Lucrare Licenta

1 iunie 2016

0.1 Introducere

0.2 Tehnologii folosite

0.2.1 Java

Java este un limbaj de programare orientat-obiect realizat de către James Gosling în cadrul companiei Sun Mycrosystems (acum filiala Oracle) și a fost creat să aibă cât mai puține dependențe legate de implementare. A fost creat ca să ajute dezvoltatorii de aplicații, limbajul fiind unul de tip WORA (Write Once Run Anywhere, traducere: Scrie odată rulează peste tot). A fost lansat în anul 1995 iar de atunci a suferit o popularitate foarte mare, majoritatea aplicațiilor distribuite fiind scrise în Java, iar odată cu evoluția tehnologiei, limbajul de programare a devenit utilizat și la scrierea aplicațiilor mobile, agendelor telefonice, etc. Java este utilizat în prezent , având un real succes, și la programarea aplicațiilor destinate intranet-urilor. Astăzi, Java este unul dintre cele mai utilizate limbaje de programare, în special pentru aplicațiile web de tip client-server, având reportat un număr de 9 milioane de dezvoltatori.

Limbajul Java își însușește o mare parte din sintaxa limbajelor C și C++. Diferența dintre aceste limbaje este făcută în cadrul nivelului de jos al programării, Java având mai puține facilități , dar obiectul modelelor este de asemenea simplificat. Un program Java care este corect și care nu suferă modificări poate rula pe orice platformă care are instalată mașina virtuală Java(JVM care inseamnă Java Virtual Machine). Acest lucru este posibil deoarece codul scris în Java este compilat într-un format standard numit cod ce octeți(in engleză byte-code). Codul de octeți este de fapt un cod intermediar între codul mașină și codul sursă. Java Virtual Machine este disponibilă pentru toată lumea incepând din anul 2006, atunci când Sun a publicat un articol prin care anunțau că varianta companiei de JVM va fii open-source. Există și alți furnizori de JVM, printre care Oracle, IBM, Bea, FSF.

Oracle a furnizat pentru Java 4 platforme:

- Java Card – pentru smartcarduri (carduri cu cip)
- Java Platform, Micro Edition (Java ME) – pentru hardware cu resurse limitate gen telefoane mobile
- Java Platform, Standard Edition(Java SE)- pentru sisteme gen workstation (gasim la PC-uri)
- Java Platform, Enterprise Edition(Java EE) – pentru sisteme de calcul mari

În funcție de modul de execuție a aplicaților, limbajele de programare se împart în două categorii:

- Interpretate: un program numit interpretor citește instrucțiunile linie cu linie și le traduce în instrucțiuni mașină. Această soluție prezintă un avantaj mare datorită simplității și a portabilității. Portabilitatea este dată de faptul că sursa programului este interpretată direct. Viteza de execuție redusă este un dezavantaj evident pentru această categorie. Limbajul de programare Basic este, probabil, cel mai cunoscut limbaj interpretat.

Mai multe versiuni au apărut de-a lungul timpului, și anume un număr de 9 versiuni. Acestea sunt :

- JDK 1.0 (1996) – versiunea inițială
- JDK 1.1 (1997)

- J2SE 1.2 (1998)
- J2SE 1.3 (2000)
- J2SE 1.4 (2002)
- J2SE 5.0 (2004)
- Java SE 6 (2006)
- Java SE 7 (2012)
- Java SE 8 (2014)

Ultima versiune a limbajului Java (Java SE 8 -2014) este singura versiune gratis suportată de Oracle.

Un IDE (Integrated Development Environment) este, după cum spune numele, un mediu de lucru care permite dezvoltarea aplicațiilor specifice unui limbaj de programare. Pentru Java sunt următoarele : JCreator, Eclipse, NetBeans,BEA Workshop, BlueJ, CodeGuide,Dr Java, IntelliJ IDEA, JBuilder,JDeveloper, KDevelop.

Limbajul de programare Java a fost creat pe baza a cinci obiective majore:

1. Trebuie să fie simplu, orientat pe obiect și familiar
2. Trebuie să fie robust și sigur.
3. Trebuie să fie neutru din punct de vedere al arhitecturii și în același timp, portabil.
4. Trebuie să fie executat cu performanță ridicată.
5. Trebuie să fie interpretat, filetat și dinamic.

Oracle Corporation este proprietarul actual al implementăriioficiale a platformei Java Standard Edition, ca urmare a achiziției companiei Sun Microsystems în anul 2010. Implementarea actuală este bazată pe implementarea originală facută de cei de la Sun. Implementarea Oracle este disponibilă pentru Microsoft Windows (încă merge și pentru Windows XP), Mac OS X, Linux și Solaris.

Implementarea Oracle este distribuită în două moduri:

- The Java Runtime Environment (JRE) - conține părțile din platforma JAVA SE necesare pentru a rula programe Java
- The Java Development Kit (JDK) - este destinat programatorilor și conține programele de dezvoltare precum compilatorul Java, Javadoc, Jar și un debugger.

OpenJDK este altă implementare din familia JAVA SE care este sub licența GNU GPL. Implementarea a început când Sun a lansat codul sursă pentru Java sub licența GPL. Ca și în JAVA SE 7, OpenJDK este referința oficială a implementării limbajului de programare Java.

