Universitatea Politehnica București

Facultatea de Electronica, Telecomunicații si Tehnologia Informației

Portabilitatea limbajelor de nivel înalt

Java

Matragociu Bogdan-Petru

432A

**Cuprins**

1. Introducere
2. Platforme disponibile
3. Crearea unui program - Compilare/Interpretare
4. Cum păstram portabilitatea ?
5. Concluzii
6. Bibliografie
7. **Introducere**

**Ce este Java**? Java este un limbaj de programare de nivel înalt, dezvoltat in anul 1995 de către SUN Microsystems, si utilizat astăzi pe scara larga atât in aplicațiile desktop sau internet cat si pentru cele destinate dispozitivelor mobile.

**De ce Java?** Java a fost de la bun început un limbaj gândit pentru ușurința si eficienta creării de software, eliminând neajunsurile altor limbaje (de ex. C++). Calitatea sa principala este independentă de platformă - un program scris o dată poate fi rulat pe diverse arhitecturi (Intel x86, SPARC etc.) fără a fi necesara ajustarea lui.

**Unde este secretul?** Dintotdeauna, una din problemele majore ale limbajelor a fost portabilitatea. Java rezolvă această problemă prin faptul ca programele, odată compilate, generează cod mașina care nu este destinat direct procesorului calculatorului gazda, ci unui procesor virtual Java, simulat de către un software ce poarta numele de Java Virtual Machine (JVM). JVM „traduce” codul mașina Java in instrucțiuni înțelese de către procesorul gazdă. Existând cate o varianta JVM pentru toate sistemele de operare si arhitecturile de procesor - însă toate simulând același procesor virtual Java! - codul mașină Java este practic independent de platforma si arhitectura. [[1]](#N1)[[4]](#N4)

Mașina virtuala Java

Compilare

|elemente independente de platforma | |elemente strict dependente de platforma|

1. **Platforme disponibile**

Date fiind direcțiile multiple de utilizare Java si necesitățile lor diferite, Oracle oferă in acest moment trei platforme Java:

* **Java SE** (Standard Edition) - platforma Java de baza pentru aplicații desktop sau server. Cuprinde componentele necesare pentru orice aplicație Java, inclusiv design de interfețe grafice complexe.
* **Java EE** (Enterprise Edition) - platforma Java dedicata aplicațiilor server, cu grad sporit de complexitate. Java EE include resursele disponibile in Java SE si adaugă elemente necesare serverelor de aplicație folosite pentru aplicații distribuite.
* **Java ME** (Micro Edition) - platforma Java dedicata dispozitivelor portabile sau cu resurse limitate (telefoane mobile, PDA-uri etc). Platforma include facilitați built-in de adaptare a platformei Java la platforma hardware pe care va rula, facilitați de comunicare in rețea si de realizare de interfețe grafice etc. [[1]](#N1)

1. **Crearea unui program - Compilare/Interpretare**

Pentru a putea scrie si rula programe Java in mod eficient, programatorul are nevoie de următoarele resurse:

* **Java Runtime Environment** (JRE) - reprezintă totalitatea resurselor necesare pentru a putea rula programe Java. Printre ele se numără mașina virtuala si un set minimal de clase predefinite Java, corespunzătoare operațiilor des întâlnite in programare (ex: lucrul cu șiruri de caractere, operații matematice uzuale etc).
* **Java Development Kit** (JDK) - pachetul conține resursele necesare creării de programe Java: compilator, generator de documentație, arhivator si o întreagă serie de alte utilitare implicate direct sau indirect in procesul de dezvoltare. In general JDK conține si JRE, pentru a putea rula aplicațiile create.
* **un mediu de dezvoltare Java** (IDE - Integrated Development Environment). Deși codul Java poate fi scris folosind orice editor de text, un mediu de dezvoltare oferă multiple facilități care sporesc eficienta programatorului (un editor avansat, managementul facil al proiectelor si resurselor atașate acestora, debugging avansat etc). Putem privi IDE-ul ca pe o interfață la uneltele puse la dispoziție de JDK; IDE-ul se folosește de ele si nu poate funcționa in lipsa lor.

