâlniți des așa abrevieri precum **JDK, JRE** și **JVM.** Am putea să spunem că acestea 3 sunt componente Java.

JDK – Java Development Kit reprezintă un pachet care pune la dispoziție mediul necesar pentru a elabora și a rula programe Java.

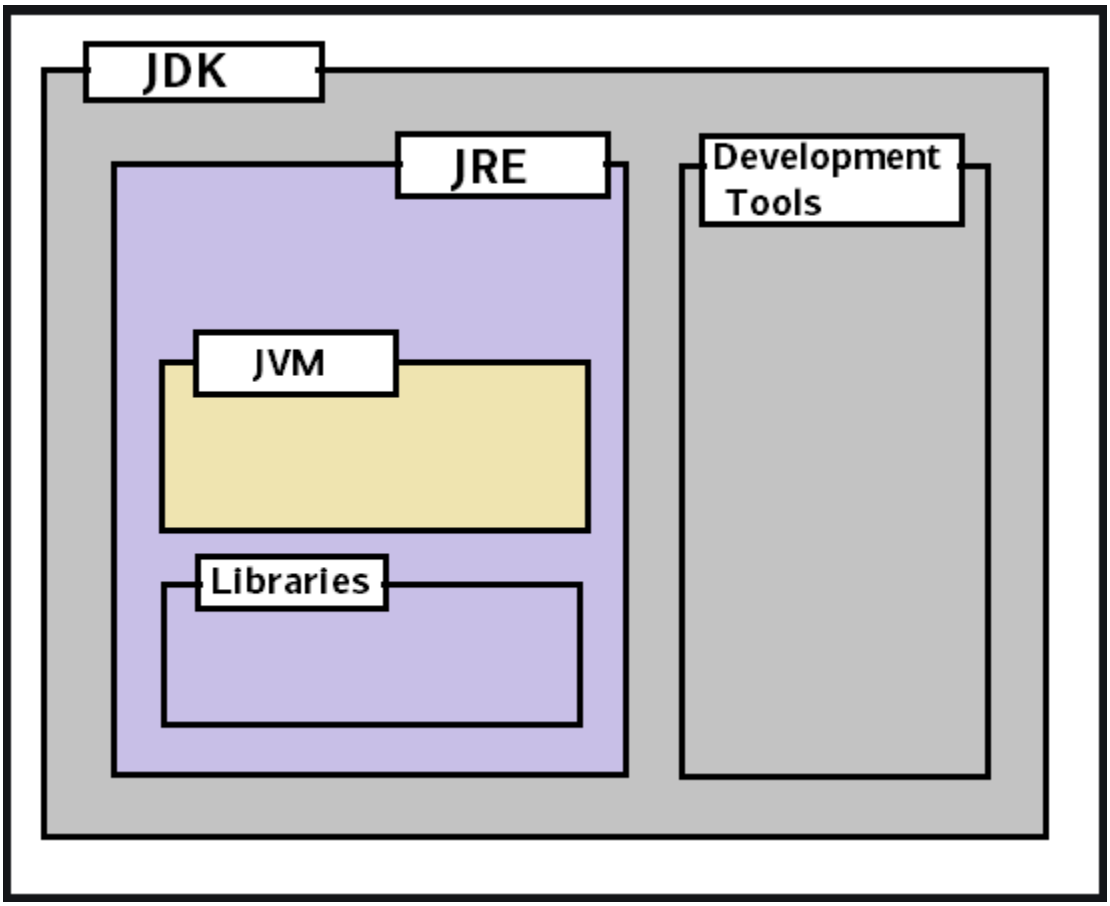
JDK include două lucruri:

- Development Tools (pentru dezvoltarea/elaborarea programului Java)
- JRE (pentru rularea programului creat)

JRE – Java Runtime Environment reprezintă un pachet de instalare care asigură rularea programului. Atenție: JRE răspunde doar de rularea/posibilitatea executării programului nu și de dezvoltarea/elaborarea lui.

JVM – Java Virtual Machine reprezintă o parte importantă atât a JDK, cât și a JRE, pentru că JVM este conținută sau încorporată în ambele.

Deci, dacă mai simplu, atunci putem spune că aplicația rulează pe o așa numită Mașină Virtuală Java. Acest fapt face posibil ca aplicațiile Java să poată să fie rulate pe diferite platforme precum MacOS, Windows, Linux fără a fi nevoie să se recompileze pentru fiecare platformă în parte.


