Java baitkood
Programmeerimiskeeled

Toomas Römer ja Rein Raudjärv, 16.04.2008

Kava

- Virtuaalmasinad
- Kompilaatorid
- Ajalugu
- Baitkood vs masinkood
- Magasinipõhised keeled
- Baitkood detailsemalt
  - Klassi struktuur
  - Jooksutamise mudel
  - Instruktsioonid
- Demo
- Miks?

Java baitkood – CAFEBABE

```
0000:0000 CA FE BA BE 00 00 00 30 00 17 01 00 24 6F 72 67
0000:0010 2F 73 70 72 69 6E 67 66 72 61 6D 65 77 6F 72 6B
0000:0020 2F 6A 6D 78 2F 4A 6D 78 45 78 63 65 70 74 69 6F
0000:0030 6E 07 00 01 01 00 2F 6F 72 67 2F 73 70 72 69 6E
0000:0040 67 66 72 61 6D 65 77 6F 72 6B 2F 63 6F 72 65 2F
0000:0050 4E 65 73 74 65 64 52 75 6E 74 69 6D 65 45 78 63
0000:0060 65 70 74 69 6F 6E 07 00 03 01 00 11 4A 6D 78 45
0000:0070 78 63 65 70 74 69 6F 6E 2E 6A 61 76 61 01 00 06
0000:0080 3C 69 6E 69 74 3E 01 00 15 28 4C 6A 61 76 61 2F
0000:0090 6C 61 6E 67 2F 53 74 72 69 6E 67 3B 29 56 0C 00
0000:00a0 06 00 07 0A 00 04 00 08 01 00 04 74 68 69 73 01
0000:00b0 00 26 4C 6F 72 67 2F 73 70 72 69 6E 67 66 72 61
0000:00c0 6D 65 77 6F 72 6B 2F 6A 6D 78 2F 4A 6D 78 45 78
0000:00d0 63 65 70 74 69 6F 6E 3B 01 00 03 6D 73 67 01 00
0000:00e0 12 4C 6A 61 76 61 2F 6C 61 6E 67 2F 53 74 72 69
0000:00f0 6E 67 3B 01 00 2A 28 4C 6A 61 76 61 2F 6C 61 6E
```

Mida baitkoodiga teha saab?

- Baitkood jookseb Java Virtuaalmasinas (JVM)
- Baitkood on spetsifitseeritud Sun'i poolt
- Virtuaalmasinaid on rohkem kui slaid mahutab
  - Sun JVM
  - IBM J9 VM
  - Apple MRJ
  - BEA jRockit
  - Oracle JVM
  - MS JVM
  - Blackdown Java, CVM, Excelsior JET, IcedTea, Jamiga, JamVM, Jaos, Mika VM, NanoVM, SableVM, Wonka VM ...

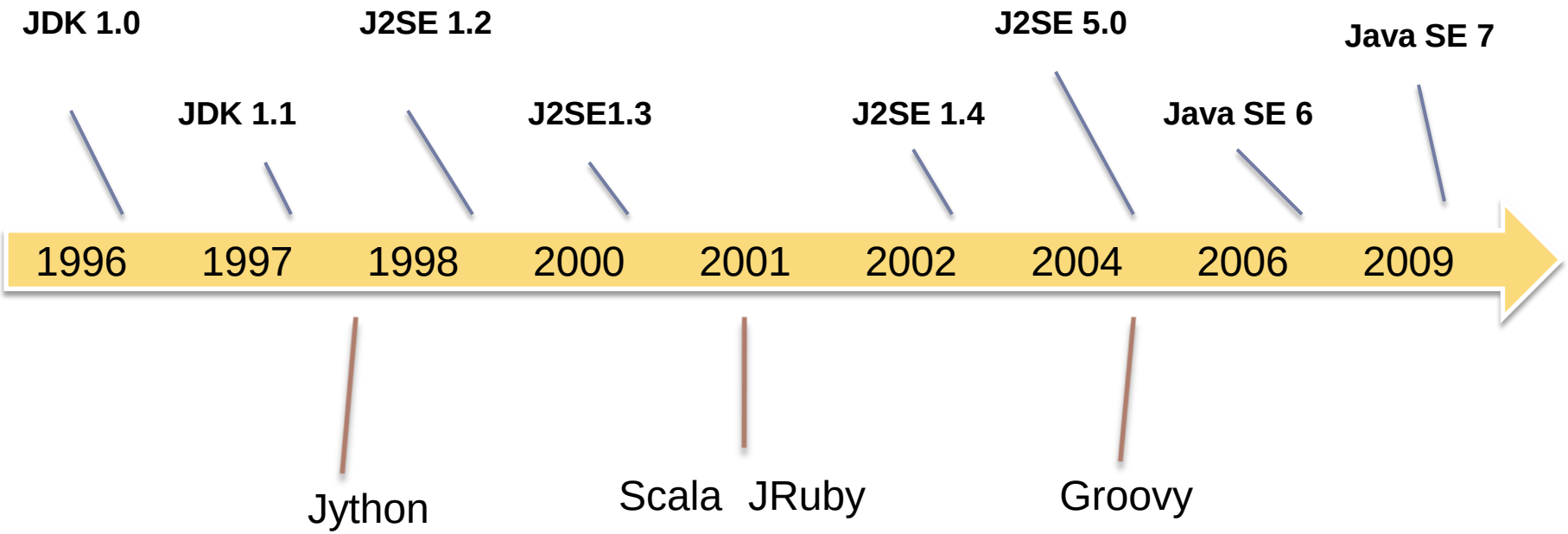
Kuidas baitkoodi saada?