Sunt disponibile doua categorii de medii de dezvoltare:

* medii de dezvoltare open-source - sunt softuri public disponibile, care pot fi descărcate si rulate de către orice utilizator. Proeminente sunt NetBeans (care beneficiază de sprijin din partea Sun si ulterior Oracle) si Eclipse (proiect inițiat si sprijinit de IBM)
* medii de dezvoltare proprietare - sunt softuri disponibile contra cost. Exemple: JBuilder, IntelliJ Idea, MyEclipse etc.

Atât JRE cat si JDK sunt public disponibile pe http://java.sun.com. JDK poate fi descărcat in cel puțin doua variante – de sine stătător („stand-alone”) sau la pachet cu mediul de dezvoltare NetBeans („bundle”). [[1]](#N1)

**Imaginea de ansamblu:**

Programele scrise in limbajul Java iau forma unui ansamblu de fișiere text cu extensia *.java,* ce conțin așa-numitul *cod sursa* (instrucțiunile Java ce compun programul). Aceste fișiere sunt apoi compilate cu ajutorul compilatorului *javac,* rezultând un set de instrucțiuni in cod mașină ce se va executa pe procesorul virtual Java si care poarta denumirea de *bytecode.* Bytecode-ul este plasat într-un fișier având același nume cu cel sursă, dar cu extensia *.class.* La rulare, mașiniivirtuale i se pasează ca argument numele clasei ce reprezintă punctul de intrare al programului; mașina virtuala încarcăbytecode-ul clasei corespunzătoare (fișierul .class) si executa codul cuprins in fișier.

Program.java Program.class Monitor PC

public class Program{

public static void main(String[] args){

System.out.println(„Hello!”);

}

}

Hello!

10101000101011110101010111001010100010101000011011111010

java

javac

Conform imaginii de mai sus, deducem ca Java este atât un limbaj compilat (codul sursa este transformat in bytecode) cat si unul interpretat (bytecode-ul este interpretat de către mașina virtuala Java astfel încât sa poată fi executat pe procesorul gazda). [[1]](#N1)[[4]](#N4)

1. **Cum păstram portabilitatea ?**

Java este prin definiție un limbaj de nivel înalt portabil deoarece bytecode-ul poate fi rulat pe orice mașină pe care are mașina virtuală Java (JVM), dar codul nu este întotdeauna portabil deoarece programatorul trebuie sa ia în vedere anumite aspecte.

* Accesarea resurselor:

Sistemele de fișiere întâlnite in sistemele de operare modeme prezintă un număr de diferențe de care programatorul Java trebuie sa tina cont daca dorește sa scrie aplicații cu adevărat independente de platforma. Aspectele majore sunt:

* numărul de arbori de resurse distincți in care este organizata informația:

- in Windows, corespunzător fiecărei partiții se creează implicit un așa-numit drive, care beneficiază de propria sa litera (C: , D: etc.) si care conține propriul arbore de resurse. Trei partiții înseamnă trei arbori diferiți, fiecare cu rădăcină sa, si deci trei rădăcini distincte

- in Linux si Unix, chiar si atunci când sistemul de fișiere este distribuit pe mai multe partiții, acestea sunt cuplate astfel încât formează un singur arbore de resurse. Nu mai exista drive-uri si litere de drive, iar rădăcină este unica.

* simbolul pentru rădăcină sistemelor de fișiere:

- in Windows acesta este \ (backslash). Exista cate o rădăcină pentru fiecare drive:

C:\, D:\ etc.

- in Linux simbolul folosit este /, iar rădăcină este unica.

* simbolul care separa doua elemente consecutive ale unei cai: in Windows se folosește \ (backslash) iar in Linux/Unix caracterul / (forward slash).
* aspectul cailor absolute (consecință a celor prezentate mai sus):

- in Windows, o cale absoluta este de forma *c: \temp\fl.txt*

- in Linux/Unix, o cale absoluta ia forma */usr/local/src/fl.txt*

* de asemenea Linux/Unix este case-sensitive, pe când alte sisteme de operare cum ar fi Windows nu.

