

📖 **Bibliografie**

📖 **Scurtă introducere**

📖 **Descarcare JDK. Medii de dezvoltare**

📖 **Crearea unei aplicații simple**

📖 **Exerciții**

📖 **Clase**

Bibliografie

1. <http://www.oracle.com/technetwork/java/javase/documentation/index.html>
2. Horia Georgescu. **Introducere în universul Java**. Editura Tehnică București 2002
3. Ivor Horton – **Beginning Java 2, Java 7 Edition**, Wiley Pub., 2011
4. Ștefan Tanasă, Cristian Olaru, Ștefan Andrei, **Java de la 0 la expert**, ediția a II-a, Polirom 2007
5. M. Naftalin, P. Wadler - **Java Generics and Collections**, O'Reilly, 2007

Scurtă introducere

Java este un limbaj de programare de nivel înalt, dezvoltat de JavaSoft, companie în cadrul firmei Sun Microsystems.

Limbajul de programare Java a fost folosit la dezvoltarea unor tehnologii dedicate rezolvării unor probleme din cele mai diverse domenii. Aceste tehnologii au fost grupate în așa numitele platforme de lucru, ce reprezintă seturi de librării scrise în limbajul Java, precum și diverse programe utilitare, folosite pentru dezvoltarea de aplicații sau componente destinate unei anume categorii de utilizatori. Printre acestea sunt

- **J2SE** (Standard Edition) – platforma standard de lucru ce oferă suport pentru crearea de aplicații independente și appleturi
- **J2ME** (Micro Edition) – pentru programarea dispozitivelor mobile
- **J2EE** (Enterprise Edition) – oferă API-ul necesar dezvoltării de aplicații complexe, formate din componente ce trebuie să ruleze în sisteme eterogene, cu informațiile memorate în baze de date distribuite, etc, precum și suportul necesar pentru crearea de aplicații și servicii Web, bazate pe componente cum ar fi servleturi, pagini JSP, etc.

Descarcare JDK. Medii de dezvoltare

Toate distribuțiile Java sunt oferite gratuit și pot fi descărcate de la adresa

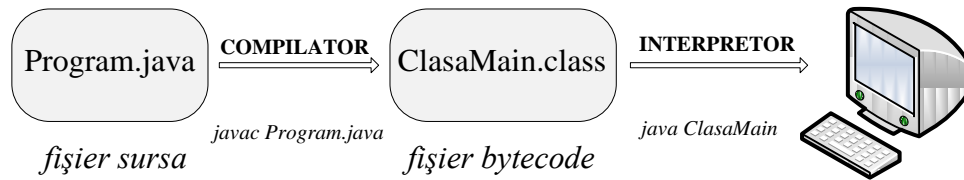
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

(vechea adresă era <http://java.sun.com>).

Menționăm că există însă mai multe medii de programare Java, ca de exemplu

- **NetBeans** - <http://www.oracle.com/technetwork/java/javase/downloads/jdk-7-netbeans-download-432126.html>
- **Eclipse**
- **JCreator**

Crearea unei aplicații simple



1. Scrierea codului sursă

Vom prezenta un exemplu clasic – afișarea argumentelor

```

class Unu{
    public static void main(String args[]){
        System.out.println("Sunt "+args.length+" argumente:");
        for(int i=0; i<args.length; i++)
            System.out.println(args[i]);
    }
}
  
```

Toate aplicațiile Java conțin o clasă principală în care trebuie să se găsească metoda `main`. Aceasta este metoda “principală”: la executarea programului sistemul detectează și execută metoda `main` a clasei al cărei nume coincide cu numele fișierului bytecode rezultat în urma compilării clasei care conține metoda principală.

i Observații

1. În Java operatorul de concatenare `+` este foarte flexibil, în sensul că permite concatenarea șirurilor de caractere cu obiecte de orice tip care au o reprezentare de tip șir de caractere (este apelată metoda `toString()` asupra căreia vom reveni).

2. În Java există o serie de convenții de scriere a numelor claselor, metodelor, câmpurilor (<http://www.oracle.com/technetwork/java/codeconventions-135099.html>) pe care vă recomand **să le respectați** și în programele scrise la laborator

- NumeClasaInitialeMari
- numeMetodaPrimaLiteraMica
- numeCampPrimaLiteraMica
- NUME_CONSTANTA

Clasele aplicației se pot găsi fie într-un singur fișier (unitate de compilare), fie în mai multe. Un fișier poate conține cel mult o clasă publică (cu modificatorul `public`).