- Kompilaatoriga ☺
  - Sun javac
  - Eclipse JDT
  - Gcc GCJ
  - IBM Jikes (arendus lõppenud)
  - Excelsior JET
  - JythonC (Python lähtekood kompileeritakse Java baitkoodi)

Millest baitkoodi saab?

- Java lähtekoodist ☺
- Scala (OO & funktsionaal keel JVM jaoks)
- Groovy (OO keel JVM jaoks)
- Ruby lähtekoodist (JRuby)
- Pythoni lähtekoodist (Jython)
- Ada (JGNAT ja AppletMagic kompileerivad Ada baitkoodi)

Ajalugu


JDK 1.0
JDK 1.1
J2SE 1.2
J2SE1.3
J2SE 1.4
J2SE 5.0
Java SE 6
Java SE 7
1996
1997
1998
2000
2001
2002
2004
2006
2009
Jython
Scala
JRuby
Groovy

Baitkood vs masinkood

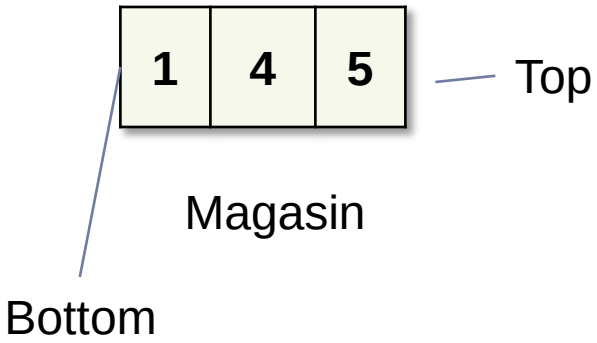
- JIT – Just in Time - baitkood
  - Baitkood jookseb virtuaalmasinas
  - Programmi jooksmise ajal optimiseerib instruktsioone
    - Puhverdamine
    - Masinkoodi kompileerimine
- AOT – Ahead of Time - masinkood
  - Baitkood kompileeritakse masinkoodiks
  - Gcc GCJ, Excelsior JET

Baitkood vs masinkood (2)

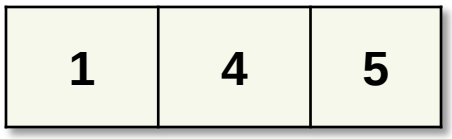
- JIT võib tihti peale olla efektiivsem kui AOT
  - Konkreetne CPU ja OP Süsteem
  - Runtime statistika
  - Globaalne koodi optimiseerimine
- JIT stardib aeglasemalt
- Muidugi saab AOT'ga konkreetse süsteemi jaoks kiire binaari kompileerida!

