

Programare orientată pe obiecte

Laboratorul 1

Responsabil laborator: Mihai Nan

mihai.nan.cti@gmail.com

Profesor titular: Carmen Odubășteanu



Facultatea de Automatică și Calculatoare
Universitatea Politehnica din București
Anul universitar 2016 - 2017

1 Introducere

Programarea Orientată pe Obiecte este o paradigmă de programare care utilizează obiecte și interacțiuni între acestea pentru a modela arhitectura unui program.

Aceasta constă în identificarea unor obiecte cu operații (metode) specifice asociate și realizarea comunicării între aceste obiecte prin intermediul unor mesaje. Elementul constructiv - **obiectul** - este o instanță a unei **clase** (tip definit de utilizator), iar clasele sunt membre ale unei ierarhii, fiind corelate între ele prin relații de **moștenire**.

Folosirea **POO** permite realizarea de sisteme informatice de dimensiuni mărite, cu timpi de dezvoltare, testare și mentenanță reduși față de paradigmele anterioare. Totuși, pentru a crea un sistem funcțional este necesară înțelegerea corectă a conceptelor care stau la baza acestui tip de programare, iar de acest aspect se vor ocupa cursul și laboratoarele de la această disciplină.

2 Platforma Java

Java este un limbaj de programare orientat-obiect, puternic tipizat, conceput de către James Gosling la *Sun Microsystems* la începutul anilor '90, fiind lansat în 1995. Cele mai multe aplicații distribuite sunt scrise în **Java**, iar noile evoluții tehnologice permit utilizarea sa și pe dispozitive mobile. În felul acesta, se creează o platformă unică, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat. De asemenea, acest limbaj este utilizat, în prezent, cu succes și pentru programarea aplicațiilor destinate intranet-urilor.

Limbajul împrumută o mare parte din sintaxă de la **C** și **C++**, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos. Un program Java compilat, corect scris, poate fi rulat fără modificări pe orice platformă pe care e instalată o mașină virtuală Java (în engleză **Java Virtual Machine**, prescurtat **JVM**). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format standard numit cod de octeți (în engleză byte-code) care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

Mașina virtuală Java este mediul în care se execută programele Java. În prezent, există mai mulți furnizori de **JVM**, printre care **Oracle**, **IBM**, **Bea**, **FSF**. În 2006, **Sun** a anunțat că face disponibilă varianta sa de **JVM** ca open-source.

3 Mediul integrat Eclipse

3.1 Scrierea și rularea programelor Java

Pentru scrierea și rularea unei aplicații Java, utilizând acest IDE, se vor executa următorii pași:

1. Se va alege din meniul **File** opțiunea **New** si apoi **Java Project**.
2. Se va introduce numele proiectului - **Project Name**.
3. La alegere, se poate schimba locația unde se va salva proiectul (prin modificarea câmpurilor **Location**), precum și versiunea de JRE, recomandându-se JavaSE-1.7 sau 1.8 .
4. După ce proiectul a fost creat, se vor adăuga fișiere sursă după preferințe. Acest lucru se face selectând, cu click dreapta, numele proiectului din meniul **Package Explorer**, apoi se selectează **New..** si se alege **Class**. Pentru fișierul sursă ce va conține metoda main se poate bifa opțiunea **"public static void main(String[] args)"** pentru ca aceasta să fie adăugată automat. De asemenea, se poate introduce numele pachetului în care va fi adăugată sursa, în caz contrar, aceasta fiind pusă într-un pachet "default".
5. După introducerea numelui și apăsarea butonului **Finish**, în IDE se va deschide fișierul sursă nou creat.
6. Pentru Build, se va apăsa **CTRL + B** .
7. Pentru rulare, se va apăsa **CTRL + F11** sau se va selecta, din IDE, butonul având simbol un triunghi verde.

⚠ IMPORTANT !



În **Eclipse** nu se pot compila și rula fișiere de sine stătătoare. Acestea trebuie să facă obligatoriu parte dintr-un proiect.

3.2 JDK Javadoc

Se va download documentația JDK de pe site-ul oficial [Oracle](#). Se downloadează din stanga JDK 7 Documentation. Apoi, în IDE, efectuați următorii pași:

1. Urmăriți calea în meniul **Window** → **Preferences** → **Java** → **Installed JREs** → **Edit**. Se selectează **resources.jar**, **rt.jar**, **jsse.jar**, **jce.jar** și **charsets.jar**. Apoi, se apasă **Javadoc Location..** și se completează calea către directorul **"/docs/api/"** din fișierele extrase din arhiva downloadată.

