Tema: Tipet e të dhënave në Java
(Data Types in Java)

Programuesja

Tabela e përmbajtjes

Programuesja

Përshëndetje të gjithëve!

Në temën e sotme do të flasim për tipet e të dhënave në Java. Janë tepër të rëndësishme pasi çdo variabël duhet të deklarohet me një tip të caktuar, për t'u përdorur. Tipet e të dhënave specifikojnë madhësitë dhe vlerat e ndryshme që mund të ruhen në variabla.

Ekzistojnë dy kategori të tipeve të të dhënave në Java:

- **Tipet e të dhënave primitive** *(Primitive data types)*: Tipet e të dhënave primitive përfshijnë *boolean, byte, char, short, int, long, float* dhe *double*. Përdoren për të ruajtur një vlerë të vetme.

- **Tipet e të dhënave jo-primitive** *(Non-primitive data types / Reference data typres)*: Tipet e të dhënave jo-primitive krijohen nga programuesit. Ato nuk janë të paracaktuara në Java si tipet e të dhënave primitive. Këto tipe të dhënash përdoren për të ruajtur një grup vlerash. Përfshijnë: *Class, Object, String, Array, Interface.*

Tipet e të dhënave primitive

Këto janë tipet më themelore të të dhënave në Java. Të dhënat primitive janë vlera të vetme dhe nuk kanë aftësi të veçanta.

> ***Java është një gjuhë programimi statically-typed, që do të thotë, të gjitha variablat duhet të deklarohen para përdorimit të tyre. Kjo është arsyeja pse duhet të deklarojmë tipin dhe emrin e variablit.***

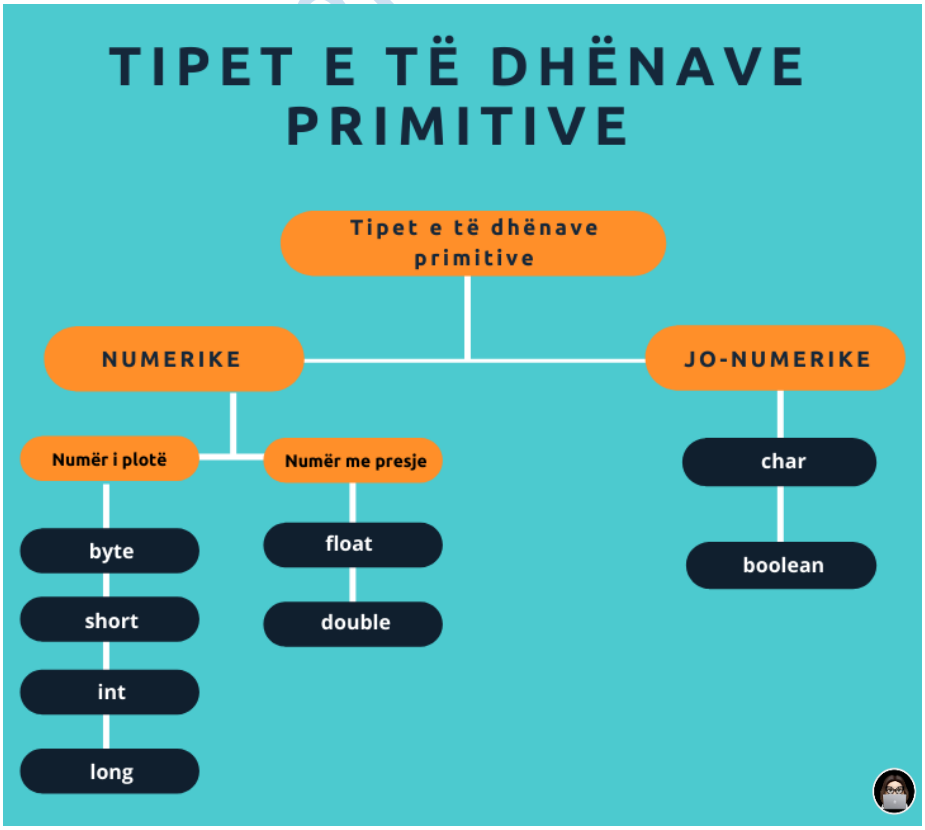

Fig. 1 Primitive data types

1. **boolean:** tipi i të dhënës *boolean* përfaqëson vetëm **1 bit** informacion me vlerën *true* ose *false*, por "madhësia (size)" e tij nuk është diçka e përcaktuar saktësisht pasi është virtual machine dependent (varet nga makina virtuale).

   Sintaksa: boolean variableName;

   Madhësia: virtual machine dependent

   Vlerat: true / false

   Vlera default: false

2. **byte:** tipi i të dhënave *byte* është një numër i plotë **8-bit** me shenjë, komplement i dyshit (two's complement). Ka një *vlerë minimale -128* dhe *vlerë maksimale 127*. Është i dobishëm për të kursyer memorien në *Arrays* të mëdha, ku kursimi i memories ka rëndësi. Kursen hapësirë, sepse një *byte* është 4 herë më i vogël se një *integer*. Gjithashtu, mund të përdoret në vend të *"int"* kur limiti i vlerave ndihmon në dokumentimin e kodit.