Scopul limbajului de programare Java este de a face compatibile toate implementările. De-a lungul istoriei, marca înregistrată a limbajului Java pentru utilizare (Sun trademark) a insistat ca toate implementările să fie compatibile. Acest lucru a iscat un conflict legal cu Microsoft după ce Sun a susținut că implementarea realizată de cei de la Microsoft nu avea suport pentru RMI (Remote Method Invocation) sau JNI (Java Native Interface) și au adăugat unele caracteristici specifice doar platformei lor. Sun a dat în judecată Microsoft în anul 1997, iar în anul 2001 au câștigat daune în valoare de 20 de milioane de dolari. Microsoft a fost de asemenea obligată să

respecte termenii licenței impuși de Sun. Microsoft nu a acceptat acest lucru și începând din anul 2001 ei nu mai livrează Java împreună cu sistemul de operare Windows.

Platforma independentă Java este esențială către Java EE (Enterprise Edition) și validare mai strictă este necesară pentru a obține o implementare. Acest mediu permite aplicații portabile pe partea de server.

Programele scrise in limbajul Java sunt recunoscute pentru viteza scăzută și consumul mai mare de memorie față de cele scrie in limbajul C++. În orice caz, viteza de execuție a programelor scrise în Java s-a îmbunătățit semnificativ cu introducerea compilării "chiar la timp" în anul 1997/1998 pentru versiunea Java 1.1, adăugarea de noi caracteristici pentru limbaj suportă o mai bună analiză a codului (clase în clase, clasa StringBuilder, optimizări în mașina virtuală Java). Odată cu versiunea 1.5, performanța a fost îmbunătățită cu adăugarea pachetului java.util.concurrent. Această implementare a fost îmbunătățită ulterior cu versiunea 1.6.

Unele platforme oferă suport hardware direct pentru Java, există microcontrollere care rulează Java în hardware și nu în mașina virtuală, și procesoare ARM care oferă suport pentru execuția codului de octeți Java.

Java utilizează un Garbage Collector automat pentru organizarea memoriei în ciclul de viață al unui obiect. Programatorul determină când sunt create obiectele iar Java runtime este responsabil cu recuperarea memoriei odată ce obiectele nu mai sunt utilizate. Odată ce nu rămân referințe la obiecte, memoria neaccesibilă devine disponibilă pentru eliberare automată de către Garbage Collector. Ceva similar unei scurgeri de memorie poate să apară dacă codul implementat de programator stochează o referință la un obiect care nu mai este nevoie de el, de obicei apare acest lucru când obiectele de care nu mai sunt nevoie sunt stocate în containere care încă sunt utilizate. Dacă sunt executate operații asupra unui obiect neexistent, va fi aruncată excepția "Null Pointer Exception".

Una din ideile din spatele organizării automată a memoriei este aceea că programatorii sunt scutiți de povara de a face manual administrarea memoriei. În unele limbaje de programare, memoria penru creare obiectelor este alocată implicit în stack (grămadă), sau explicit alocată și dealocată în memoria heap, în acest ultim caz, programatorului îi revine responsabilitatea de a administra memoria. Dacă programul nu dealocă obiectul, va apărea o scurgere de memorie. Dacă programul încearcă să acceseze sau dealoce o zonă de memorie care a fost deja dealocată, rezultatul nu este definit și este dificil de prezis, iar programul este foarte probabil să devină instabil sau să se distrugă. Acest lucru poate fii remediat parțial utilizând pointeri smart, dar aceștia adaugă o complexitate mai mare. De reținut este faptul că Garbage Collector-ul nu previne scurgerile de memorie, acele cazuri în care memoria este referențiată dar nu este folosită.

Procesul de colectare a resturilor poate să aibă loc în orice moment. De obicei, va porni atunci când un program este inactiv. Este garantat că procesul va fi declanșat daca nu mai este suficientă memorie heap liberă pentru a crea un nou obiect. Acest lucru poate duce la o pauză a procesului de rulare a programului pentru un moment. Administrarea explicită a memoriei nu este posibilă in Java.

Java nu suportă operațiile aritmetice cu pointeri din C sau C++, unde adresele obiectelor și valorile de tip usigned int (sau long int) pot fi utilizate pentru interschimbare. Acest lucru alocă colectorului de deșeuri (garbage collector) să realoce obiectele referențiate și să asigure siguranța și securitatea.

Ca în C++ sau în alte limbaje de programare orientată obiect, variabilele tipurilor primitive de date din Java sunt stocate direct in câmpuri (pentru obiecte) sau în stack (pentru metode) și nu în zona de memorie heap. Această decizie a fost luată de designerii Java din motive de performanță.

Limbajul de programare Java conține diferite tipuri de colectoare de deșeuri (garbage collectors). În mod implicit, HotSpot foloseste "parallel scavenge garbage collector". În orice caz, sunt multe alte colectoare de deșeuri care pot fi utilizate pentru administrarea memoriei heap. Pentru

90% din aplicațiile Java, este suficient utilizarea colectorului de deșeuri “Concurrent Mark-Sweep (CMS)”. Oracle tinde să înlocuiască colectorul de deșeuri CMS cu colectorul “Garbage First Collector (G1) “.