Magasini põhised keeled

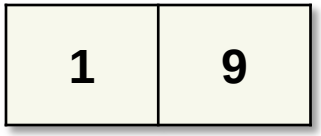
- Programmi mälu on organiseeritud 1...n magasinis
- Operaatorid "**söövad**" ja **lisavad** magasini väärtusi



Näide. Olgu meil "+" operaator
+(x,y) := x+y



Magasin enne tehet



Magasin pärast tehet

Magasini põhised keeled (2)

- Forth (1970)
- PostScript (1976)
- Java baitkood! (1996)
- Factor (2004)

JVM baitkood detailsemalt

```
L0 LINENUMBER 14 L0
NEW MyClassLoader
DUP
INVOKESPECIAL MyClassLoader.<init>() : void
ASTORE 2: myClassLoader
L1 (5)
LINENUMBER 15 L1
ALOAD 2: myClassLoader
LDC "sampleClass"
LDC "/"
LDC "."
INVOKEVIRTUAL String.replace(CharSequence,CharSequence) : String
ALOAD 0: this
INVOKEVIRTUAL AbstractExecuter.dump() : byte[]
INVOKEVIRTUAL MyClassLoader.defineClass(String,byte[]) : Class
ASTORE 3: c
L2 (15)
LINENUMBER 16 L2
ALOAD 3: c
LDC "main"
ICONST_1
ANEWARRAY Class
DUP
ICONST_0
LDC [Ljava/lang/String;.class
AASTORE
```

Kompileeritud klassi struktuur

Modifikaatorid, nimi, ülemklass, liidesed	
Lähtekoodi failinimi	
Välimise klassi viide	
Annotatsioonid	
Sisemine klass*	Nimi
Väli*	Modifikaatorid, nimi, tüüp
	Annotatsioonid
Meetod*	Modifikaatorid, nimi, tagastus- ja paremeetrite tüübid
	Annotatsioonid
	Kompileeritud kood

Konstruktor

```
public class InitTest {
  public InitTest() { throw new Error("viga"); }
  public static void main(String[] a) {
    new InitTest();
  }
}
```

```
Exception in thread "main" java.lang.Error: viga
at InitTest.<init>(InitTest.java:5)
at InitTest.main(InitTest.java:9)
```

Konstruktor on meetod nimega **<init>**

Staatiline “konstruktor”

```
public class StaticInitTest {
  static {
    if (1 == 1) throw new Error("viga");
  }
  public static void main(String[] a) {}
}
```

```
java.lang.Error: viga
at StaticInitTest.<clinit>(StaticInitTest.java:3)
Exception in thread "main"
```

Staatiline “konstruktor” on meetod nimega **<clinit>**

Sisemine klass

```
public class A {
  public void meetod() { throw new Error("viga"); }
  public class B { public B() { meetod(); } }
  public static void main(String[] a) {
    new A().new B();
  }
}
```

```
Exception in thread "main" java.lang.Error: viga
at A.meetod(A.java:3)
at A$B.<init>(A.java:6)
at A.main(A.java:9)
```

Sisemine klass on iseseisev klass nimega **välimineklass$sisemineklass**

Sisemine klass (2)

```
public class A {
  private void meetod() { throw new Error("viga"); }
  public class B { public B() { meetod(); } }
  public static void main(String[] a) {
    new A().new B();
  }
}
```

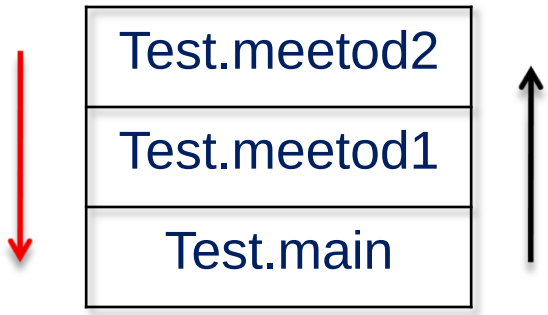
```
Exception in thread "main" java.lang.Error: viga
at A.meetod(A.java:2)
at A.access$0(A.java:2)
at A$B.<init>(A.java:3)
at A.main(A.java:5)
```

