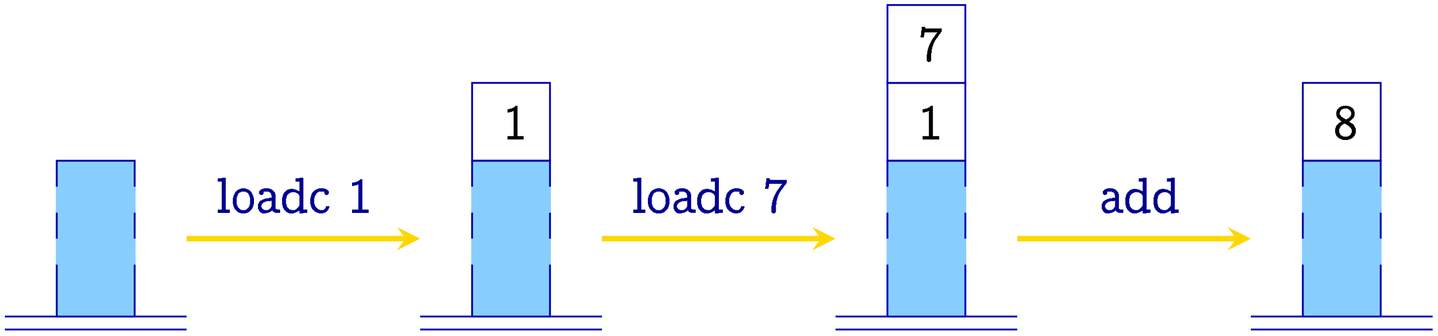
Java Virtuaalmasin

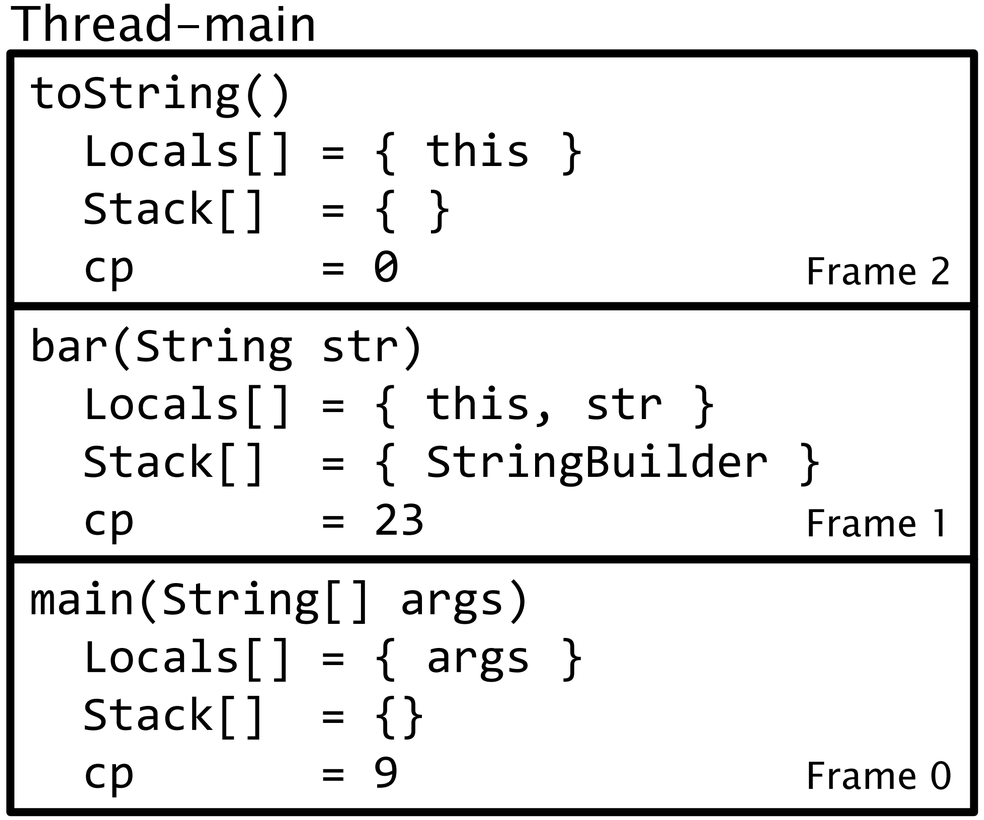
JVM tutvustus
Java baitkood
CMa ja JVM sarnasused ja erinevused

JVM on abstraktne masin

- JVM on samuti magasinipõhine abstraktne masin.
- Tal on sarnased andmestruktuurid ja käsud.
- Detailides on suured erinevused, aga üldpilt on ikkagi sama:



