C2

- Variabile
- Tipuri de date
  primitive şi
  structurate.
- Paradigeme de programare.
- Concepte de bază ale programării orientate pe obiect.

Obiective

Dupa parcurgerea acestui curs ar trebuie sa puteti:

- ☐ descrie conceptele de variabila, tip de data simplu si structurat
- ☐ constanta, literali;
- ☐ caracteristicile programarii orientate pe obiect.

Variabile

- **Variabila = nume dat unui grup de locaţii de memorie ce la un moment dat, stochează o singură valoare**
- **Orice variabilă trebuie declarată**
- **Caraterizată prin: nume, valoare, tip, persistenţă şi vizibilitate.**
- **Variabilele pot fi iniţializate în momentul declarării**

```
Tip   Nume  Valoare
int    i  =  5;
```

La nivel conceptual, **variabila** este o abstactizare a componentei electrice numite memorie internă. La nivel de limbaj variabila reprezintă **un nume simbolic**, numit şi identificator, prin intermediul căruia vom putea accesa şi stoca valori în RAM-ul calculatorului. În Java orice variabilă trebuie declarată explicit, adică pentru fiecare nume de variabilă din program este obligatoriu să avem o line de forma `Tip Nume = [Valoare iniţială];` ce se citeşte, variabila cu numele `Nume` se declară de tipul `Tip`. Declaraţia leagă (asociază) de numele variabilei un grup de caracteristici. Variabila poate fi declarată în orice locaţie din bloc, dar este preferabil să fie scrisă la începutul acestuia. Se pot declara mai multe variabile într-o singură linie.În Java **variabilele** sunt de două feluri: **locale** (declarate într-o metodă) şi **câmp** (declarate într-o clasă). Caractersticile legate de numele variabilei prin declaraţie sunt:

Numele variabilei este formula prin care variabila este identificată în textul aplicaţiei Java.
Valoarea variabilei este conţinutul locaţiilor de memorie alocate în RAM. Modul de interpretare al locaţiilor este determinat de tipul variabilei.

Tipul asociat unei variabile prin declaraţie determină mulţimea valorilor şi a operatorilor ce pot acţiona asupra variabilei. Java este un limbaj puternic tipizat, adică tipul oricărei expresii, în particular a unei variabile, trebuie să poată fi determinat în momentul compilării.
Persistenţa defineşte intervalul de timp din execuţia aplicaţiei Java în care se zice ca variabila există (în sensul că i se alocă o anumită regiune de RAM). Variabilele locale există atât timp cât metoda în care s-au declarat este activă în sensul că aceasta se rulează. Variabilele câmp există atât timp cât obiectul al cărui membru sunt există. Dacă variabila câmp este şi statică câmpul va exista cât timp clasa respectică rămâne încărcată în JVM

Vizibilitatea defineşte porţiunea de cod din care un nume de variabilă este recunoscut de compilator. În afara acestei porţiuni de cod utilizarea respectivului nume va genera o eroare de sintaxă. Pentru variabilele declarate în interiorul unui bloc, vizibilitatea începe din locul declaraţiei şi se termină la acolada de închidere a blocului. Este posibil într-un bloc intern acelaşi numele de variabilă să fie redeclarat, situaţie în care vizibilitatea numelui extern nu o include pe cea a numelui intern.
Iniţializarea variabilelor se poate face explicit, adică ele primesc o valoarea explicită în momentul delcarării. Dacă această valoare implicită nu este dată, variabilele de tipul întreg primesc valorea 0.