Sisemise klassi jaoks luuakse abimeetodid nimega **access$0, access$1** jne

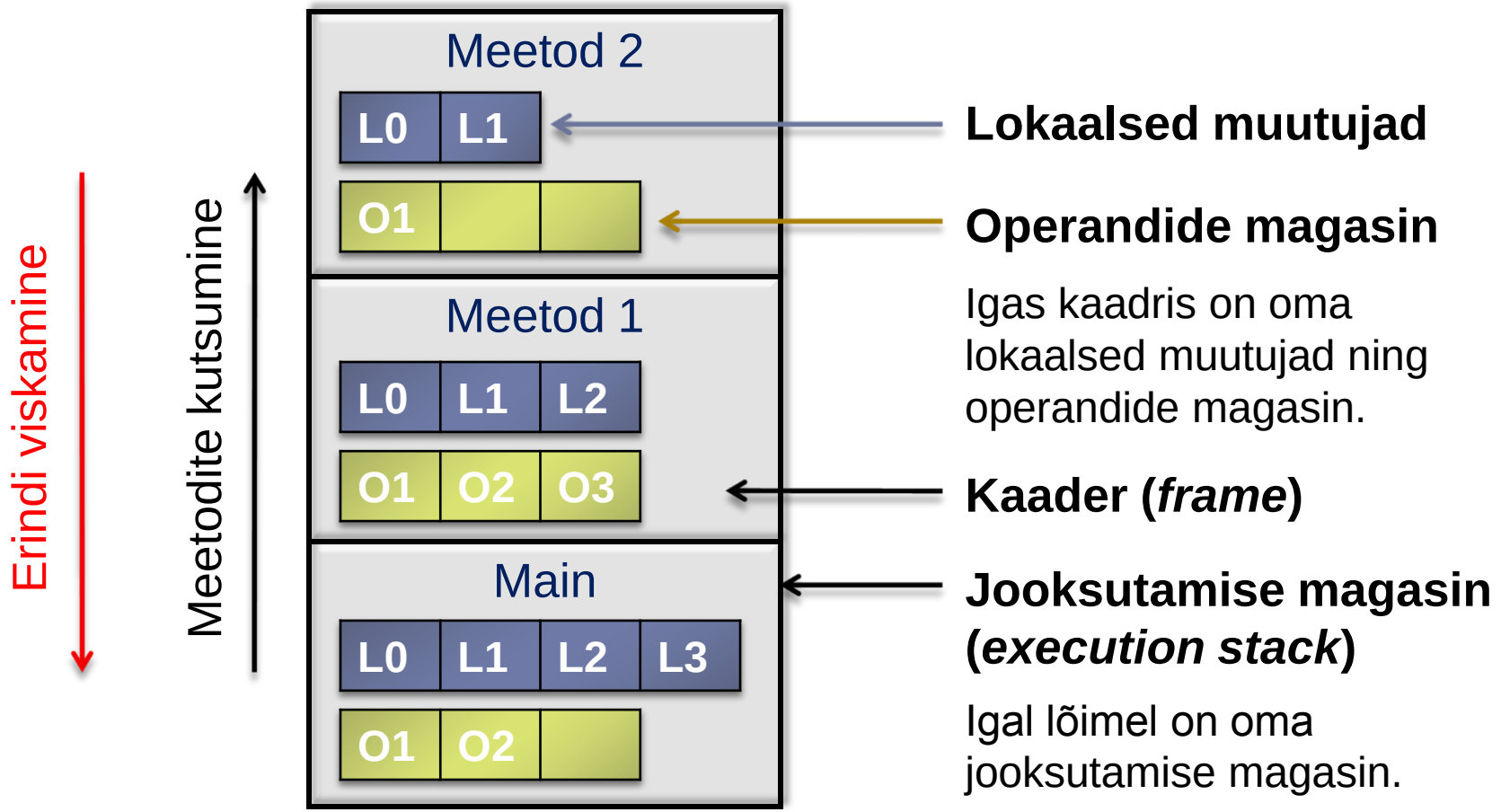
Jooksutamise magasin

```
class Test {
  public static void main(String[] a) { meetod1(); }
  static void meetod1() { meetod2(); }
  static void meetod2() { throw new Error("Viga"); }
}
```

```
java.lang.Error: Viga
at Test.meetod2(Test.java:4)
at Test.meetod1(Test.java:3)
at Test.main(Test.java:2)
Exception in thread "main"
```



Jooksutamise magasin (2)



