Andmetüübid Javas

Java on staatiliselt tüübitud keel

- i.k. statically typed
- Staatilise analüüsiga on võimalik osaliselt kontrollida programmi õigsust
- Andmetüübid, meetodite nimed, meetodite argumendid

Staatilise analüüsi puudumine

```
def add(a, b):
    return a + b


print(add(1, 2))

if flag:
    print(addd(1))
```

Staatiline analüüs

```
String name = "Alice";

name = 1;  // ei kompileeru!

long l = 1;

int i = l; // ei kompileeru!
```

Kommentaarid

- // ja /* ... */

```
// x = 1;

/*
x = 1;
y = 2;
z = 3;
*/
```

boolean

- true/false

```
boolean flag = true;
```

char

```
char c = 'a';

String s = 'hello'; // ei kompileeru!
```

String

```
String s = "Hello!";

System.out.println(s.toUpperCase());
```

String

```
String s = """
           multi line
           string""";

System.out.println(s);

// multi line
// string
```

Täisarvud (int)

- 32 bitti
- 2^32 (~4 miljardit) kombinatsiooni
- väärtused -2 miljardit kuni 2 miljardit

Overflow

```
int i = 1_500_000_000;


System.out.println(i); // 1500000000

System.out.println(i + i); // -1294967296
```

Täisarvud (long)

- 64 bitti
- 2^64 kombinatsiooni

```
System.out.println(Long.MIN_VALUE);
```

-9223372036854775808

```
System.out.println(Long.MAX_VALUE);
```

9223372036854775807

Ujuvkoma arvud (float)

- 32 bitti
- 2^32 (~4 miljardit) kombinatsiooni
- Koosneb kahest osast: mantiss ja eksponent

Ujuvkoma arvud Püütonis

```
print(6 * 0.1)  # 0.6000000000000001
```

Analoogia

1	2	3	4	10^-2

= 12,34

1	2	3	4	10^-5

= 0,01234

1	2	3	4	10^2

= 123400

Ujuvkoma arvud (float)

```
System.out.println(Float.MIN_VALUE);
```

1.4E-45

```
System.out.println(Float.MAX_VALUE);
```

3.4028235E38

```
System.out.println(Float.MAX_VALUE == Float.MAX_VALUE - 1);
// true
```

Ujuvkoma arvud (double)

- 64 bitti

Teisendamine

```
int x = 1;
int y = 2;

System.out.println(x / y); // 0
```

Casting

```
int i = 0;

long l = i; // kompileerub

i = l; // ei kompileeru!
```

Casting (implicit)

- byte -> int -> long -> float -> double

Casting (implicit)

- byte -> int -> long -> float -> double

```
int x = 1234567890;

float y = x;

System.out.println(y); // 1.23456794E9
```

Casting (explicit)

```
long l = 0;

int i = (int) l;
```

Casting (explicit)

```
int x = 1;
int y = 2;

System.out.println(x / y); // 0

System.out.println((float) x / y); // 0.5

System.out.println(Float.valueOf(x) / y); // 0.5
```

Casting (explicit)

```
double x = 1;
int y = 2;

System.out.println(x / y); // 0

System.out.println((float) x / y); // 0.5

System.out.println(Float.valueOf(x) / y); // ei kompileeru!
```

Demo: cast vs valueOf()

Casting

- Kui kuidagi ilma ei saa
- Võtan kompilaatorilt vastutuse ära
- Teen koodi raskemini loetavaks

Väärtuse puudumine

```
String firstName = "Alice";

String lastName = "Smith";

int weight = -1;
```

Väärtuse puudumine

```
int age;

age = 18;

System.out.println(age); // 18
```

Väärtuse puudumine

```
int age; // ok

System.out.println(age); // compilation error
```

null (null viit)

- null - väärtus puudub

```
String name = null;

Person alice = null;

int x = null; // ei kompileeru!

Integer i = null; // ok

Long l = null; // ok
```

Primitiivid vs objektid

```
int age = 18;

String name = "Alice";
```

Primitiivid vs objektid

```
String name = "Alice";

Integer age = 18;

Integer weight = null;
```

Objektid

- Integer, Long, Float, Double, Boolean, Character

Primitiivid vs objektid

```
int x = 1;

x.hashCode(); // ei kompileeru!

Integer y = x; // ok - autoboxing.

System.out.println(y.hashCode());
```

Demo: primitiivid ja jõudlus

Massiiv (array)

```
int[] numbers = {1, 2};

String[] names = {"Jill", "Jack", "Joe"};
```

```
System.out.println(numbers[0]); // 1

numbers[1] = 9;

System.out.println(numbers[3]); // error!
```

Massiiv (array)

```java
int[] numbers = {1, 2};

System.out.println(numbers); // [I@1cd072a9

System.out.println(Arrays.toString(numbers)); // [1, 2]
```

Massiiv (array)

```
int[] numbers = {1, 2};

numbers = new int[] {1, 2, 3};
```

Massiiv (array)

```
boolean[][] matrix = new boolean[4][5];

boolean[][][] cube = new boolean[5][5][5];
```

Kontrollstruktuurid

If-else

```java
if (x == 1) {
    System.out.println("one");
}
```

```python
if "":
    print("empty")
else:
    print("not emtpy")
```