Nume de variabile

- **Numele unei variabile trebuie să înceapă cu o literă din alfabet sau underscore (_) sau $**
- **Următoarele caractere pot conţine şi cifre, dar nu spaţii sau operatori (+, -, *, / etc.)**
- **Nu se poate folosi un cuvânt rezervat al limbajului Java pe post de nume de variabilă**
- **Lungimea numelui este nelimitată**
- **Este semnificativă scrierea cu litere mari sau mici**

OK - ✔	**Greşit - ✘**
`a x1 masa x2 masa$ideala masa_ideala`	`masa#ideala a-1 x*1 for if 1x`

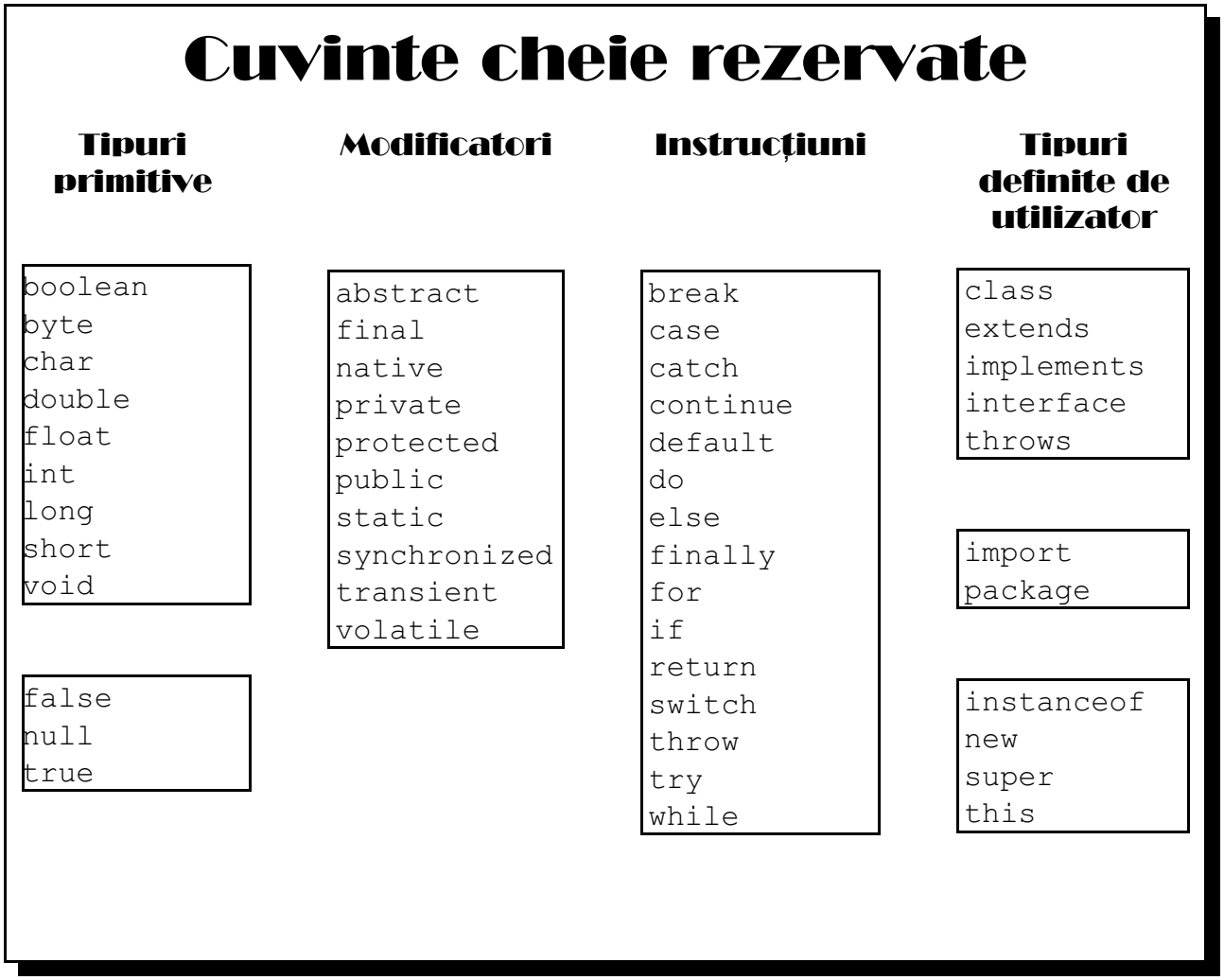
Variabile locale
Variabilele locale sunt declarate într-o metodă sau un bloc de cod Java. Din acest motiv vizibilitatea şi persistenţa lor este restrânsă la respectiva porţiune de cod. Acestea vor putea fi acceste (sunt vizibile) numai în interiorul respectivului bloc (variabilele declarate în afara metodelor pot fi accestate din orice metodă) şi vor avea alocat un spaţiu propriu în RAM atât timp cât blocul de cod respectiv se rulează. O variabilă locală trebuie să primească valoare înainte de a fi utilizată într-o expresie, altfel compilatorul va da un mesaj de eroare.

