Curs 1
Introducere in Java

Ce este Java ?

Limbajul de programare Java

Java este un limbaj de programare de nivel înalt, dezvoltat de JavaSoft, companie în cadrul firmei Sun Microsystems. Dintre caracteristicile principale ale limbajului amintim:

- **simplitate**, elimina supraîncarcarea operatorilor, mostenirea multipla si toate "facilitatile" ce pot provoca scrierea unui cod confuz.
- **robustete**, elimina sursele frecvente de erori ce apar in programare prin eliminarea pointerilor, administrarea automata a memoriei si eliminarea fisurilor de memorie printr-o procedura de colectare a 'gunoiului' care ruleaza în fundal. Un program Java care a trecut de compilare are proprietatea ca la executia sa nu "crapa sistemul".
- **complet orientat pe obiecte** - elimina complet stilul de programare procedural
- **usurinta in ceea ce priveste programarea in retea**
- **securitate**, este cel mai sigur limbaj de programare disponibil în acest moment, asigurând mecanisme stricte de securitate a programelor concretizate prin: verificarea dinamica a codului pentru detectarea secventelor periculoase, impunerea unor reguli stricte pentru rularea programelor lansate pe calculatoare aflate la distanta, etc
- este **neutru din punct de vedere arhitectural**
- **portabililtate**, cu alte cuvinte Java este un limbaj independent de platforma de lucru, aceeasi aplicatie ruland, fara nici o modificare, pe sisteme diferite cum ar fi Windows, UNIX sau Macintosh, lucru care aduce economii substantiale firmelor care dezvolta aplicatii pentru Internet.
- **compilat si interpretat**
- asigura o **performanta ridicata** a codului de octeti
- permite programarea cu **fire de executie** (multitheaded)

- **dinamicitate**
- este **modelat dupa C si C++**, trecerea de la C, C++ la Java facându-se foarte usor.
- permite creearea unor documente Web îmbunatatite cu animatie si multimedia.

Java : un limbaj compilat si interpretat

In functie de modul de executie al programelor, limbajele de programare se împart în doua categorii :

- interpretate : instructiunile sunt citite linie cu linie de un program numit interpretor si traduse în instructiuni masina; avantaj : simplitate; dezavantaje : viteza de executie redusa
- compilate : codul sursa al programelor este transformat de compilator într-un cod ce poate fi executat direct de procesor; avantaj : executie rapida; dezavantaj : lipsa portabilitatii, codul compilat într-un format de nivel scazut nu poate fi rulat decât pe platforma pe care a fost compilat.

Programele Java sunt fi atât interpretate cât si compilate.
Codul de octeti este diferit de codul masina. Codul masina este reprezentat de o succesiune de 0 si 1; codurile de octeti sunt seturi de instructiuni care seamana cu codul scris în limbaj de asamblare. Codul masina este executat direct de catre procesor si poate fi folosit numai pe platforma pe care a fost creat; codul de octeti este interpretat de mediul Java si de aceea poate fi rulat pe orice platforma care foloseste mediul de executie Java.

```
Cod sursa Java -> (compilare) -> Cod de octeti -> (interpretare)
```

Crearea unei aplicatii simple

1. **Scriererea codului sursa**

```
2.  class FirstApp {
3.          public static void main( String args[]) {
4.                  System.out.println("Hello world");
5.          }
6.  }
```

Toate aplicatiile Java contin o clasa principala(primara) în care trebuie sa se gaseasca metoda `main`. Clasele aplicatiei se pot gasi fie într-un singur fisier, fie în mai multe.

7. **Salvarea fisierelor sursa**

Se va face în fisiere cu extensia `.java`
Fiserul care contine codul sursa al clasei primare trebuie sa aiba acelasi nume cu clasa primara a aplicatiei (clasa care contine metoda `main`)
Obs: Java face distinctie între literele mari si mici.

```
C:/java/FirstApp.java
```

8. **Compilarea aplicatiei**

   Se foloseste compilatorul Java, `javac`
   Apelul compilatorului se face pentru fisierul ce contine clasa principala a aplicatiei.
   Compilatorul creeaza câte un fisier separat pentru fiecare clasa a programului; acestea au extensia .class si sunt plasate în acelasi director cu fisierele sursa.

```
javac FirstApp.java -> FirstApp.class
```

9. **Rularea aplicatiei**

   Se face cu interpretorul java, apelat pentru unitatea de compilare corespunzatoare clasei principale, fiind însa omisa extensia `.class` asociata acesteia.

```
java FirstApp
```

   Rularea unei aplicatii care nu foloseate interfata grafica, se va face într-o fereastra sistem.

Crearea unui applet

Crearea structurii de fisere si compilarea applet-urilor sunt identice ca în cazul aplicatiilor. Difera în schimb structura programului si modul de rulare al acestuia.