3.3 Parametri în linia de comandă

Pentru a folosi parametri în linia de comandă, se vor efectua următorii pași:

1. Se va executa un click dreapta pe numele proiectului și se va alege ***Properties*** din meniul pop-up.
2. Se va selecta ramura ***Run*** din arborele de configurare al proiectului și în câmpul ***Arguments*** se vor scrie parametrii doriți.

4 Mediul integrat NetBeans

4.1 Scrierea și rularea programelor Java

Pentru scrierea și rularea unei aplicații Java, utilizând acest IDE, se vor executa următorii pași:

1. Se va alege din meniul **File** opțiunea **New Project**.
2. Se va selecta **Java** din panoul **Categories**, respectiv **Java Application** din panoul **Projects**.
3. După apăsarea butonului **Next**, se va introduce numele proiectului - ***Project Name***.
4. La alegere, se poate schimba locația unde se va salva proiectul, precum și directorul de lucru (prin modificarea câmpurilor ***Project Location***, respectiv ***Project Folder***).
5. Se va bifa căsuța ***Set as Main Project*** și se va debifa casuța ***Create Main Class***. În acest fel, clasa care va conține metoda *main()* va avea același nume cu cel al proiectului.
6. După ce proiectul a fost creat, se vor adăuga fișiere sursă după preferințe. Acest lucru se face selectând, cu click dreapta, ***Source Packages***, apoi din meniul **New** se va alege ***Java Class***.
7. După introducerea numelui și apăsarea butonului **Finish**, în IDE se va deschide fișierul sursă nou creat.
8. Pentru compilare, se va apăsa tasta **F1** sau se va selecta, din IDE, butonul având ca simbol un ciocan.
9. Pentru rulare, se va apăsa tasta **F6** sau se va selecta, din IDE, butonul având simbol un triunghi verde.

⚠ IMPORTANT !



În **NetBeans** nu se pot compila și rula fișiere de sine stătătoare. Acestea trebuie să facă obligatoriu parte dintr-un proiect.

4.2 JDK Javadoc

Se va download documentația JDK de pe site-ul oficial [Oracle](https://www.oracle.com/technetwork/java/javase-downloads-1344955.html). Apoi, în IDE, efectuați următorii pași:

1. Alegeți din meniul principal **Tools** opțiunea **Java Platforms**.
2. Selectați platforma la care vreți să adăugați documentația din panoul din stânga al ferestrei de dialog.
3. În tab-ul **Javadoc** apăsați butonul **Add ZIP/Folder** și apoi specificați locația fișierelor **Javadoc**.
4. Apăsați butonul **Close**. Restartați IDE-ul. În acest moment, puteți accesa documentația din meniul **Help**, alegând **Javadoc References** și indicând **Java Platform SE 8**.

Observație

Există două modalități prin care se pot afla informații despre o clasă:

1. Se va accesa documentația din meniul **Help** și se va căuta numele clasei în pachetul care o conține.
2. Se va apăsa combinația de taste **ALT-F1** când cursorul este deasupra numelui clasei.

4.3 Parametri în linia de comandă

Pentru a folosi parametri în linia de comandă, se vor efectua următorii pași:

1. Se va executa un click dreapta pe numele proiectului și se va alege **Properties** din meniul pop-up.
2. Se va selecta ramura **Run** din arborele de configurare al proiectului și în câmpul **Arguments** se vor scrie parametrii doriți.

5 Reguli de baza

În continuare, se vor prezenta o serie de reguli de bază în scrierea codului sursă în limbajul Java.

1. **Nu se pot** defini variabile sau metode globale în afara unei clase (așa cum este posibil în C++). Astfel, **totul** este definit în interiorul unei clase.
2. Metoda **main** trebuie să conțină tot timpul modificatorul de acces **public**.
3. Cuvintele rezervate în Java sunt, cu câteva excepții, cele din C++. Acestea nu pot fi folosite ca nume de clase, interfețe, variabile sau metode. **true**, **false**, **null** nu sunt cuvinte cheie, dar nu pot fi nici ele folosite ca nume în aplicații.
4. În Java există trei feluri de comentarii:
 - Comentarii pe mai multe linii, închise între **/*** și ***/**.
 - Comentarii pe mai multe linii care țin de documentație, închise între **/**** și ***/**. Textul dintre cele doua secvențe este automat mutat în documentația aplicației de către generatorul automat de documentație **javadoc**.
 - Comentarii pe o singură linie, care încep cu **//**.
5. Metoda **main** primește ca parametru un vector cu elemente de tip String. **Nu** se mai poate neglija acest parametru ca în limbajele C / C++.