Fie declaraţia:

```
int i = 5;
```

Aceasta declară o variabilă cu numele `i` de tipul `int` şi care are valoarea iniţială `5`.

Variabila, ca rezultat al unui proces de abstractizare, constă dintr-un grup de caracteristici (o denumire alternativă este cea de atribute). Procesul de asociere a unei valori unei caracteristici de variabilă poartă denumirea de legare (**binding**). Unele dintre carateristicile variabilei sunt cunoscute la momentul compilării, de exemplu numele şi tipul acesteia. Se zice că aceste caracteristici sunt legate static (**static binding**). Alte caracteristici, de exemplu valoarea - cu excepţia cazului când aceasta este iniţializată explicit, se leagă în timpul rulării aplicaţiei. Se zice că ele sunt legate dinamic (**dinamic binding**).



Se numeşte gramatică o mulţime de definiţii formale ce alcătuiest structura sintactică (numită, pe scurt şi sintaxă) a unui limbaj. Gramatica este definită în termenii unor reguli de generare ce descriu ordinea constituenţilor dintr-o propoziţie. Fiecare regulă are o parte stângă, numită categorie sintactică şi o parte dreaptă formată dintr-o secvenţă de simboluri. Simbolurile pot fi terminale sau neterminale. Un simbol terminal corespunde unui constituent de limbaj care nu mai are structură sintactică internă (este un element minimal de limbaj). Cuvintele cheie reprezintă o mulţime de simboluri terminale care fac parte din sintaxa limbajului Java. Restricţia de bază referitoare la acesta este cea conform căreia nu se pot da nume de variabile (sau alte elemente de limbaj definite de programator: constante, clase, metode ...) care sa aibă aceeaşi scriere cu acestea.

Pe lângă cuvintele cheie de mai sus, numele `const` şi `goto` sunt şi ele rezervate şi nu pot fi folosite ca nume de variabile.

Tipuri de date în Java

- **Tipuri de date primitive:**
  - ☞ **6 tipuri de date numerice:**
    - **întregi:** `byte`, `short`, `int`, `long`
    - **reali:** `float`, `double`
  - ☞ **1 tip de dată pentru caractere:** `char`
  - ☞ **1 tip de dată bolean:** `boolean`
- **Tipuri de date definite de utilizator:**
  - ☞ **clasele:** `class`
  - ☞ **intefeţele:** `interface`
  - ☞ **tablourile**

Tipuri de date primitive

Tipurile de date primitive sunt implementate la nivelul JVM. Tipic pentru Java este că numele lor începe cu literă mică. Astfel, `float` este un tip de data primitiv, în timp ce `Float` este un obiect.