Pentru a evita diferențele dintre sistemele de operare este recomandata folosirea clasei [File](http://docs.oracle.com/javase/7/docs/api/java/io/File.html) si a metodelor ei din API-ul Java. [[3]](#N3)[[2]](#N2)

* Elementele grafice

Elementele grafice in Java se pot realiza cu ajutorul componentelor *awt* sau *Swing.* Se recomanda folosirea elementelor Swing din punct de vedere al portabilității deoarece ele sunt desenate in totalitate in Java, spre deosebire de cele awt care folosesc posibilitățile sistemului de operare pentru a desena componente grafice care pot fi limitate. Componentele awt sunt indicate doar in cazul in care se doreste o viteza mare a aplicatiei, ele nu garantand in totalitate portabilitatea.

Se recomanda folosire unui LayoutManager si nu plasarea manuala a fiecărei componente pe ecran (FreeLayout folosit de plugin-urile pentru Netbeans sau Eclipse pentru a plasa in modul Design componente direct in fereastra). Folosirea LayoutManager-elor ofera o garanție mult mai mare ca aplicația va arata identic pe diferite platforme.

De asemenea nu se recomanda folosirea culorilor manual definite (in format RGB) ci folosirea constantelor predefinite in java.awt.Color si java.awt.SystemColor.

O alta practica necesara programatorul este aceea de a nu utiliza valori specific pentru rezoluție ci sa folosească java.awt.Toolkit.getScreenResolution. [[2]](#N2)

* Thread-uri

In Java exista următoarele abordări ale lucrului cu fire dc execuție:

* utilizarea unor soluții dc nivel înalt, care “îmbracă” lucrul cu fire de execuție oferind servicii la cheie. Amintim aici metodele *invoke\*()* din clasa **SwingUtilities;** clasele **java.util.Timer** si **java.util.TimerTask** care permit rularea de cod intr-un fir de execuție separat, sub forma unei execuții unice sau periodice; de asemenea, clasa **javax.swing.Timer** poate genera evenimente *ActionEvent* periodice pentru listenerii săi, cu perioada configurabila de către utilizator. Rularea unica sau periodica a unui job in paralel cu restul aplicației este o operație frecventa in programare si de aceea multe situații se vor putea rezolva apelând la aceste clase de nivel înalt
* utilizarea directa a claselor de baza, **Thread** si **Runnable.** Acestea prezintă o soluție flexibila pentru aplicații/necesitați simple si reprezintă de fapt baza pentru toate celelalte variante
* pentru aplicații mai complexe sau necesitați ce merg dincolo de simpla rulare unica/periodica exista infrastructuri care organizează rularea unui număr mare de thread-uri, sub forma claselor din pachetul **java.util.concurrent** si

sub pachetele sale.

Pentru a păstra codul portabil programatorul nu trebuie sa lase thread-urile sa dicteze logica programului. Thread-urile oferă puține garanții in Java, iar prioritățile lor depind de implementarea mașinii virtuale. [[2]](#N2)[[4]](#N4)

1. **Concluzii**

Java este un limbaj de programare portabil, având o multitudine de întrebuințări. Spre deosebire de alte limbaje de programare, codul este compilat o singura data, putând fi apoi rulat pe diferite mașini ce dispun de JVM (Java Virtual Machine). Cu toate acestea exista unele aspecte de care programatorul trebuie sa aibă grija pentru a menține codul portabil, cele mai importante dintre acestea fiind prezentate in acest material. Java este folosit astăzi pe scara larga, fiind unul dintre cele mai populare limbaje de programare , de aceea mulți studenți ai universităților cu profil tehnic aleg sa se specializeze in acest limbaj, printre ei numărându-se si subsemnatul.

1. **Bibliografie**

[1] - Oracle – <http://www.oracle.com/technetwork/topics/newtojava/unravelingjava-142250.html>

[2] - Java Practices – <http://www.javapractices.com/topic/TopicAction.do?Id=38>

[3] - Java API - <http://docs.oracle.com/javase/6/docs/api/>

[4] - Java How to Program (early objects) (9th Edition)

Editura: Deitel, Autori: Paul Deitel , Harvey Deitel