   Sintaksa: byte variableName;

   Madhësia: 1 byte (8 bits)

   Vlerat: -128 deri 127

   Vlera default: 0

3. **char:** tipi i të dhënave *char* është një karakter Unicode i vetëm **16-bit**. Gama e vlerave të tij qëndron midis '\ **u0000**' (ose 0) deri '\ **uffff**' (ose 65,535). *Char* përdoret për të ruajtur karaktere.

   Sintaksa: char variableName;

   Madhësia: 2 byte (16 bits)

   Vlerat: '\u0000' (0) deri '\uffff' (65535)

   Vlera default: '\u0000'

Pse madhësia e *char* është 2 byte në Java?

Në gjuhë të tjera si C/C ++ përdoren vetëm karaktere ASCII dhe për të përfaqësuar të gjithë karakteret ASCII mjaftojnë 8-bit, por Java përdor sistemin Unicode jo sistemin ASCII dhe për të përfaqësuar sistemin Unicode 8-bit nuk mjaftojnë për të përfaqësuar gjithë karakteret kështu që Java përdor **2 byte** për karaktere.

Unicode përdoret për të përfaqësuar të gjitha gjuhët njerëzore në botë. Është një bashkim i dhjetëra grupeve të karaktereve, të tilla si Latinishtja, Greqishtja, Arabishtja etj.

Programuesja

4. **short:** është një numër i plotë **16-bit** me shenjë, komplement i dyshit (two's complement). Njëlloj si byte, është i dobishëm për të kursyer memorie në *Arrays* të mëdha, ku kursimi i memories është me rëndësi.

   Sintaksa: short variableName;

   Madhësia: 2 byte (16 bits)

   Vlerat: -32,768 deri 32,767

   Vlera default: 0

5. **int:** është një numër i plotë **32-bit** me shenjë, komplement i dyshit. Përgjithësisht, *int* është tipi i preferuar i të dhënave kur krijohen variabla me vlerë numerike.

   Sintaksa: int variableName;

   Madhësia: 4 byte (32 bits)

   Vlerat: -2^{31} deri 2^{31} -1

   Vlera default: 0

   Që prej Java SE 8 mund ta përdorim int për të përfaqësuar një numër të plotë pozitiv 32-bit, i cili i ka vlerat në rangun [0, 2^{32} -1]. Për këtë përdorim klasën Integer, e cila përmban metoda si comapareUnsigned, divideUnsigned, parseUnsignedInt etj.

6. **long:** është një numër i plotë **64-bit**, komplement i dyshit. Ky tip i të dhënave kryesisht ruan të dhëna numerike me madhësi (size) më të madhe se *int*.

   Sintaksa: long variableName;

   Madhësia: 8 byte (64 bits)

   Vlerat: -2^{63} deri 2^{63} -1

   Vlera default: 0

   Që prej Java SE 8 mund ta përdorim long për të përfaqësuar një numër të plotë pozitiv 64-bit, i cili i ka vlerat në rangun [0, 2^{64} -1]. Për këtë përdorim klasën Long, e cila përmban metoda si comapareUnsigned, divideUnsigned etj.

7. **float:** tipi i të dhënave *float* është numër me presje single-precision **32-bit** IEEE 754. Përdorim float në vend të double nëse duam të kursejmë memorie në Array të mëdha me numra me presje.

   Sintaksa: float variableName;

   Madhësia: 4 byte (32 bits)

   Vlerat: deri në 7 shifra dhjetore

   Vlera default: 0.0F

   Ky lloj i të dhënave nuk duhet të përdoret kurrë për vlera preçize, të tilla si valuta. Për këtë, do të duhet të përdorim klasën java.math.BigDecimal.

8. **double:** tipi i të dhënave *double* është numër me presje double-precision **64-bit** IEEE 754. Për vlerat dhjetore, ky tip të dhënash është në përgjithësi zgjedhja default. Siç u përmend më lart, ky tip i të dhënave nuk duhet të përdoret kurrë për vlera preçize, siç është valuta.

   Sintaksa: double variableName;

   Madhësia: 8 byte (64 bits)

   Vlerat: deri në 16 shifra dhjetore

   Vlera default: 0.0

Tipet e të dhënave jo-primitive

Tipet e të dhënave jo-primitive i referohen objekteve dhe prandaj quhen tipe reference (reference types).