Tipuri întregi

Tipurile întregi se folosesc pentru manipularea numerelor ce nu au parte zecimală. Valorile numerice întregi pot fi atât negative cât şi pozitive, se mai spune că pot fi cu semn. Tipurile întregi din Java sunt:

Tip	Spaţiu alocat	Domeniu de valori
`byte`	1 octet	-128 la 127
`short`	2 octeţi	-32,768 la 32,767
`int`	4 octeţi	-2,147,483,648 la 2,147,483,647
`long`	8 octeţi	-9,223,372,036,854,775,808 la -9,223,372,036,854,775,807

Exemple de valori întregi:

- în baza 10: 1, 2, 6700000000000000000L
- în baza 8 (numărul este prefixat cu 0): 07
- în baza 16 (numărul este prefizăt cu 0x): 0xaa

Tipuri reale

Tipurile reale se folosesc pentru manipularea numerelor ce au pare zecimală. Ele se repezintă în virgulă flotantă conform standardului IEEE 745. Tipurile reale din Java sunt:

Tip	Spaţiu alocat	Domeniu de valori	Precizie
`float`	4 octeţi	aproximativ ± 3.40282347E+38F	7 zecimale
`double`	8 octeţi	aproximativ ± 1.7.9769313486231570E+308	15 zecimale

La tipul `double` precizia este dublă faţă de tipul `float`.

Exemple de valori reale:

`float`: 1.34F;
`double`: .337, 4.345345E108, 3.0, 3.46D\

Tipul caracter

Tipul `char` se foloseşte pentru stocarea caracterelor individuale, spre deosebire de tipul obiect `String` ce se foloseşte pentru stocarea şirurilor de caractere. Codificarea caractereleor în Java respectă standardul Unicode pentru reprezentarea caracterelor în orice limbă pe 16 biţi (2 octeţi). Primele 256 de caractere corespund cu mulţimea de caractere ISO Latin 1 din a cărei parte este şi codificarea ASCIII.

Exemple de valori caracter:

'A' - litera A din alfabet
'\u03C0' - valoare definită prin codificare Unicode în hexazecimal a lui π
'\n' - caracter special - line feed

Majoritatea valorilor de tipul caracter se afişează pe ecran, există totuşi un grup de caractere neafişabile acesta fiind rezevate pentru comanda dispozitivelor de afişare. Tipic, ele încep cu backslash \ urmate de un alt caracter

Tipul bolean

Tipul `boolean` are două valori distincte, `false` şi `true`. Se foloseşte pentru evaluarea unor condiţii locice. Nu se poate converti la o valoare întreagă

Tipuri de date definite de utilizator

Clasele, intefeţele, tablourile sunt tipuri de date definite de utilizator. După definirea acestora de vor putea declara variabile de acel tip după modalitatea utilizată la tipurile primitive.

Literali

Literalii reprezintă valori care nu sunt stocate în variabile.

- **numerici:**

- întregi: (Implicit int)

0	18	-23232	(`int` în baza 10)
02	077	0123	(`int` în baza 8)
0x0	0xff	0X1FF	(`int` în baza 16)
1L	022L	0x1FFFL	(`long`)

- reali: (Implicit double)

1.0	4.2	0.47	(`double`)
1.23e12		4.12E-9	(`double`)
6.3f	5.62F	4.12E9F	(`float`)

- **nenumerici:**

- booleeni:

```
true false
```

- caracter:

```
'a' '\n' '\077' '\u005F'
```

- şir:

```
"Salut/n"
```

Literalii şir se formează prin cuprinderea între ghilimele a unui grup de caractere. Pentru manipularea acestra se va folosi un tip obiect numit `String`, implementet la nivel de bibliotecă în Java şi nu tipul primitiv `char` ce poate stoca numai o singură valoare.

Erori la declararea de variabile

Fie următoarele declaraţii de variabile:

```
1 byte b = 130;

2 short s1 = 123, s2

3 int i = b*b*b*b;

4 long l = i+i+i;

5 double new=73.8;

6 boolean mergeinvacanta = true;
```

```
7 boolean max = s1>b;

8 char sal=”Salutare !”;

9 char amic = ’a’;
```

Linia 1: b este o variabilă de tipul byte, deci poate stoca valori în domeniul -128 , 127. Compilatorul va da un mesaj de eroare deoarece valoarea de iniţializare este în afara domeniului admis.

Linia 2: Este omis terminatorul de instrucţiune (;) de la sfărşitul declaraţiei.