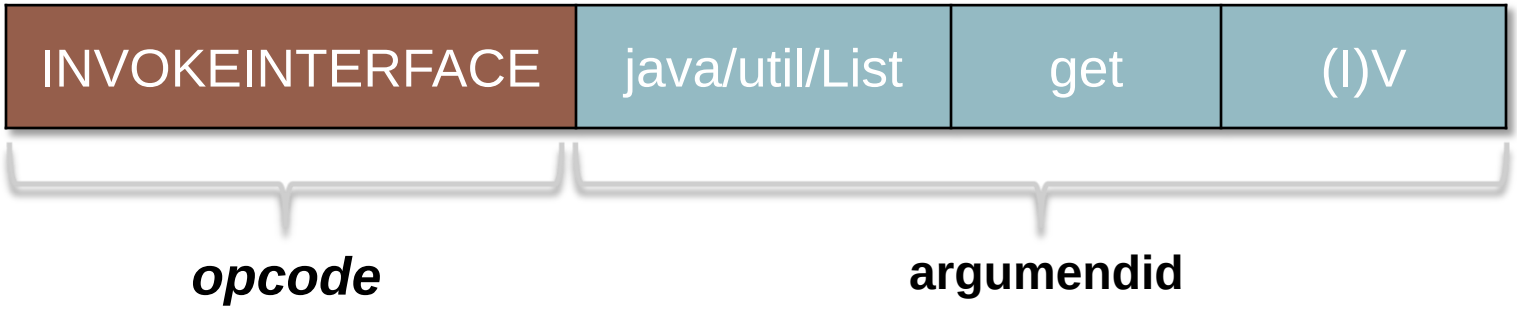
Instruktsioon

- Kompileeritud meetodi kood koosneb käskudest e. **instruktsioonidest**

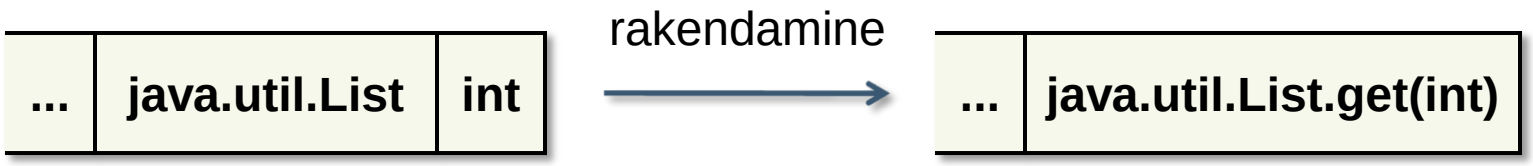
- Kaks rühma:
  - Instruktsioonid, mis tõstavad väärtusi **lokaalsete muutujate** ja **operandide magasini** vahel
  - Instruktsioonid, mis kasutavad ainult **operandide magasini**

Instruktsiooni näide

Instruktsioon:



Operandide magasin instruktsiooni rakendamisel:



viimased 2 elementi eemaldatakse ning lisatakse 1 uus element

Lokaalsete muutujate instruktsioonid

Opcode		Lokaalse muutuja tüüp
ILOAD	**ISTORE**	boolean, byte, char, short, int
LLOAD	**LSTORE**	long
FLOAD	**FSTORE**	float
DLOAD	**DSTORE**	double
ALOAD	**ASTORE**	Object

- ***x*LOAD** loeb lokaalse muutuja ning lisab selle magasini.
- ***x*STORE** eemaldab magasinist elemendi ning salvestab selle lokaalseks muutujaks.
- Üks argument, mis määrab ära lokaalse muutuja indeksi.

Magasini instruktsioonid

Opcode	Argument	Magasin enne	Magasin pärast
POP		..., v	...
DUP		..., v	..., v, v
SWAP		..., v_1, v_2	..., v_2, v_1

- **POP** eemaldab magasinist viimase elemendi
- **DUP** lisab magasini viimasest elemendist koopia
- **SWAP** loeb magasinist kaks elementi ning lisab samad elemendid vastupidises järjekorras

Konstantide instruktsioonid

Opcode	Argument	Tegevus
ACONST_NULL		lisab magasini NULL viite
ICONST_0		lisab magasini väärtuse 0
FCONST_0		lisab magasini väärtuse 0f
DCONST_0		lisab magasini väärtuse 0d
BIPUSH	*b*	lisab magasini byte tüüpi väärtuse *b*
SIPUSH	*s*	lisab magasini short tüüpi väärtuse *b*
LDC	*cst*	lisab magasini int, float, long, double, String või Class tüüpi konstandi *cst*

Aritmeetika instruktsioonid

Opcode	Magasin enne	Magasin pärast
IADD, LADD, FADD, DADD	..., a, b	..., a + b
ISUB, LSUB, FSUB, DSUB	..., a, b	..., a – b
IMUL, LMUL, FMUL, DMUL	..., a, b	..., a * b
IDIV, LDIV, FDIV, DDIV	..., a, b	..., a / b
IREM, LREM, FREM, DREM	..., a, b	..., a % b