If-else

```
if (x == 1) {
    System.out.println("one");
} else if (x == 2) {
    System.out.println("two");
} else {
   System.out.println("not one nor two");
}
```

While

```
int i = 0;

while (i < 10) {
    System.out.println(i);

    i++; // i += 1
}
```

For

```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

Muutuja skoop

```
int i = 0;
while (i < 10) {
    System.out.println(i);

    i++; // i += 1
}

System.out.println(i); // 10
```

Muutuja skoop

```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}

System.out.println(i); // ei kompileeru!
```

Muutuja skoop

```
if (1 == 1) {
    int i = 1;
}

System.out.println(i); // ei kompileeru!
```

Foreach

```
int[] numbers = {1, 2, 3};

for (int number : numbers) {
    System.out.println(number);
}
```

```
for (int i = 0; i < numbers.length; i++) {
    System.out.println(numbers[i]);
}
```

Operaatorid

==	võrdub
!=	ei võrdu
&&	konjunktsioon (nt. x && y)
||	disjunktsioon (nt. x || y)
!	eitus (nt. !x)

Demo

- Summa, keskmine

Idea otseteed

- sout + Enter -> System.out.println();
- fori + Enter -> for (int i = 0; i < ; i++) {
- iter + Enter -> for (Integer each : numbers) {
- ctrl + numpad / -> rea kommentaar (// ...)
- ctrl + shift + A -> otsi kõike

Idea otseteed

- ctrl + B -> näita meetodi deklaratsiooni
- ctrl + alt + Left -> navigeeri tagasi

null (null viit)

- null - väärtus puudub

```
Integer i = null;

Long l = null;

String s = null;

int x = null; // ei kompileeru!
```

null

```
sum("1", "2"); // ei kompileeru!

sum(1); // ei kompileeru!

sum(null, null); // ok
```

```
public static Integer sum(Integer a, Integer b) {
    return a + b;
}
```

null, kui märgend

```
String firstName = "Alice";

String lastName = "Smith";

Integer age = null;
```

null

- Tähistamaks väärtuse puudumist

```
public static Integer minimumElement(int[] integers) {
    Integer minimumElement = integers[0]; // ei sobi!

    for (...

        ...

    return minimumElement;
```

null

- Tähistamaks väärtuse puudumist

```
public static Integer minimumElement(int[] integers) {
    Integer minimumElement = null;

    for (...

        ...


    return minimumElement;
```

null

- Kui võimalik, võiks null väärtust vältida

Jõudlus: Java vs Püüton

```java
double start = System.currentTimeMillis();

for (int i = 0; i < 1e9; i++) {
}

System.out.println((System.currentTimeMillis() - start) / 1000);

// 1.0
```

Jõudlus: Java vs Püüton

```
import time

start = time.time()

i = 0
while i < 1e7:
    i += 1

print(time.time() - start)  # 1.8
```

Arvutüübid kokkuvõte

Tavad Javas (nimetamine)

```
String firstName = "Alice";

int age = 15;

public static String getFirstName() {
    ...
}

public static String getLastName()
{
    ...
}
```

Nimetamine

- i, myVariable

Nimetamine

```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

Nimetamine

```
int min = Constants.MINIMUM_INSTALLMENT_AMOUNT;
```

Nimetamine

- processInvoice(Invoice invoice)

Nimetamine

- Üllatuste vältimine

  Nt. getItems() ei tohiks andmeid muuta

Väide

- Kui kõigil programmi osadel (klassid, meetodid, muutujad) on selgelt arusaadavad nimed on programm hästi disainitud.

Koodi täitmise järjekord

```
method3(method1(), method2());

public static int method1() {
    System.out.println("2");
    return 2;
}

public static int method2() {
    System.out.println("1");
    return 1;
}

public static void method3(int x, int y) {
    System.out.println(x + y);
}
```

Stack Trace

Nimeruum

Kompileerimine ja nimeruum

Staatiline analüüs

```
sum("1", "2"); // ei kompileeru!

sum(1); // ei kompileeru!

sum(1, 2); // ei kompileeru!

sum(1L, 2L); // ok
```

```
public static long sum(long a, long b) {
    return a + b;
}
```

Termin: meetodi signatuur (method signature)

- Meetodi nimi
- Argumentide arv, tüübid ja järjekord

```
public int sum(int a, int b) {
    return a + b;
}
```

Meetodite ülelaadimine (overload)

```
public int sum(int a, int b) {
    return a + b;
}

public long sum(long a, long b) {
    return a + b;
}

public long sum(long a, long b, long c) {
    return a + b + c;
}
```

Autoboxing

```
int primitiveX = 1;

Integer objectX = primitiveX;

primitiveX = objectX;
```

Autoboxing

```
Integer x = 1;
Integer y = 2;

sum(x, y);

public static Integer sum(int a, int b) {
    return a + b;
}
```

Autoboxing

```
int primitiveX = 1;

Integer objectX = null;

primitiveX = objectX; // NullPointerException
```

Autoboxing

```
Integer x = 1;
Integer y = null;

sum(x, y); // NullPointerException

public static Integer sum(int a, int b) {
    return a + b;
}
```

Autoboxing

```
Boolean isFirst = null;

// ...

if (isFirst) { // NullPointerException
    System.out.println("First");
}
```