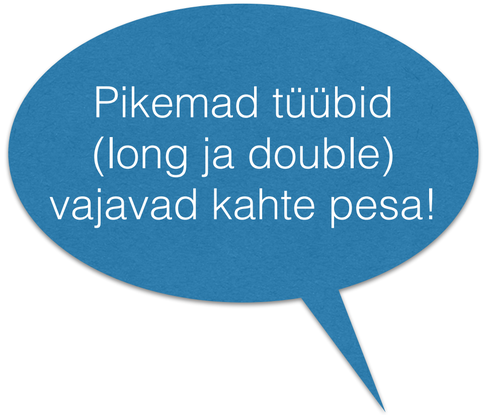
Andmestruktuurid



- Koodi jaoks on olemas *method area.*
- Igal lõimel on programmiloendur (PC) ja raamidega kutsemagasin.
- Raamis on meetodikutse lokaalsed muutujad, arvutamise magasin ja viit klassi konstantkogumile (*constant pool*).
- Objektid elavad kuhjas (*heap*).

JVM käsustik (Java baitkood) on tüübitud!

- Näiteks erienvate tüüpide liitmisel peab kasutama erinevaid käske:
  - `iadd`: lühikesed täisarvud (int, short, byte)
  - `ladd`: pikad täisarvud (long)
  - `fadd` ja `dadd`: ujukomaarvud (float ja double)
- Objektide korral kasutatakse prefiksit „a“.
- Massiivi jaoks on käsud kujul „iaload“ ja „daload“.



I	8–32 bit integer	(1 stack slot)
L	64 bit integer	(2 stack slots)
F	32 bit float	(1 stack slot)
D	64 bit float	(2 stack slots)
A	Objects	(1 stack slot)
?A	Arrays	(1 stack slot)

Konstantide käsud

- CMa lihtsa „loadc“ asemel on JVM’il üsna rikkalik valik käske.
- Näiteks konstandi 5 magasini laadimiseks on võimalik kasutada käske: `iconst_5`, `bipush`, `sipush` ja `ldc`.
- Selleks peab nüüd põhjalikumalt vaatama, mis asi on baitkood!?

```
public class Demo {

    public static void main(String[] args) {
        System.out.println(42);
    }

}
```

Java lähtekood

Kompileerime käsuga `javac Demo.java`.

Index Baidid Teksti kujul

```
00000000: cafe babe 0000 003c 001b 0a00 0200 0307 0004 0c00 0500 0601  .......<........
00000018: 0010 6a61 7661 2f6c 616e 672f 4f62 6a65 6374 0100 063c 696e  ..java/lang/Object...<in
00000030: 6974 3e01 0329 5609 0008 0009 0700 0a0c 000b 000c 0100  it>...()V.......
00000048: 106a 6176 612f 6c61 6e67 2f53 7973 7465 6d01 0003 6f75 7401  .java/lang/System...out.
00000060: 0015 4c6a 6176 612f 696f 2f50 7269 6e74 5374 7265 616d 3b0a  ..Ljava/io/PrintStream;.
00000078: 000e 000f 0700 100c 0011 0012 0100 106a 6176 612f 696f 2f50  ...............java/io/P
00000090: 7269 6e74 5374 7265 616d 0100 0770 7269 6e74 6c6e 0100 0428  rintStream...println...(
000000a8: 4929 5607 0014 0100 0444 656d 6f01 0004 436f 6465 0100 0f4c  I)V.....Demo...Code...L
000000c0: 696e 654e 756d 6265 7254 6162 6c65 0100 046d 6169 6e01 0016  ineNumberTable...main...
000000d8: 285b 4c6a 6176 612f 6c61 6e67 2f53 7472 696e 673b 2956 0100  ([Ljava/lang/String;)V..
000000f0: 0a53 6f75 7263 6546 696c 6501 0009 4465 6d6f 2e6a 6176 6100  .SourceFile...Demo.java.
00000108: 2100 1300 0200 0000 0000 0200 0100 0500 0600 0100 1500 0000  !.......................
00000120: 1d00 0100 0100 0000 052a b700 01b1 0000 0001 0016 0000 0006  .........*..............
00000138: 0001 0000 0001 0009 0017 0018 0001 0015 0000 0025 0002 0001  ...................%....
00000150: 0000 0009 b200 0710 2ab6 000d b100 0000 0100 1600 0000 0a00  ........*...............
00000168: 0200 0000 0300 0800 0400 0100 1900 0000 0200 1a              ................
```

Versioon 60 = Java 16.

Konstantide kogum

bipush 42

Java baitkood

Vaatame baitkoodi sisse käsuga `xxd Demo.class`.

(See näitab binaarfaili kuueteistkümnendsüüsteemis: iga bait on kaks sümbolit)

```
Compiled from "Demo.java"
public class Demo {
  public Demo();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: bipush        42
       5: invokevirtual #13                 // Method java/io/PrintStream.println:(I)V
       8: return
}
```

Konstruktori kood

Konstandi sisu

Viide kogumisse

Java assemblerkod

Koodi saab loetavamal kujul „disassembleri“ abil: `javap -c Demo.class`.

