Datové typy, generika, výčtové datové typy, struktura, anotace, operátory

Datové typy

Datový typ je způsob definování významu a obsahu hodnoty proměnné nebo konstanty. Každý datový typ má různé funkce a operace, které se s nimi dají provádět.

Začínají malým písmenem.

Datové typy se dělí na dvě skupiny – primitivní a neprimitivní.

Primitivní

Primitivní datové typy jsou takové typy, které mají definovanou velikost a paměťovou kapacitu.

byte

Byte je datový typ, který uchovává číslo v rozpětí mezi -128 do 127.
Jeho paměťová kapacita 8 bitů - jeden byte…
Základně je nastaven na 0.

short

Short je datový typ, který uchovává číslo v rozpětí -32 768 do 32 767.
Jeho paměťová kapacita je 16 bitů - dva byte.
Základně nastaven na 0

int

Int je datový typ, který uchovává číslo v rozpětí -2 147 483 648 do 2 147 483 647.
Jeho paměťová kapacita je 32 bitů - čtyři byty.
Základně nastaven na hodnotu 0

long

Long je datový typ, který uchovává číslo v rozmezí
-9,223,372,036,854,775,808 do 9,223,372,036,854,775,807
Má paměťovou kapacitu 64 bitů – 8 bytů
Základně nastaven na hodnotu 0L – L indikuje long.

float

Float je datový typ, který uchovává desetinná čísla. Je vhodný pro zapsání šesti až sedmi desetinných čísel.
Má paměťovou kapacitu 32 bitů – 4 byty.
Základně nastaven na hodnotu 0f – f indikuje float.

double

Float je datový typ, který uchovává desetinná čísla. Je vhodný pro zapsání až patnácti desetinných čísel
Má paměťovou kapacitu 64 bitů – 8 bytů.
Základně nastaven na hodnotu 0d – d indikuje double.

boolean

Boolean je datový typ, který uchovává pouze dvě informace – true a false. V paměti je zapsána ovšem 0 nebo 1, ne „true“ nebo „false“.
Má paměťovou kapacitu pouze jednoho bitu.

Char

Char je datový typ, který uchovává informaci o jednotlivém znaku / písmeně, nebo může obsahovat hodnotu ASCII tabulky.
Defaultně je nastaven na hodnotu '\u0000'.
Má paměťovou kapacitu 16 bitů - 2 bytů. Je to proto, že Java nepoužívá ASCII tabulku, ale Unicode.

Neprimitivní

Neprimitivní datové typy jsou také zvány referencové, jelikož odkazují na objekt. Kromě stringu jsou veškeré neprimitivní datové typy nedefinované a programátor si je musí vytvořit sám. („new“ keyword)

Neprimitivní datové typy začínají s velkým písmenem. Mohou také být využiti k volání metod nebo provádět operace.

Nemají pevnou paměťovou kapacitu

String

String můžeme vytvořit dvěma způsoby – Bez keywordu „new“ a s keywordem „new“. V prvním případě se bude jednat o String literal, v druhém případě o String objekt.

String funguje zcela stejně jako pole znaků – i proto se mu říká řetězec znaků.

Pokaždé, když vytváříme string bez „new“ keywordu, zkontrolujeme oblast v Heapu, která si říká String constant pool. Pokud již existuje string s takovýmto obsahem, nevytvoříme novou instanci, ale uděláme na něj již referenci. I proto se mnohem více využívají String literal.

Fun fact: Java má 13 různých konstruktorů, přes který můžete udělat nový string, včetně pole.

Array (pole)

Java Array je objekt, který nám umožňuje do něj ukládat elementy stejného typu. Objekty uložené v poli jsou ukládány vedle sebe i v paměti. Jedná se také o datovou strukturu - Způsob orgnanizace dat. Do té se řadí například i fronta, zásobník, množina, halda či binární strom.

Elementy jsou v poli ukládány pomocí indexů. První prvek ja uložen na nultém indexu, třetí na druhém, dvanáctý na jedenáctém atd. Počet prvků v poli můžeme zjistit pomocí vlastnosti .length.

Můžeme vytvořit i multidimenzionální pole pomocí uložení pole do pole. Pro použití Array nemusíme používat import.

Je rozdíl mezi délkou a velikostí. Velikost pole může být pevně definována, zatímco délka může být proměnlivá dle počtu elementů uvnitř.

Class

Třídy jako takové definuje uživatel. Třída většinou obsahuje nějakou metody a proměnné. Pomocí datového typu třídy můžeme vytvořit novou instanci.

