## 1.1 Structura lexicală a limbajului Java

## 1.2 Primul program

Crearea oricarei aplicaţi Java presupune efectuarea urmatorilor pasi:

1. **Scriererea codului sursa**

```
class FirstApp {
  public static void main( String args[]) {
  System.out.println("Hello world!");
  }
}
```

Toate aplicaţiile Java conţin o clasa principala(primară) în care trebuie sa se gasească metoda **main**. Clasele aplicaţiei se pot gasi fie într-un singur fisier, fie în mai multe.

2. **Salvarea fişierelor sursa**

Se va face în fisiere care au obligatoriu extensia `java`, nici o alta extensie nefiind acceptata. Este recomandat ca fisierul care contine codul sursă al clasei primare sa aiba acelasi nume cu cel al clasei, desi acest lucru nu este obligatoriu. Sa presupunem ca am salvat exemplul de mai sus în fisierul `c:\intro\FirstApp.java`.

3. **Compilarea aplicaţiei**

Pentru compilare vom folosi compilatorul **javac** din distribuia J2SDK. Apelul compilatorului se face pentru fisierul ce contine clasa principală a aplicatei sau pentru orice fisier/fisiere cu extensia `java`. Compilatorul creeaza cate un fisier separat pentru fiecare clasă a programului. Acestea au extensia `.class` si implicit sunt plasate în acelasi director cu fisierele sursă.

```
javac FirstApp.java
```

In cazul în care compilarea a reusit va fi generat fisierul `FirstApp. class`.

4. **Rularea aplicaţiei**

Se face cu interpretorul **java**, apelat pentru unitatea de compilare corespunzăatoare clasei principale. Deoarece interpretorul are ca argument de intrare numele clasei principale si nu numele unui fisier, ne vom pozitiona în directorul ce contine fisierul `FirstApp.class` si vom apela interpretorul astfel:

```
java FirstApp
```

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

**Atenţie**

Un apel de genul `java c:\intro\FirstApp.class` este greşit!

## 1.3

### 1.3.1 Setul de caractere

Limbajului Java lucrează în mod nativ folosind setul de caractere Unicode. Acesta este un standard international care înlocuieşte 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 doar 256 de caractere. Primele 256 caractere Unicode corespund celor ASCII, referirea la celelalte facandu-se prin `\uxxxx`, unde `xxxx` reprezintă codul caracterului.

O alta caracteristica a setului de caractere Unicode este faptul că întreg intervalul de reprezentare a simbolurilor este divizat în subintervale numite **blocuri**, cateva exemple de blocuri fiind: Basic Latin, Greek, Arabic, Gothic, Currency, Mathematical, Arrows, Musical, etc.

Mai jos sunt oferite cateva exemple de caractere Unicode.

- \u0030 **\u0039** : cifre ISO-Latin 0-9
- \u0660 **\u0669** : cifre arabic-indic 0-9
- \u03B1 **\u03C9** : simboluri grecesti $a — u$
- \u2200 **\u22FF** : simboluri matematice (V, 3, 0, etc.)
- \u4e00 **\u9fff** : litere din alfabetul Han (Chinez, Japonez, Coreean)

Mai multe informatii legate de reprezentarea Unicode pot fi obtinute la adresa ''http://www.unicode.org".

### 1.3.2 Cuvinte cheie

Cuvintele rezervate în Java sunt, cu cateva excepţii, cele din C++ si au fost enumerate în tabelul de mai jos. Acestea nu pot fi folosite ca nume de clase, interfeţe, variabile sau metode. **true, false, null** nu sunt cuvinte cheie, dar nu pot fi nici ele folosite ca nume în aplicatii. Cuvintele marcate prin * sunt rezervate, dar nu sunt folosite.

| | | | |
|---|---|---|---|
| **abstract** | **double** | **int** | **strictfp** |
| **boolean** | **else** | **interface** | **super** |
| **break** | **extends** | **long** | **switch** |
| **byte** | **final** | **native** | **synchronized** |
| **case** | **finally** | **new** | **this** |
| **catch** | **float** | **package** | **throw** |
| **char** | **for** | **private** | **throws** |
| **class** | **goto*** | **protected** | **transient** |
| **const*** | **if** | **public** | **try** |
| **continue** | **implements** | **return** | **void** |
| **default** | **import** | **short** | **volatile** |
| **do** | **instanceof** | **static** | **while** |

începând cu versiunea 1.5, mai exista şi cuvântul cheie **enum.**

### 1.3.3 Identificatori

Sunt secvente nelimitate de litere si cifre Unicode, începand cu o litera. Dupa cum am mai spus, identificatorii nu au voie sa fie identici cu cuvintele rezervate.