Samad käsud, aga …

Kirjapilt	Baitkood	baitide arv
iconst_5	“08”	1
bipush 5	“10 05”	2
sipush 5	“11 00 05”	3
ldc 5	“08 00 00 00 05 … 12 idx”	7

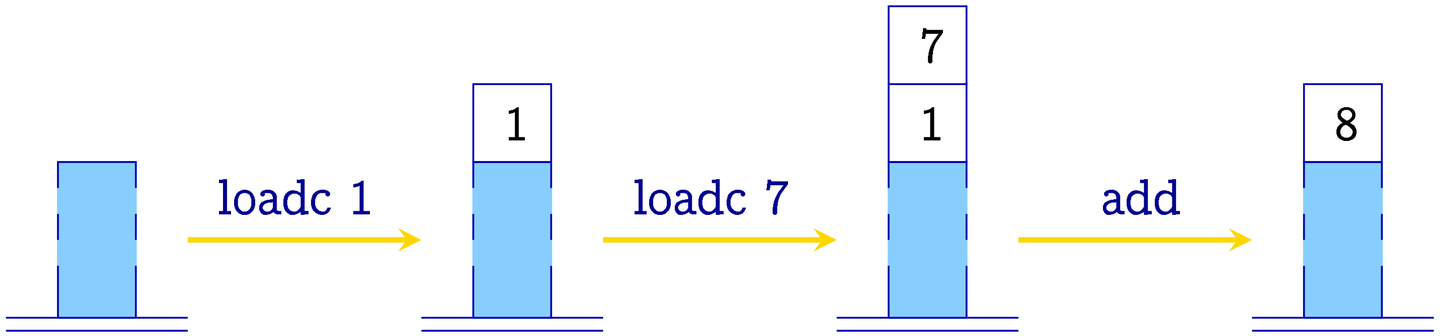
Märgend (int)
+ 4 baiti

1 bait (viit)

CMa versus JVM

loadc 1
loadc 7
add

iconst_1
bipush 7
iadd



Muutujad!

- Lokaalsete muutujate jaoks on nüüd eraldi massiiv.
- Aga samamoodi loome deklaratsioonide põhjal aadresskeskkonna ρ.
- CMa's oli üldine käsk `load` ja defineerisime `loada n` = `loadc n; load`.
- JVM'il on käsud `iload` ja `istore`, mis käituvad nagu `loada` ja `storea`.

Siiamaani kõik hästi…

```
loadc 7
load
loadc 1
sub
loadc 4
store
```

```
loada 7
loadc 1
sub
storea 4
```

```
iload 7
iconst_1
isub
istore 4
```

(x = y - 1)

Pergel peitub peensustes …

- Mõned sarnased käsud käituvad ka erinevalt (Soome hallitus).
- JVM load ja store käsud eemaldavad väärtused ära.
  (Kui väärtused vaja alles jätta, genereerib kompilaator `dup` käske.)
- Oluline erinevus meie jaoks on hüppekäsud!

Hüppamine!

- CMa'l on käsk `jump A` ja JVM'il on käsk `goto A`.
- Aga… `jumpz A` asemel on JVM'il kõik võrdlused ja hüped koos, näiteks `ifeq A`, `ifne A`, `ifle A`, `iflt A`, …
- Ja… lisaks sellistele, mis nulliga võrdlevad, on tal ka kahe elemendi võrdlemise käsud: `if_icmpeq A`, `if_icmpne A`, …

```
if (arg != null && arg.equals(…)) ...
```

Me ei pea nii keeruliselt tegema

- Java baitkood on võrdlemisi rikas keel, aga …
- Võime seda ignoreerida, näiteks võib kõikide konstantide korral täiesti muretult kasutada käsku `ldc`.
- Näiteks kõikide arvukonstantide jaoks koodi genereerimine:

```
protected void visitVoid(IntegerLiteral integerLiteral) {
    mv.visitLdcInsn(integerLiteral.getValue());
}
```

Edasi …

- Siin videos oli lühiülevaade JVM’ist, millest kõik võiksid teada.
- Kui tahad rohkem teada, saad vaadata Michael Rasmusseni täispikki loenguid.
- Kui tahad vähema vaevaga siiski kodutööga alustada, siis asendusõpetaja Vesal lindistas ka video selle kohta.
- (ja tema suureks üllatuseks oli punkti saamine päris lihtne!)

Kokkuvõte

- Tutvusime Java virtuaalmasina (JVM) andmestruktuuridega ja tema käsustikuga, mille nimeks on Java baitkood.
- Baitkood on Java klassfailides salvestatud.
  Nägime, kuidas seda saab vaadata.
- JVM töötab põhimõtteliselt nagu CMa, aga mõned käsud töötavad teistmoodi — koodi geneerimisel ei pea kõiki võimalusi kasutama.