Teorie_Fișa_2A

2. Tipuri de date primitive

În limbajul Java există două categorii de tipuri de date:

- tipuri primitive: tipurile numerice, tipul caracter şi tipul logic;
- tipul referință: reține o referință (adresă) către o instanță a unei clase (obiect).

Limbajul Java defineşte opt tipuri de date primitive. Acestea se pot clasifica în:

- tipuri întregi: `byte`, `short`, `int` şi `long`;
- tipuri reale: `float` şi `double`;
- tipul caracter: `char;`
- tipul logic: `boolean`.

4.1. Tipuri întregi

Aceste tipuri sunt reprezentate în memorie ca numere cu semn (reprezentare în complement față de 2). Limbajul Java nu oferă tipuri întregi fără semn.

Tipul	Octeți alocați	Acoperirea
`byte`	1	-128 până la 127
`short`	2	-32768 până la 32767
`int`	4	-2147483648 până la 2147483647
`long`	8	-9223327036854755808L până la 922332703685475807L

Tipul **`byte`** este folosit, de obicei, pentru efectuarea unor operații la nivel de bit sau la prelucrarea la nivel scăzut (*low-level*) a fluxurilor (*stream* – fişiere, socket-uri, etc.).

Tipul **`short`** este deseori folosit în locul tipului `int`, mai ales din motive de optimizare a utilizării memoriei.

Tipul **`int`** este cel mai des folosit, având avantajul că are o acoperire suficient de mare, iar numărul de octeți folosiți pentru memorare este de două ori mai mic decât tipul următor (`long`).

Tipul **`long`** este mai rar folosit, înlocuind tipul `int` doar în cazul depăşirii acoperirii. Literalii întregi de tip `long` vor avea sufixul *L*; de exemplu: `long l = 23568912457801112L`.

În mod implicit, literalii întregi sunt de tip `int`. Java suportă utilizarea literalilor întregi reprezentați şi în bazele de numerație hexazecimal şi octal.

Utilizarea literalilor în bazele de numerație hexazecimal şi octal

```
// numărul 10 scris în hexazecimal

int example_hexa = 0xA;


// numărul 8 scris în sistemul octal (folosirea acestei baze de numeraţie
// nu este recomandată dată fiind forma confuză de declarare)

int example_octa = 010;
```

4.2. Tipuri reale

Numerele reale sunt reprezentate în limbajul Java ca numere cu semn (reprezentare în complement față de 2, virgulă mobilă simplă sau dublă precizie). Limbajul Java nu oferă tipuri reale fără semn.

Există două tipuri de numere reale reprezentate în virgulă mobilă: **`float`** (simplă precizie) şi **`double`** (dublă precizie). Numerele iraționale (cum ar fi π sau $\sqrt{2}$) sunt trunchiate, putându-se lucra doar cu o aproximare a lor.

Este recomandată folosirea tipului `float` numai când viteza la care trebuie să ruleze aplicația trebuie să fie foarte mare (sau memoria folosită de aceasta trebuie să fie foarte mică).

În mod implicit literalii numere reale sunt reprezentați ca valori `double`. Literalii de tip `float` poartă sufixul „F”. De exemplu:

```
// declarare de float
float example_float = 5.505F;
```

Caracteristicile celor două tipuri este prezentat în tabelul următor:

Tipul	Octeți alocați	Acoperirea
`float`	4	Aproximativ: ± 3.40282347E+38F (6-7 zecimale)
`double`	8	Aproximativ: ± 1.79769313486231570E+308 (13-14 zecimale)

4.3. Tipul caracter

Utilizarea caracterelor în Java se face cu ajutorul tipului **`char`**.

Tipul caracter utilizează pentru stocare doi octeți ce reţin codul *Unicode* al caracterului.

Avantajul folosirii standardului *Unicode* este acela că acoperirea este de 65535 de caractere (valorile limită fiind `'\u0000'` şi `'\uFFFF'`), față de codul *ASCII* (*American Standard Codes for Information Interchange*) care are doar 128 de caractere sau de *ASCII extins* cu 256 de caractere. Deşi se poate utiliza orice caracter, afişarea acestuia depinde de sistemul de operare (caracterele *Unicode* necunoscute sistemului de operare nu vor putea fi afişate).