Dacă un fișier conține o clasă publică, atunci numele fișierului trebuie să coincidă cu cel al clasei publice.

Scrierea programului poate fi realizată cu orice editor text.

2. Salvarea fișierelor sursă

Se va face în fișiere cu extensia **java**. Pentru exemplul considerat să presupunem că numele fișierului este `PrimaAplicatie.java`

Este **indicat**, dar nu obligatoriu (așa cum se vede din exemplul prezentat), ca numele unității de compilare să coincidă cu numele clasei ce conține metoda principală; în primele versiuni ale limbajului exista însă această obligativitate. În versiunile recente nici precizarea atributului `public` pentru metoda principală nu mai este obligatorie.

 **Observație:** Java face distincție între literele mari și mici.

3. Compilarea aplicației

Se folosește compilatorul Java, `javac`.

Apelul compilatorului se face pentru fișierul ce conține clasa principală a aplicației. Compilatorul creează câte un fișier separat pentru fiecare clasă a programului; acestea au extensia `class` și sunt plasate implicit în același director cu fișierele sursă.

```
javac PrimaAplicatie.java
```

În cazul unei compilări reușite va fi generat fișierul `Unu.class`.

4. Rularea aplicației

Se face cu interpretorul `java`, apelat pentru unitatea de compilare corespunzătoare clasei principale, fiind însă omisă extensia `class` asociată acesteia.

```
java nume_clasa lista_de_argumente
```

unde **nume_clasa** este numele clasei principale, iar `lista_de_argumente` reprezintă argumentele aplicației, primite de aplicație ca parametru al metodei `main` (sub forma unui vector de șiruri de caractere). `lista_de_argumente` poate lipsi dacă aplicația nu are nevoie de argumente.

Pentru exemplul prezentat, apelul poate fi de forma

```
java Unu primul ultimul
```

sau

```
java Unu
```

(**NU** java PrimaAplicatie primul ultimul)

Deoarece interpretorul are ca prim argument **numele clasei principale** și **NU numele unui fișier**, ne vom poziționa în directorul ce conține fișierul `Unu.class` înainte de a apela interpretorul. **Nu este corect un apel de genul**

```
java c:\laboratoare\Unu primul ultimul
```

Folosirea argumentelor de la linia de comandă

O aplicație Java poate primi oricâte argumente de la linia de comandă în momentul lansării ei. Aceste argumente se specifică după numele aplicației și sunt separate prin spațiu.

În cazul în care unul din argumente conține spații, atunci el trebuie pus între ghilimele.

O aplicație poate să nu primească nici un argument sau poate să ignore argumentele primite. În momentul lansării unei aplicații interpretorul parcurge linia de comandă cu care a fost lansată aplicația și, în cazul în care există argumente, trimite aplicației aceste argumente sub forma unui vector de șiruri. Acesta este primit de aplicație ca parametru al metodei `main`.

```
java Unu primul al doilea 3  
java Unu primul "al doilea" 3
```

i Observație: Argumentele sunt primite sub forma unui vector de șiruri de caractere (de obiecte de tip `String`). În cazul în care unele din acestea reprezintă valori numerice, vor trebui convertite din șiruri în numere. Acest lucru se realizează cu metode de tipul `parseNumeric` aflate în clasa corespunzătoare tipului în care vrem să facem conversia. Vom reveni asupra acestor metode în cele ce urmează.

Modul de compilare și executare a programelor Java descris mai sus este cel mai simplu posibil. El poate fi folosit utilizând *Java Development Kit* (JDK). După cum am amintit, există medii de programare Java care facilitează compilarea și executarea programelor.

Exerciții

1. Se dau ca argumente unei aplicații numere. Să se calculeze suma argumentelor primite din linia de comandă.
2. Să se scrie o aplicație Java care calculează suma radicalilor argumentelor primite de la linia de comandă. Programul va afișa un mesaj corespunzător dacă argumentele nu sunt numere pozitive.

Pentru a calcula radicalul unui număr există metoda `sqrt` a clasei `Math`, care se apelează astfel:

```
double x = Math.sqrt(25);
```