### 1.3.4 Literali

Literalii pot fi de urmatoarele tipuri:

- **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 reprezintă pe 4 octet (32 biti)
  — **lungi** - se reprezinta pe 8 octet (64 bit) si se termina cu caracterul L (sau l).
- **Flotanţi**
  Pentru ca un literal sa fie considerat flotant el trebuie sa aiba cel puţin o zecimala dupa virgula, sa fie în notatie exponentiala sau sa aiba sufixul F sau f pentru valorile normale - reprezentate pe 32 biti, respectiv D sau d pentru valorile duble - reprezentate pe 64 biti.
  Exemple: 1.0, 2e2, *3f, 4D.*
- **Logici**
  Sunt reprezentati de **true** - valoarea logică de adevar, respectiv **false** - valoarea

logica de fals.

**Atenţie**

Spre deosebire de C++, literalii întregi 1 si 0 nu mai au semnificata de adevarat, respectiv fals.

- **Caracter**

  Un literal de tip caracter este utilizat pentru a exprima caracterele codului Unicode. Reprezentarea se face fie folosind o litera, fie o secventă *escape* scrisa între apostrofuri. Secventele escape permit specificarea caracterelor care nu au reprezentare grafica si reprezentarea unor caractere speciale precum backslash, apostrof, etc. Secventele escape predefinite în Java sunt:

| | |
|---|---|
| - '\b' | Backspace (BS) |
| - '\t' | Tab orizontal (HT) |
| - '\n' | Linie nouă (LF) |
| - '\f' | Pagină noua (FF) |
| - '\r' | început de rând (CR) |
| '\"' | Ghilimele |
| - '\'' | Apostrof |
| - '\\' | Backslash |

- **Şiruri de caractere**

  Un literal şir de caractere este format din zero sau mai multe caractere între ghilimele. Caracterele care formeaza şirul pot fi caractere grafice sau secvenţe escape.

  Daca sirul este prea lung el poate fi scris ca o concatenare de subsiruri de dimensiune mai mica, concatenarea sirurilor realizându-se cu operatorul `+` , ca în exemplul: `"Ana " + " are " + " mere "`. Sirul vid este `""`.

  După cum vom vedea, orice sir este de fapt o instantă a clasei `String`, definita în pachetul `java.lang`.

### 1.3.5 Separatori

Un separator este un caracter care indică sfârsitul unei unitati lexicale si începutul alteia. In Java separatorii sunt urmatorii: `( )  { }  [ ]  ; , . :`

Instructiunile unui program se separă cu punct si virgula.

### 1.3.6 Operatori

Operatorii Java sunt, cu mici deosebiri, cei din C++:

- atribuirea: `=`
- operatori matematici: `+, *, /, %, ++, —` .

  Este permisă notaţia prescurtata de forma `lval op= rval`: `x += 2 n -= 3`

  Există operatori pentru autoincrementare si autodecrementare (post si pre): `x++, ++x, n—, —n`

  Evaluarea expresiilor logice se face prin metoda *scurtcircuitului:* evaluarea se opreste în momentul în care valoarea de adevăr a expresiei este sigur determinată.

- **operatori logici:** `&&(and), ||(or), !(not)`
- operatori relationali: `<, <=, >, <=, ==, ! =`
- **operatori pe biti:** `&(and), |(or), ~ (xor), ~ (not)`
- **operatori de translase:** `<<, >>, >>> (shift la dreapta fara semn)`
- **operatorul if-else:** `expresie-logica ? val-true : val-false`

- operatorul , (virgula) folosit pentru evaluarea secvenţiala a operaţiilor: `int x=0, y=1, z=2;`
- operatorul + pentru concatenarea şirurilor:

```
String s1="Ana";
String s2="mere";
int x=10;
System.out.println(s1 + " are " + x + " " + s2);
```

- operatori pentru conversii *(cast)* : `(tip-de-data)`

```
int a = (int)'a';
char c = (char)96;
int i = 200;
long l = (long)i; //widening conversion
long l2 = (long)200;
int i2 = (int)l2; //narrowing conversion
```

### 1.3.7 Comentarii

In Java exista trei feluri de comentarii:

- Comentarii pe mai multe linii, închise între /* si */.
- Comentarii pe mai multe linii care tin de documentare, închise între /** si */. Textul dintre cele doua secvente este automat mutat în documentatia aplicatiei de catre generatorul automat de documentare **javadoc**.
- Comentarii pe o singuraa linie, care incep cu `//`.

Observatii:

- Nu putem scrie comentarii în interiorul altor comentarii.
- Nu putem introduce comentarii în interiorul literalilor caracter sau sir de caractere.

Secventele `/*` si `*/` pot sa apara pe o linie dupa secventa `//` dar îsi pierd semnificatia. La fel se întampla cu secventa `//` în comentarii care incep cu `/*` sau `*/`.