Linia 5: Declaraţia este incorectă deoarece `new` este cuvânt rezervat în Java;

Linia 8: Variabilele de tipul `char` pot stoca o singură valoare. Dacă se foloseşte `String` în loc de `char` e Ok.

Iată două aplicaţii ce folosesc declaraţii de variabile locale şi pachete Java diferite pentru a calcula masa ideala în funcţie de vârsta şi înalţinea unei persoane:

Varianta 1:

```
import java.util.Scanner;

public class MasaIdealaV1 {
 public static void main(String[] args) {
    //declaratii de variabile locale
    double masa, varsta, inaltimea;
    Scanner intrare;

    intrare = new Scanner(System.in);


    //afisarea pe ecran a textului Ce varsta ai:
    System.out.print("Ce varsta ai: ");
    //citirea unui numar real de la tastatura
    varsta = intrare.nextFloat();

    System.out.print("Ce inaltime ai: ");
    inaltimea = intrare.nextFloat();

    //formula de calcul a masei ideale
    masa = 50 + 0.75 * (inaltimea-150) + 0.25 * (varsta - 20);

    System.out.println("Masa ideala (barbat) = " + masa + " kg");
    System.out.println("Masa ideala (femeie) = " + 0.9*masa + " kg");

 }
}
```

Varianta 2:

```
import javax.swing.JOptionPane;
public class MasaIdealaV2 {
 public static void main(String[] args) {

  //declaratii de variabile locale
  float masa, inaltimea;
```

```
  int varsta;

  //afisarea pe ecran a ferestrei de dialog Ce varsta ai:
  String intrare = JOptionPane.showInputDialog("Ce varsta ai: ");
  //citirea unui numar real de la tastatura
  varsta = Integer.parseInt(intrare);

  intrare = JOptionPane.showInputDialog("Ce inaltime ai (in cm): ");
  inaltimea = Float.parseFloat(intrare);

  masa = 50F + 0.75F * (inaltimea-150F) + 0.25F * (varsta - 20F);

  String masaideala = "Barbat = " + masa + " kg\n";
  masaideala = masaideala + "Femeie = " + 0.9*masa + " kg";

    JOptionPane.showMessageDialog(null,    masaideala,"Masa
ideala",JOptionPane.INFORMATION_MESSAGE);
 }
}
```

Constante Java

În Java cuvântul cheie `final` **defineşte o constantă**.

Se declară la fel ca şi o variabilă însă poate primi valoare o singură dată.

Prin convenţie, numele constantelor se scriu cu majuscule.

```
final double PI = 3.1415926535;
```

Dacă constanta este declarată într-o metodă, ea va fi vizibilă numai în interiorul acesteia. În situaţia în care se doreşte ca o constantă să fie vizibilă la nivel de clasă se zice că declarăm o constantă de clasă şi tebuie să folosim cuvintele cheie **`static final`** . Declaraţia ei se scrie în afara metodelor clasei şi va fi de forma:

```
public static final double PI = 3.1415926535;
```

Concepte de programare orientată pe obiect

Abstractizare ☞	**Trecerea de la realitate la un model (o simplificare a realităţii)**
Clasă ☞	**Tip de dată definit de utilizator ce realizează încapsulare a datelor (câmpuri) şi a operaţiilor (metodele) ce se pot face cu datele**
Instanţiere ☞	**procesul de creare a unei variabile de tipul clasă prin folosirea lui** `new`
Obiect ☞	**denumire dată unei variabile de tip clasă**

Soluţionarea unei probleme implică izolarea ei de realitate. Se numeşte abstractizare procesul prin care se elimină detaliile realităţii fiind păstrate numai acele aspecte ce se consideră a fi relevante pentru soluţionarea problemei. Prin procesul de abstractizare se obţine un model al realităţii. Activitatea de trecere de la acest model la universul sistemului de calcul folosit pentru în scopul obţinerii rezultatelor poartă denumirea paradigmă sau metodologie (tehnologie) de programare.