Pokud máme například třídu Student, můžeme použít Student jako datový typ. Podobné je to i s importy typu ArrayList

Generika

Generika nastoupila ve verzi Javy 5. Obecně nám umožňuje do určitých kolekcí vkládat prvky stejného datového typu, jako například pouze integery, pouze double nebo pouze objekty třídy Student.

Výhoda generiky je mimo jiné i to, že nemusíme při získání určité proměnné či objektu z kolekce přetypovávat, jak tomu do verze 5 bylo. Pokud jsme měli totiž list všeho, při získání prvku jsme museli přetypovat získaná data z kolekce na ty, které jsme potřebovali. Zároveň, kontrolování, jestli jsou prvky v kolekci stejného typu je chyba v kompilačním čase a ne v běžícím čase.

Příklad kolekce

```
List<String> list = new ArrayList<String>();

list.add("ahoj");

list.add(4);//Error v kompilačním čase
```

Obecný syntax

```
ClassOrInterface<Type>
```

Výčtové datové typy

V Javě je znám pouze jeden výčtový datový typ – Enum.

Obsahuje fixní set konstant, může být použit třeba pro dny v týdnu, měsíce, barvy atd. Podle Java naming convention, všechny by měli začínat velkým písmenem. Stejně jako třídy, enum umožňuje vytvořit vlastní datový typ.

Výhodami je jednoduché použití ve switchích, možnost mít vlastní konstruktory, metody i pole a mohou implementovat Interface.

Příklad Enumu, ve kterém je možné hodnotám přiřadit proměnná.

```
class EnumExample4{
enum Season{
WINTER(5), SPRING(10), SUMMER(15), FALL(20);

private int value;
private Season(int value){
this.value=value;
}
}
public static void main(String args[]){
for (Season s : Season.values())
System.out.println(s+" "+s.value);

}}
```

Anotace

Anotace jsou značky, metadata, která jsou vkládaná do zdrojového kódu a jsou převážně určené pro různé nástroje, se kterými program pracuje. Anotace mohou být zpracovány v třech různých etapách

1.) Při práci se zdrojovým kódem, kterou například využívají překladače a nebo javadoc – Dokumentační nástroj.
2.) Před spuštěním programu v souborech tříd – například nástroje, které připravují instalaci.

3.) Při běhu programu v souborech tříd.

Anotace se vyskytují u deklarací tříd, metod, atributů a podobají se modifikátorům. Všechny anotace začínají znakem zavináče - @

Možnosti anotací jsou velmi široké, jsou ale určeny převážně pro pokročilejší uživatele.

Základní anotace

@Override řekne kompilátoru, aby použil method overriding. To může způsobit i kompilační problémy, pokud metoda není nalezena v třídě rodiče.

@Deprecated označí metodu jako zastaralou. Způsobí warning, pokud je metoda použita.

@SupressWarnings řekne kompilátoru, aby ignoroval a potlačil varování, které jsou v kódu.

@SafeVarargs – Řekne kompilátoru, aby ignoroval varování i pro metody, které onu metodu volají.

@FunctionalInterface – Specifikuje, že deklarace typu je úmyslně funkční interace

Operátory

Operátoy v Javě nám umožňují provádět různé operace na hodnotách a nebo proměnných

Je dohromady osm druhů

Unární – Mají za úkol odečíst či přičíst jedna a provádět negace

Aritmetické – Jsou používány k provádění aritmetických operací

Shift – Používány k posouvání všech bitů v hodnotě do druhého čísla

Relační – Zjišťuje vztah mezi dvěmi proměnnými.

Bitwise – Provádění operací s jednotlivými bity v čísle.

Logické – Používány k provádění AND, OR a NOT logických operací

Ternární – Ternární operátor nahrazuje if, then a else v jednom řádku

Přiřazovací – Přiřazujeme proměnné hodnotu

Operator Type	Category	Precedence
Unary	postfix	`expr++ expr--`
	prefix	`++expr --expr +expr -expr ~ !`
Arithmetic	multiplicative	`* / %`
	additive	`+ -`
Shift	shift	`<< >> >>>`
Relational	comparison	`< > <= >= instanceof`
	equality	`== !=`
Bitwise	bitwise AND	`&`
	bitwise exclusive OR	`^`
	bitwise inclusive OR	`|`
Logical	logical AND	`&&`
	logical OR	`||`
Ternary	ternary	`? :`
Assignment	assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`