JDK
JRE
Development Tools
JVM
Libraries

Primul program explicat pe lung:

```java
/*
Acesta este un simplu program Java.
Atribuiti file-ului numele "Exemplu.java"
 */
public class Exemplu {
    //acest program începe cu apelarea metodei main()
    public static void main(String[] args) {
        System.out.println("Un simplu program Java.");
    }
}
```

Fiecare linie de cod ce rulează în Java trebuie să se afle în interiorul unei **clase.** În exemplul nostru, am creat o clasă cu denumirea **Exemplu**. Numele clasei trebuie MEREU să înceapă cu o majusculă.

Totodată, nu trebuie să uităm despre faptul că Java este un limbaj case-sensitive, ceea ce înseamnă că **Exemplu** și **exemplu** au diferite înțelesuri.

Numele fișierului java trebuie să coincidă (să se potrivească) cu numele clasei, altfel programul nu va rula.

Notă:

Din acest moment și mai departe noi vom folosi pachetul standart al developerului Java. Adică JDK – Java Development Kit SE (Standart Edition) furnizat de către compania Oracle.

Introducerea codului. Primii pași:

Pentru multe limbaje de programare numele fișierului, care conține codul sursă al programului nu are importanță. În Java lucrurile stau altfel. Înainte de toate trebuie să înțelegem faptul că fișierului cu sursa codului trebuie neapărat să-i fie atribuit un nume. În cazul nostru am atribuit fișierului numele **Example.java**

În Java, fișierul cu codul sursă este numit oficial *unitate de compilare(*compilation unit*).* Este un fișier de tip text care conține una sau mai multe clase. O să folosim/lucrăm momentan doar fișiere care conțin doar o clasă. Compilatorul Java cere ca fișierul cu codul sursă să aibă obligatoriu extensia **.java**.

Din câte purtem observa, numele clasei este tot **Example**. Asta nu este nicidecum o coincidență. În limbajul de programare Java, tot codul trebuie să aparțină unei clase. Să se afle în interiorul ei (al clasei).

Conform regulilor, numele clasei de bază trebuie să coincidă cu numele fișierului care conține codul programului.

Compilarea programului

Ca să compilăm programul **Exemplu**, trebuie să pornim compilatorul (javac), făcând referire la fișierul cu sursa codului în command prompt.

C:\>javac Exemplu.java

Compilatorul **javac** creează fișierul **Exemplu.class** care conține versiunea bytecode al programului. Cum am spus mai devreme bytecode este reprezentarea preliminară a unui program care conține pe carea Mașina Virtuală Java le va executa. Astfel, outputul compilatorului javac nu este un cod care poate fi executat direct.

Ca să executăm programul, trebuie să utilizăm lansatorul **java**. Pentru aceasta trebuie să lansăm clasa noastră **Exemplu** ca un argument din command prompt:

C:\>java Exemplu

După executarea programului, la output vom primi mesajul din interiorul sout (System.out.println)

O analiză mai minuțioasă a primului exemplu

Cu toate că programul **Exemplu.java** nu este atât de mare, de el sunt legate câteva caracteristici destul de importante ale tuturor programelor ce au la bază limbajul de programare Java. Vom analiza minuțios fiecare parte a acestui program.

Programul începe cu următoarele linii:

```
/*
Acesta este un simplu program Java.
Atribuiti file-ului numele "Exemplu.java"
 */
```

Aceste linii conțin un *comentariu*. La fel ca și multe alte limbaje de programare, Java îți oferă dreptul de adăugare a notițelor în interiorul porgramului. Conținutul comentariului este ignorat de către compilatorul Java. Aceste comentarii descriu și explică acțiunile/operațiile programului pentru cei ce citesc codul sursă al programului dat.

Comentariul dat este un comentariu multi-lines.

Următoarea linie de cod este:

```
public class Exemplu {
```

Această linie de cod utilizează cuvântul-cheie *class* pentru declararea unei noi clase. Tot aici trebuie să menționăm și faptul că *Exemplu* reprezintă un *identificator*, adică numele clasei.

Tot ceea ce se definește în interiorul clasei, inclusiv membrii săi, trebuiesc definiți/creați în interiorul acoladelor de deschidere ({) și închidere (}) Deci limitele clasei în limbajul de programare Java sunt acoladele.

Pentru moment ne vom opri aici, dar nu înainte să spunem că absolut orice în java se petrece în interiorul unei clase. Acest fapt tot reprezintă un factor datorită căruia toate programele în Java (aproape toate) sunt object-oriented.

Următoarea linie este:

```
    //acest program începe cu apelarea metodei main()
```

Acesta este cel de-al doilea tip de comentarii în Java. Comentariu single-line. El se inițializează cu dublu slash (//).

Trecem la următoarea linie:

```
    public static void main(String[] args) {
```

Această linie de cod începe cu declararea metodei *main()*. Comentariul precedent single-line ne sugerează că anume de aici începe executarea programului nostru. Ca o regulă generală, programul Java începe mereu execuția lui prin apelarea metodei **main()**.