Literalii de tip caracter sunt scrişi între apostrofuri şi pot conține secvențe *escape* (secvențe de caractere care încep cu '\').

Declararea unei variabile de tip caracter cu inițializare:

```
// variabila ch va contine caracterul avand codul 1111
// (codul este un numar hexazecimal)

char ch = '\u1111';
```

Următorul tabel prezintă cele mai folosite secvențe escape:

Secvența escape	Valoarea Unicode	Denumire
\b	\u0008	Backspace
\t	\u0009	tab (TAB)
\n	\u000A	Linefeed (LF)
\r	\u000D	carriage return (CR)
\"	\u0022	ghilimele
\'	\u0027	apostrof
****	\u005C	backslash

4.4. Tipul logic

Tipul logic permite memorarea doar a două valori: *adevărat* (**`true`**) şi *fals* (**`false`**).

Desemnarea acestui tip, în limbajul Java, se face cu ajutorul cuvântului cheie **`boolean`**.

Tipul `boolean` este folosit, în prinicpal, pentru evaluarea condițiilor logice. Introducerea sa a eliminat neclarităţile din limabjele C şi C++, unde evaluarea condițiilor se făcea folosind întregi.

Nu se poate face conversie între tipurile `boolean` şi cele întregi.

4.5. Conversii între tipurile de date primitive

Tabelul următor sintetizează conversiile permise între tipurile de date primitive:

	byte	short	int	long	char	float	double
byte	=	*	*	*	!	≈	*
short	!	=	*	*	!	≈	*
int	!	!	=	*	!	≈	*
long	!	!	!	=	!	≈	≈
char	!	!	*	*	=	≈	*
float	!	!	!	!	!	=	*
double	!	!	!	!	!	!	=

= - acelaşi tip;
! - conversia nu se poate face;
***** - conversia se poate face;
≈ - conversia se poate face, dar cu pierdere de precizie.

4.6. Variabile

Ca în orice limbaj de programare, tipurile de date sunt utilizate, în principal, pentru descrierea *variabilelor* şi stabilesc felul datelor pe care le pot memora, cantitatea de memorie necesară şi valorile ce le pot fi asociate.

Mai concret, o variabilă este un spațiu de memorie destinat stocării unei valori într-un format corespunzător tipului declarat şi căreia i se ataşează o etichetă – numele variabilei. Variabilele îşi pot schimba valoarea oricând pe parcursul programului.

Pentru *declararea* variabilelor, limbajul Java utilizează sintaxa:

Sintaxă utilizată: `id_tip id_var1[, id_var2 [, ...]];`

De exemplu, se poate scrie:

```
int  variabla;
boolean bool;
```

Se mai pot face şi declarări de forma:

```
int v1, v2;
```

dar această modalitate nu este indicată.

Deseori este recomandat ca variabila declarată să conțină şi o valoare inițială:

```
int v1 = 10;
```

Astfel, se poate spune, pe scurt, că *definirea* unei variabile reprezintă *declararea* ei, la care se adaugă operația de *inițializare*.

Declararea variabilelor se poate face oriunde pe parcursul programului. De asemenea, variabilele se pot redeclara, cu restricția ca redeclararea să nu se facă în acelaşi bloc cu prima declarație (deci să nu se afle în acelaşi domeniu de declarație – *scope*).

Referitor la numele variabilelor se recomandă denumirea lor în concordanță cu semnificația datelor pe care le reprezintă. Această măsură conferă o claritate sporită sursei şi minimizează timpul necesar efectuării unor corecturi sau modificări.

De asemenea, numele unei variabile nu poate începe cu o cifră; ele pot începe cu orice literă sau caracterul *underscore* ('_').

4.7. Constante

Definirea constantelor în Java se face prefixând definirea unei variabile cu **`final`**. Folosirea constantelor este indicată când programatorul vrea ca valoarea unei variabile să nu poată fi schimbată sau când se doreşte stabilirea unor aliasuri. Imaginați-vă că într-un program este necesară folosirea de mai multe ori a constantei matematice π. Acest lucru se poate face elegant făcând o declarare de forma (exemplul este pur didactic; în practică, clasa `java.lang.Math` poate oferi valoarea acestei constante):

```
final double PI = 3.14159;
```