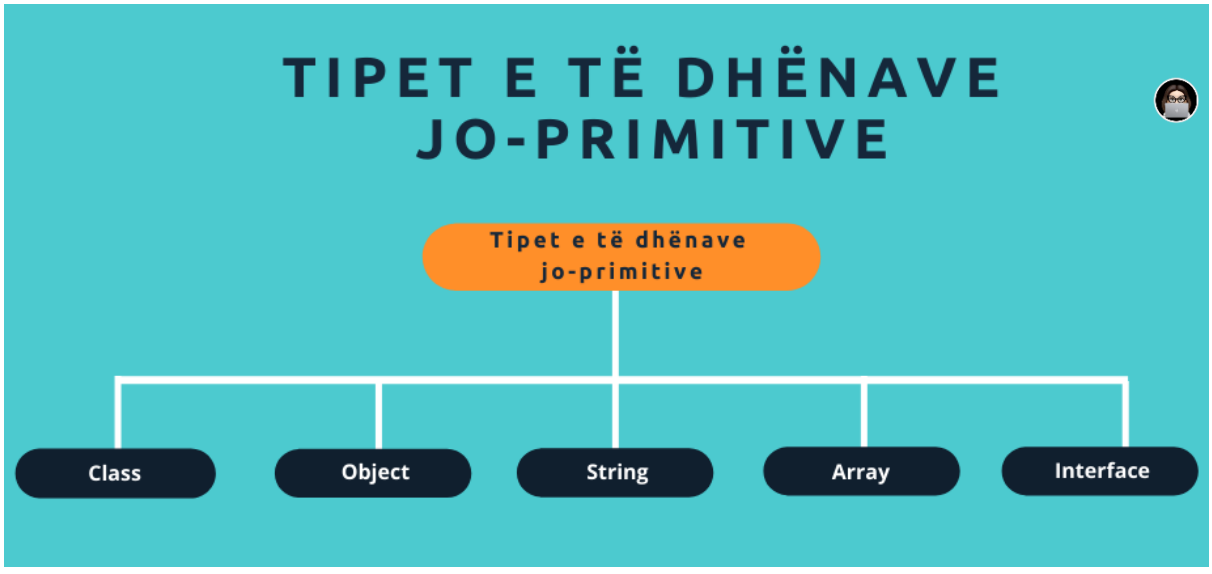

Fig. 2 Non-primitive data types

Për shembull, marrim një *array*. Në një array mund të ruajmë një group me vlera. Një shembull tjetër është klasa, e cila mund të ruaj vlera të ndryshme *(kujtoni shembullin me Shpendët në temën e OOP)*. Kur definojmë një variabël të tipit jo-primitiv, ajo referon një vendndodhje në memorie ku të dhënat janë ruajtur në Heap Memory. Pra, referon një memorie ku një objekt është vendosur.

Dy pika kyçe:

a. Vlera default e variablave të tipit jo-primitiv është **null**.
b. Kurdo që do i kaloni një tip të dhëne jo-primitive një metode, në fakt po kaloni një adresë të atij objekti ku të dhënat janë ruajtur.

1. **Arrays:** Një *array* në Java është një objekt, i cili përdoret për të ruajtur shumë variabla të të njëjtit lloj. Këto variabla mund të jenë tipe të të dhënave primitive ose jo-primitive. Një element specifik në një *array* aksesohet nga indeksi i tij. *Arrays* në Java mund të vendosen si parametra të metodave, variabla lokale dhe fusha statike. Madhësia e një *array* duhet të specifikohet nga një vlerë **int**, jo long ose short. Superclass-a direkte e një tipi array është Object.

> ***Gjithmonë indeksimi i një array fillon nga 0.***

2. **Strings:** Një *string* është një objekt që përfaqëson një sekuencë karakteresh. Klasa *java.lang.String* përdoret për të krijuar një objekt String. Diferenca mes një array karakteresh dhe një string-u në Java është se String është dizenjuar që të mbajë një sekuencë karakteresh në një variabël të vetme, një *array* karakteresh është një koleksion entitetesh të tipit char.

3. Përsa i përket Objektit, Klasës dhe Ndërfaqes (Interface) janë të shpjeguara me detaje në kapitullin e OOP.

Ndryshime mes tipeve të të dhënave Primitive dhe Jo-primitive në Java:

- Tipet e të dhënave primitive janë të paracaktuara në Java kurse ato jo-primitive krijohen nga programuesit (nuk janë të paracaktuara).
- Në tipet e të dhënave primitive, variablat mund ruajnë vetëm 1 vlerë çdo herë, kur në ato jo-primitive mund të ruajmë disa vlera (të të njëjtit tip ose të ndryshme)
- Të gjitha të dhënat për variablat e tipeve primitive ruhen në **memorien stack**, kurse për tipet jo-primitive stack-u ka një pointer te objekti në **memorien heap**.
- Tipet jo-primitive mund të përdoren për të thirrur metoda që kryejnë veprime të caktuara, kurse tipet primitive nuk munden.
- Një tip primitiv ka gjithmonë një vlerë (përmendëm vlerat *default* më sipër), kurse tipet jo-primitive mund të jenë **null**.