23.02.2018

Laborator PA – Java
-1-

Pagina Web de la care puteți porni căutarea documentației originale oferite de firma Sun Microsystems pentru utilizatorii limbajului Java www.java.com.

Se va lucra cu NetBeans: (download JDK 8 & NetBeans)
1) http://www.oracle.com/technetwork/java/javase/downloads/index.html#close

2) Selectati NetBeans with JDK8

3) Bifați Accept License Agreement și alegeti versiunea de NetBeans corespunzătoare SO dumneavoastră

De ce Java?
Un program este scris într-un limbaj de progrmare care trebuie apoi compilat în limbaj mașină. Cele mai multe compilatoare crează un fișier executabil separat pentru fiecare platformă, însă Java este independent de platforma deoarece transformă programul scris în java (.java) în bytecode (.class) care rulează pe (JVM) mașina virtuală java care face parte din Java Runtime Environment. Scopul lui JRE este să citească fișierele bytecode (.class) și să le ruleze oriunde cu JVM.
JDK (Java Development Kit include JRE, compilatorul de java și alte tool-uri adiționale). Scopul lui este să traducă fișierele .java în .class.
NetBeans este in IDE (Integrated Development Environment) care oferă un editor text performant, oferă asistență pentru debugging și management pentru proiecte.

Introducere în programarea orientată pe obiecte

Programarea orientată pe obiecte (POO) este o formă de programare, în care programatorii definesc ***clase*** de ***obiecte***, iar programul conţine un ansamblu de clase şi obiecte, care comunică între ele prin ***mesaje***.

Clasa este o extensie a conceptului de tip de date şi conţine o structură de date, împreună cu metodele (functiile) care se aplica acestor date.

Obiectul este o *instantiere (o instanta)* a clasei. In acelasi program se pot folosi mai multe obiecte apartinand aceleeasi clase, sau unor clase diferite. Fiecare obiect se caracterizeaza prin *stare* si *comportament*. **Starea** obiectului depinde de datele pe care acesta le contine, in timp ce **comportamentul** este dat de metodele clasei respective.

In general, comunicarea prin mesaje consta in invocarea de metode. Daca obiectul *a* invoca o metoda a obiectului *b*, aceasta poate avea ca efect modificarea starii obiectului *b* (adica modificarea unor date continute in structura de date a lui *b*) si/sau poate primi o

valoare intoarsa de metoda respectiva. Se considera ca, prin invocarea metodei, obiectul *a* a transmis un mesaj obiectului *b*, ceea ce a provocat din partea acestuia un anumit raspuns (deci *b* a manifestat o anumita comportare).

Exemplu
Sa consideram clasa poligoanelor regulate. *Structura de date* contine doua variabile: numarul de laturi (care este o variabila de tip intreg) si lungimea unei laturi (care este o variabila de tip real). *Metodele* pot fi, de exemplu, mici programe (functii) prin care se calculeaza aria, perimetrul, apotema, raza cercului circumscris etc. Pot exista si metode prin care se modifica numarul de laturi sau lungimea laturii, deci se modifica *starea* poligonului. Clasa are un nume (de ex. *PoligonRegulat*). Pot exista, evident, mai multe instante (obiecte) ale acestei clase, care toate sunt poligoane regulate, dar difera intre ele prin numarul de laturi si/sau prin lungimea laturii.
Fie *p* un *PoligonRegulat* cu 4 laturi de lungime 3.5. Invocarea metodei *p.arie()* are ca efect calcularea functiei *arie()*pentru obiectul *p*, deci se va obtine ca raspuns valoarea 12.25. Prin invocarea acestei metode, s-a transmis obiectului *p* mesajul ca se cere calcularea ariei, iar comportamentul acestui obiect a constat in punerea in executie a metodei invocate si intoarcerea valorii calculate a ariei. Valoarea intoarsa depinde, evident, de *starea* in care se gaseste *p* in momentul invocarii metodei, adica de numarul de laturi si de lungimea laturii.

Atat variabilele, cat si metodele pot fi statice sau nestatice.
Variabilele statice (ale clasei) apartin clasei, adica au aceeasi valoare pentru toate obiectele clasei respective.

Variabilele de instanta (nestatice) apartin obiectului (instantei), deci au valori diferite de la un obiect la altul.

De exemplu, daca clasa *Pasare* contine variabila *numarAripi*, aceasta variabila va avea valoarea 2 pentru toate pasarile, deci pentru toate instantele clasei respective, fiind o variabila *statica* sau *a clasei*. In schimb, variabila *varsta* va avea valori diferite pentru fiecare pasare, deci este o variabila *a instantei* (nestatica).

Metodele statice (ale clasei) pot folosi numai variabilele statice ale clasei respective, in timp ce metodele nestatice pot folosi atat variabilele statice, cat si pe cele ale instantei.