6 Probleme de laborator

Observație

Datele de intrare se vor da fie ca parametri în linia de comandă, fie ca valori fixe ale aplicațiilor!

6.1 Probleme standard

Problema 1 - 1 punct

Să se introducă programul de mai jos (clasa **Test**) într-un fișier cu numele **Prob1.java**, folosind mediul integrat NetBeans / Eclipse.

1. Să se compileze și să se ruleze următorul program:

Cod sursă Java

```
1 class Test {  
2     public static void main(String args[]) {  
3         System.out.println("Test Java");  
4     }  
5 }
```

2. Adăugați atributul **public** înaintea cuvântului cheie **class**, recompilați și executați programul. Rezolvați problema apărută!

⚠ IMPORTANT !



Clasele care **nu** au atributul **public** pot avea un nume diferit de numele fișierului sursă, însă dacă adăugăm atributul **public** este obligatoriu ca numele clasei și numele fișierului sursă, în care se află clasa, să coincidă.

Problema 2 - 1 punct

Să se scrie un program cu o clasă care conține două **metode** (funcții):

- metoda statică **print()** care primește un argument de tip **String** și care-l afișează;
- metoda statică **main()** care apelează funcția **print()** pentru afișarea unui șir constant.


Problema 3 - 2 puncte

1. Să se modifice programul scris la exercițiul anterior prin definirea a două clase, fiecare conținând câte o **metoda statică**:

- o clasă pentru metoda **main()**;
- o clasă pentru metoda **print()**.

Să se verifice dacă ambele clase din fișier pot fi publice.


⚠ IMPORTANT !



Într-un fișier **nu** pot fi definite două **clase publice**.
O **metodă statică** a unei clase se apelează prin:
NumeClasă.numeMetodăStatică(...)

2. Să se modifice programul anterior prin crearea a doua fișiere sursă, fiecare conținând o clasă cu o singură metodă statică. Încercați să executați ambele clase.

⚠ IMPORTANT !



O **clasă executabilă** este o clasă care conține obligatoriu **metoda main**.

Problema 4 - 1 punct

Să se scrie un program pentru afișarea tuturor argumentelor primite în linia de comandă.

Observatie

Argumentele se pot transmite astfel:

- dacă programul se rulează din NetBeans / Eclipse se vor urma instrucțiunile de la începutul laboratorului;
- dacă programul se rulează din linia de comandă:

java numeprogram arg1 arg2 arg3

Problema 5 - 2 puncte

Să se realizeze o clasă care cuprinde o metodă statică și recursivă care calculează puterea întreagă a unui număr întreg și o metodă statică pentru afișarea rezultatului funcției, alături de rezultatul funcției statice **Math.pow(baza, exp)** pentru a se putea valida. Clasa trebuie să fie executabilă.

Problema 6 - 1 punct

Să se implementeze un program format dintr-o singură clasă cu două metode:

- o metodă statică de tip *boolean* care verifică dacă un număr întreg dat este prim;
- o metodă statică *main()* care verifică funcția anterioară pentru toate numerele naturale mai mici ca 20.

Problema 7 - 1 punct

Să se scrie un program pentru verificarea ipotezei lui Goldbach pentru primele *n* numere pare, prin afișarea tuturor sumelor de două numere prime prin care poate fi exprimat un număr par. Variabila *n* poate fi inițializată cu o valoare constantă.

Cod sursă Java

Pentru afișarea unei expresii de forma $a = b + c$ se va scrie:

```
1 System.out.println(a + " = " + b + " + " + c);
```

unde *a*, *b*, *c* sunt variabile numerice de orice tip (*short*, *int*, *long*, *float*, *double*).

Problema 8 - 1 punct

Să se scrie un program pentru ordonarea unui vector de numere și căutarea binară în acest vector, folosind metodele statice *sort()* și *binarySearch()* din clasa *Arrays*. Vectorul va conține numere generate aleator folosind metoda statică *random()* din clasa *Math*, cu rezultat de tip *double*.

7 Bibliografie

- Suport de laborator - anul universitar 2014-2015
- Wikipedia - [Java](#)

Feedback

Pentru îmbunătățirea constantă a acestui laborator, vă rog să completați formularul de feedback disponibil [aici](#).

De asemenea, vă rog să semnalati orice greșală / neclaritate depistată în laborator pentru a o corecta.

Vă mulțumesc anticipat!