În paradigma programării orientate pe obiect, modelarea realităţii de face prin conceptul de obiect. Obiectul este o abstractizare unei realităţi. Realitatea poate să fie palpabilă (ceva fizic) sau o idee (un concept) a cărei stare trebuie reprezentată. Obiectul este o unificare între datele şi mulţimea operaţiilor ce pot fi efectuate cu acestea. Utilizatorul unui obiect nu trebuie să cunoască tipurile de date reprezentate în obiect ci doar operaţiile prin care acestea se pot modifica, se zice că reprezentarea internă a datelor este încapsulată (ascunsă) de exterior, dar poate fi manipulată prin operaţii specifice.

Clasa descrie un model sau un şablon de stare şi de comportament pentru obiecte. Definiţia unei clase constă în date (câmpuri) şi metode (proceduri de calcul). Clasa este un tip de dată definit de utilizator pe baza căreia se vor crea noi obiecte din respectiva clasă. Definiţia unei clase constă în:

- **modificatori de acces**: definesc vizibilitatea clasei în raport cu alte clase (`public`);
- `class`: cuvânt cheie care anunţă Java ca urmează un bloc pentru definirea unei clase;
- **câmpuri**: variabile sau constante care sunt folosite de obiectele clasei (`x` şi `y`);
- **constructori**: metode care controlează starea iniţială a oricărui obiect din clasă(`Punct()` şi `Punct(double abscisa, double ordonata)`);
- **metode**: funcţii care controlează valorile stocate în câmpuri (`setX()`,`setY()` `...`).

Exemplul următor defineşte o clasă cu numele Punct.

```
//Definitia unei clase
public class Punct {
```

```
//Campuri
    private double x;
    private double y;

//Constructori
    Punct() {
        setX(0);
        setY(0);
    }

    Punct(double abscisa, double ordonata) {
        setX(abscisa);
        setY(ordonata);
    }

// Metode
    public void setX(double abscisa) {
        x = abscisa;
    }

    public void setY(double ordonata) {
        y = ordonata;
    }

    public double x() {
        return x;
    }

    public double y() {
            return y;
    }

    public double distantaOrigine() {
        return Math.sqrt(x*x+y*y);
    }

     public String toString() {
        return "<" + x + "," + y + ">";
    }
}

public class Grafica {
    public static void main(String[] args) {
        Punct p1; //declararea var. obiect p1 de tipul clasa Punct
        Punct p2 = new Punct(-1,7); //decl. + creare + initializare

        p1 =  new Punct(); //creare + initializare obiecte p1

        System.out.println("p1 = " + p1);
        System.out.println("p2 = " + p2);

        p1.setX(12);
        p2.setY(13.345);

        System.out.println("p1 = " + p1.toString());
        System.out.println("p2 = " + p2);

    }
}
```

Rezultate:

```
p1 = <0.0,0.0>
p2 = <-1.0,7.0>
```

```
p1 = <12.0,0.0>
p2 = <-1.0,13.345>
```

C2 - Întrebari

1. Care sunt tipurile simple numerice din Java?
2. Dati exemple de literalii reali din Java.
3. Ce este clasa si cum se creeaza obiectele din clase?

BIBLIOGRAFIE

1. http://www.oracle.com/technetwork/java/javase/documentation/index.html

2. http://docs.oracle.com/javase/6/docs/

3. Stefan Tanasa, Cristian Olaru, Stefan Andrei, Java de la 0 la expert, Polirom, 2003, ISBN: 973-681-201-4.

4. Herber Schild, Java 2 - The Complete Reference, Fourth Edition, Osborne, 2001, ISBN: 0-07-213084-9.

5. Deitel H.M., Deitel P. J., Java - How to programm, Fith Edition, Prentice Hall, 2003, ISBN: 0-13-120236-7.

6. http://www.east.utcluj.ro/mb/mep/antal/downloads.html