Teisenduste instruktsioonid

Opcode	Argument	Magasin enne	Magasin pärast
I2B		..., I	..., (byte) i
I2C		..., i	..., (char) i
I2S		..., i	..., (short) i
L2I, F2I, D2I		..., a	..., (int) a
I2L, F2L, D2L		..., a	..., (long) a
I2F, L2F, D2F		..., a	..., (float) a
I2D, L2D, F2D		..., a	..., (double) a
CHECKCAST	*class*	..., o	..., (class) o

Objektide instruktsioonid

Opcode	Argument	Magasin enne	Magasin pärast
NEW	*class*	...	..., new *class*
INSTANCEOF	*class*	..., o	..., o instanceof class
MONITORENTER		..., o	...
MONITOREXIT		..., o	...

- **MONITORENTER** lukustab objekti antud lõime poolt
- **MOINTOREXIT** vabastab objekti antud lõime poolt

Väljade instruktsioonid

Opcode	Argument	Magasin enne	Magasin pärast
GETFIELD	c, f, t	..., o	..., o.f
PUTFIELD	c, f, t	..., o, v	...
GETSTATIC	c, f, t	...	..., c.f
PUTSTATIC	c, f, t	..., v	...

- **GETFIELD** eemaldab magasinist **objekti viite** ning lisab selle objekti väljalt loetud **väärtuse**
- **PUTFIELD** eemaldab magasinist **objekti viite** ja **väärtuse** ning salvestab selle objekti väljale
- **GETSTATIC** ja **PUTSTATIC** teevad sama staatiliste meetodite korral

Meetodite instruktsioonid

Opcode	Arg.	Magasin enne	Magasin pärast
INVOKEVIRTUAL	c, m, t	..., o, v_1, ..., v_n	..., o.m(v_1, ..., v_n)
INVOKESPECIAL	c, m, t	..., o, v_1, ..., v_n	..., o.m(v_1, ..., v_n)
INVOKESTATIC	c, m, t	... v_1, ..., v_n	..., o.m(v_1, ..., v_n)
INVOKEINTERFACE	c, m, t	..., o, v_1, ..., v_n	..., o.m(v_1, ..., v_n)

- Magasinist eemaldatakse niipalju elemente kui on meetodil argumente, lisaks üks objekti viit (v.a staatilise meetodi korral)
- **INVOKEVIRTUAL** käivitab klassis defineeritud meetodi
- **INVOKESPECIAL** käivitab privaatseid meetodeid ja konstruktoreid
- **INVOKESTATIC** käivitab staatilisi meetodeid
- **INVOKEINTERFACE** käivitab liideses defineeritud meetodeid

Hüpete instruktsioonid

Opcode	Arg.	Enne	Pärast	Tegevus
IFEQ	*label*	..., i	...	hüppa kui i == 0
IFNULL	*label*	..., o	...	hüppa kui o == null
GOTO	*label*	...	...	hüppa alati
TABLESWITCH	*label*	..., i	...	hüppa alati
LOOKUPSWITCH	*label*	..., i	...	hüppa alati

- Kõik instruktsioonid ei mahu slaidile ära
- Hüppamist kasutatakse **if, for, do, while, break, continue** jaoks
- **TABLESWITCH** ja **LOOKUPSWITCH** on **switch** jaoks

Tagastamise instruktsioonid

Opcode	Enne	Pärast
IRETURN, LRETURN, FRETURN, DRETURN	..., a	
ARETURN	..., o	
RETURN	...	
ATHROW	..., o	

- ***x*RETURN** lõpetab meetodi täitmise ja tagastab väärtuse
- **RETURN** on **void** tüüpi meetodite jaoks
- **ATHROW** lõpetab meetodi täitmise ja viskab erindi

Demo

Miks?

- **Aspektid**
  - Logimine
  - Autoriseerimine
  - Vahemälu (*cache*)
  - Transaktsioonid
- **Sogastamine** (*obfuscation*)
- **JavaRebel** (baitkoodi kuum laadimine)
- **Terracotta** (JVM tasemel klasterdamine
- **Jätkud** (lõimede osaline serialiseerimine)
- **Sulundid** (*closures*)
- **Oleku versioneerimine**

Täname