1. **Scrierea codului sursa si salvarea în fisier**

```
2.  import java.awt.* ;
3.  import java.applet.* ;
4.  public class FirstApplet extends Applet {
5.          Image img;
6.          public void init() {
7.                  img = getImage(getCodeBase(), "taz.gif");
8.          }
9.          public void paint (Graphics g) {
10.                 g.drawImage(img, 0, 0, this);
11.                 g.drawOval(100,0,150,50);
12.                 g.drawString("Hello! My name is Taz!", 110, 25);
13.         }
14. }
```

   Salvarea se va face în fisierul `FirstApplet.java`

15. **Compilarea applet-ului**

```
javac FirstApplet.java -> FirstApplet.class
```

16. **Rularea applet-ului**

Applet-urile nu ruleaza independent. Ele pot fi rulate doar prin intermediul unui browser : Internet Explorer, Netscape sau printr-un program special cum ar fi appletviewer-ul din setul JDK.

1. Creearea unui fisier HTML pentru miniaplicatie (exemplu.html)

```
<html>
        <head>
                <title>First Java Applet</title>
        </head>
        <body>
                <applet code=FirstApplet.class width=400
height=400>
                </applet>
        </body>
</html>
```

11. Vizualizarea appletului

```
appletviewer exemplu.html
```

Structura lexicala a limbajului

Setul de caractere

Limbajului Java lucreaza în mod nativ folosind setul de caractere Unicode. Acesta este un standard international care înlocuieste vechiul set de caractere ASCII si care foloseste pentru reprezentarea caracterelor 2 octeti, ceea ce înseamna ca se pot reprezenta 65536 de semne, spre deosebire de ASCII, unde era posibila reprezentarea a 256 de caractere. Primele 256 caractere Unicode corespund celor din ASCII, referirea la celelate facându-se prin `\uxxxx`, unde `xxxx` reprezinta codul caracterului.

Ex:

- `\u0030 - \u0039 : cifre ISO-Latin 0 - 9`
- `\u0660 - \u0669 : cifre arabic-indic 0 - 9`
- `\u4e00 - \u9fff : litere din alfabetul Han (Chinez, Japonez, Coreean)`

Cuvinte cheie

Cuvintele rezervate în Java sunt cele din C++, cu câteva exceptii

Identificatorii

Sunt secvente nelimitate de litere si cifre Unicode, începand cu o litera. Identificatorii nu au voie sa fie identici cu cuvintele rezervate.

Literalii (constantele)

Literalii pot fi de urmatoarele tipuri

- **literali întregi**
  Sunt acceptate 3 baze de numeratie : baza 10, baza 16 (încep cu caracterele 0x) si baza 8 (încep cu cifra 0) si pot fi de doua tipuri:
  - normali, (se reprez pe 4 octeti - 32 biti)
  - lungi (8 octeti - 64 biti) : se termina cu caracterul L (sau l).
- **literali flotanti**
  Pentru ca un literal sa fie considerat flotant el trebuie sa aiba cel putin o zecimala dupa virgula, sa fie în notatie exponentiala sau sa aiba sufixul F sau f pentru valorile normale (reprez. pe 32 biti), respectiv D sau d pentru valorile duble (reprez. pe 64 biti)
- **literali logici**
  true : valoarea booleana de adevar
  false : valoarea booleana de fals
  Atentie: spre deosebire de C++, literalii întregi 1 si 0 nu mai au rolul de adevarat si false
- **literali caracter**
  Un literal de tip caracter este utilizat pentru a exprima caracterele codului Unicode. Reprezentarea se face fie folosind o litera, fie o secventa escape scrisa între apostrofuri. Secventele escape permit reprezentarea caracterelor care nu au reprezentare grafica si reprezentarea unor caractere speciale precum backslash, caracterul apostrof, etc. Secvente escape predefinite în Java:

Cod	Secventa Escape	Caracter
\u0008	'\b'	Backspace(BS)
\u0009	'\t'	Tab orizontal (HT)
\u000a	'\n'	Linie noua - linefeed (LF)
\u000c	'\f'	Pagina noua - formfeed (FF)
\u000d	'\r'	Inceput de rand (CR)
\u0022	'\"'	Ghilimele
\u0027	'\''	Apostrof
\u005c	'\\'	Backslash

- **literali siruri de caractere**
  Un literal sir de caractere este format din zero sau mai multe caractere între ghilimele. Caracterele care formeaza sirul de caractere pot fi caractere grafice sau secvente escape ca cele definite la literalii caracter. Daca sirul este prea lung el poate fi scris ca o concatenare de subsiruri de dimensiune mai mica. Concatenarea sirurilor se face cu operatorul + (`"Ana " + "`

`are " + " mere "`). Sirul vid este `""`. Dupa cum vom vedea, orice sir este de fapt o instanta a clasei `String`, definita în pachetul java.lang.

Separatori

Un separator este un caracter care indica sfârsitul unei unitati lexicale si începutul alteia. In Java separatorii sunt urmatorii: `( ) { } [ ] ; , .` Instructiunile unui program se separa cu punct si virgula

Operatori

- atribuirea: =
- operator