3. a) Consultați documentația pentru a vedea câmpurile și metodele clasei `Math` din pachetul `java.lang` (<http://download.oracle.com/javase/7/docs/api/>)
 b) Consultați documentația pentru a înțelege cum se pot utiliza metodele `parseInt`, `toBinaryString` din clasa `Integer` pentru a transforma un număr din baza 2 în baza 10 și invers, și scrieți un program în care să folosiți aceste metode, precum și metode și câmpuri din clasa `Math` (<http://download.oracle.com/javase/7/docs/api/>)
4. Scrieți o aplicație Java care să verifice conjectura lui Goldbach: "Orice număr întreg par mai mare decât 3 poate fi scris ca suma a două numere prime" pentru numerele pare din intervalul $[m, n]$, unde m și n sunt valori primite din linia de comandă.
- 5*. Scrieți o metodă care verifică dacă un caracter primit ca parametru este literă mare sau literă mică și afișează un mesaj corespunzător, folosind instrucțiuni de tip `if-else`. Utilizați această metodă pentru un caracter generat aleator (`Math.random()`) și pentru un caracter citit de la tastatură. Modificați metoda să testeze tipul literei folosind metodele `isLowerCase` și `isUpperCase` din clasa `Character`.

```
char litera;
litera = (char) (128*Math.random()); //caracter generat aleator

//litera citita de la tastatura
try{
    System.out.println("Introduceti un caracter");
    litera = (char)System.in.read();
    //apel metoda
}
catch(Exception ioe){
}
```

Clase

Exemplu

Vom scrie o clasă `Dreptunghi` ce are două câmpuri, reprezentând lungimea și lățimea dreptunghiului și doi constructori: un constructor fără parametri în care se inițializează lungimea și lățimea dreptunghiului cu 1 și un constructor cu doi parametri, reprezentând lungimea și lățimea dreptunghiului. Clasa va avea următoarele metode:

- o metodă de calcul al ariei dreptunghiului
- o metodă ce primește ca parametru un obiect de tip `Dreptunghi` și verifică dacă aria acestuia este mai mare decât aria dreptunghiului curent
- o metodă de afișare a dimensiunilor dreptunghiului

```
class Dreptunghi{
    double lung,lat;                //campuri
    Dreptunghi () {                 //constructori
        lung=1;
        lat=1;
    }
    Dreptunghi (double lung1,double lat1){
        lung=lung1;
        lat=lat1;
    }

    double arie () {                //metode
        return lung*lat;
    }
    boolean maiMare (Dreptunghi d){
        return arie ()<d.arie ();
    }
    void afisare () {
        System.out.println("lungime "+lung+",latime "+lat);
    }
}
```

Pentru a folosi clasa `Dreptunghi` putem adăuga în această clasă metoda `main` sau putem crea o nouă clasă care să aibă doar metoda `main` (clasă principală). Vom alege în continuare cea de a doua variantă.

```
class DreptunghiMain{
    public static void main(String arg[]){
        double a;

        //cream un dreptunghi
        Dreptunghi d;
        d=new Dreptunghi (3,5) ;

        System.out.print("Dreptunghi initial: ");
        d.afisare ();
    }
}
```

```

a=d.arie();
System.out.println("arie="+a);

//cream un nou dreptunghi, de dimensiuni primite ca argumente
int x = Integer.parseInt(arg[0]);
int y = Integer.parseInt(arg[1]);

Dreptunghi d2=new Dreptunghi(x,y);

System.out.print("Noul dreptunghi: ");
d2.afisare();

System.out.println("arie="+d2.arie());

if (d.maiMare(d2))
    System.out.println("Noul dreptunghi este mai mare");
else
    System.out.println("Dreptunghiul initial este mai mare");

d2=new Dreptunghi(); //se va apela constructorul fara argumente
System.out.print("Dreptunghi unitate: ");
d2.afisare();
    }
}

```

Exercitiu

Să se scrie o clasă `Complex` pentru lucru cu numere complexe. Clasa va avea două câmpuri, reprezentând partea reală și partea imaginară a numărului complex, trei constructori:

- un constructor fără argumente, care lasă partea reală și cea imaginară zero
- un constructor cu un argument, reprezentând partea reală a numărului complex, partea imaginară fiind inițializată cu zero (pentru crearea de numere reale)
- un constructor cu două argumente, reprezentând partea reală și partea imaginară a numărului complex.

De asemenea, clasa va avea următoarele metode:

- o metodă care returnează modulul numărului complex curent
- o metodă care returnează conjugatul numărului complex curent
- o metodă care adună la numărul complex curent un număr complex primit ca parametru (plus metode similare pentru scădere, înmulțire, împărțire)
- o metodă de afișare a numărului complex
- Să se scrie apoi clasa principală care, în metoda `main` creează două obiecte de tip `Complex`, le afișează, afișează modulul și conjugatul acestora, precum și numerele obținute adunând primul număr complex la cel de al doilea, apoi scăzând, înmulțind și împărțind cele două numere. Creați și afișați apoi trei numere complexe, apelând fiecare din cei trei constructori ai clasei