Din punct de vedere al **modului de acces**, datele si metodele unei clase pot fi publice sau private. Cele **publice** sunt accesibile din orice alta clasa, in timp ce cele **private** sunt accesibile numai din clasa careia ii apartin.

Tipuri de produse software scrise în Java

Limbajul Java este folosit cel mai frecvent pentru a scrie urmatoarele trei tipuri de programe:

aplicaţie - este un produs software care se instalează pe un anumit calculator şi funcţionează direct sub controlul sistemului de operare, având acces la toate resursele calculatorului respectiv. Una din clasele aplicaţiei trebuie sa conţină metoda principală, cu care începe execuţia aplicatiei. Această metodă se numeşte `main` şi are forma:

```
public static void main(String args[]) {
    // corpul metodei
}
```

applet (miniaplicaţie) - este un program care se transmite sub formă de cod de octeţi (*bytecode*) prin reţeaua de calculatoare şi este executat în cadrul unui navigator (browser) de Web, fără a avea acces la fişierele sau sistemul de intrare/ieşire al calculatorului pe care se execută;

servlet - un program care se execută pe un server dein reţea.

Un şablon de aplicaţie simplă în limbajul Java

În prima parte a acestui curs, în aplicaţiile făcute la curs şi la laborator vom utiliza următorul şablon:

```
class <nume_clasa> {
    public static void main(String args[]) {
      // corpul metodei main
    }
}
```

Părţile scrise cu negru (inclusiv parantezele şi acoladele) le vom considera obligatorii, iar cele scrise cursiv cu roşu sunt la latitudinea programatorului.

<nume_clasa> – este numele clasei, fiind ales de către programator cu respectarea următoarelor condiţii:

- numele clasei trebuie să înceapă cu o literă şi este format numai din litere, cifre şi - eventual - liniuţa de subliniere;
- prin convenţie (deşi aceasta nu este o regulă de sintaxă), numele de clasă începe întotdeauna cu o majusculă;
- în cazul numelor compuse din mai multe cuvinte, fiecare cuvânt începe cu majusculă;
- lungimea numelui nu este limitată, dar nu este recomandabil sa fie prea mare.

// corpul metodei main – este o succesiune de instrucţiuni şi comentarii care respectă sintaxa limbajului Java.

Părţile componente ale acestui şablon pot fi explicate pe baza cunoştintelor pe care le avem deja despre programarea orientată pe obiecte şi despre limbajele de programare.

- descrierea (definirea) unei clase începe prin cuvantul `class`, urmat de numele clasei;
- corpul clasei este cuprins între acolade { ... }
- în cazul nostru, corpul clasei conţine o singura metodă;
- metoda are forma unei funcţii, care se numeşte `main` şi are un singur argument numit args. Vom vedea ulterior că acesta este un tablou de obiecte din clasa `String`, adică un tablou de şiruri de caractere. Tot atunci vom vedea că acest argument serveşte pentru preluarea parametrilor din linia de comandă;
  - metoda `main` (metoda principală) este publică, adică poate fi utilizată de obiecte din alta clasă (în cazul nostru, ea este apelată de maşina virtuală Java la punerea în execuţie a aplicaţiei);
- metoda `main` este statică, deci apartine clasei şi nu instanţelor acestei clase;
- privită ca o funcţie, metoda `main` trebuie, în principiu, să întoarcă o valoare. În acest caz, metoda nu întoarce o valoare, deci tipul valorii întoarse este `void` (loc gol, lipsă).

Exemplu de aplicaţie simplă

Dăm aici un exemplu de aplicaţie simplă scrisă în limbajul Java. Aplicaţia este o clasă Java, care conţine metoda `main`.

Exemplul 1 Considerăm următorul program:

```
class PrimaAplicatie {
   public static void main(String args[]) {
      System.out.println("Prima noastra aplicatie a reusit!");
   }
}
```

Remarcăm urmatoarele:

- programul respectă şablonul de aplicaţie simplă adoptat de noi;
- numele clasei este `PrimaAplicatie`;
- corpul clasei este constituit dintr-o singură instrucţiune, prin care este invocată metoda `println` a obiectului `out` din clasa `System`.

Deşi nu cunoaştem încă modul în care se definesc şi se utilizează clasele şi obiectele în limbajul Java, menţionam aici că instrucţiunea de forma

```
System.out.println(<sir_de_caractere>);
```

are ca efect afişarea pe ecran a şirului primit ca argument.
În limbajul Java, instrucţiunile simple se termină cu caracterul `;` (punct şi virgulă).

Efectul executării acestei aplicaţii este că se afişează pe ecran textul

Prima noastra aplicatie a reusit!