# Programozási nyelvek II JAVA 1. gyakorlat

## Berezvai Dániel jegyzete <http://elte.3ice.hu/>

Tanár: Somogyi Péter

Kezdés: 12:15

Vége: 13:45

Multiplatform, Linux és Windows is lesz.

Fejlesztői környezetet nem fogunk használni.

Csak: Tetszőleges szövegszerkesztő + Parancssori Java C fordító, Java Virtuális gép Debuggere (JDB)

## Követelményrendszer

Max. 3 hiányzás.

Pluszminusz lesz minden gyakorlaton, legalább 0 összeredmény. Javítási lehetőség nincs, hiányzásért nem jár mínusz. Előadás és gyakorlat anyagából is.

Két nagybeadandó szigorú határidőre:

* Őszi szünet vége (október 31)
* Első vizsgahét végén (december 19)

Évfolyamszintű gyakorlati ZH (pót ZH rá egy hétre)

Csak a sikertelen részt kell megismételni.

Pozitív pluszminusz esetén felfelé kerekítünk.

## Hello World! program létrehozása, fordítása és debuggolása Windows alatt

hello.java

public class HelloWorld**{**

 public static void main**(**String**[]** args**){**

 System**.**Out**.**println**(**"Hello"**);**

 **}**

**}**

### Fordítás

$$javac HelloWorld.java$$

### Futtatás

$$java HelloWorld$$

### Debug módban

$-g$ kapcsolóval fordítjuk

$jdb$ paranccsal futtatjuk

Fontosabb parancsok a JDB-ben:

* stop at <osztálynév>:<sorszám>
* run
* next
* cont

**javac -g HelloWorld.java**

**jdb HelloWorld**

Initializing jdb ...

> **stop at HelloWorld:3**

Deferring breakpoint HelloWorld:3.

It will be set after the class is loaded.

> **run**

run HelloWorld

Set uncaught java.lang.Throwable

Set deferred uncaught java.lang.Throwable

>

VM Started: Set deferred breakpoint HelloWorld:3

Breakpoint hit: "thread=main", HelloWorld.main(), line=3 bci=0

3 System.out.println("Hello World!");

main[1] **next**

> Hello W

Step completed: orld!

"thread=main", HelloWorld.main(), line=4 bci=8

4 }

main[1] **cont**

>

The application exited

### JDB segítség

connectors -- list available connectors and transports in this VM

run [class [args]] -- start execution of application's main class

threads [threadgroup] -- list threads

thread <thread id> -- set default thread

suspend [thread id(s)] -- suspend threads (default: all)

resume [thread id(s)] -- resume threads (default: all)

where [<thread id> | all] -- dump a thread's stack

wherei [<thread id> | all]-- dump a thread's stack, with pc info

up [n frames] -- move up a thread's stack

down [n frames] -- move down a thread's stack

kill <thread id> <expr> -- kill a thread with the given exception object

interrupt <thread id> -- interrupt a thread

print <expr> -- print value of expression

dump <expr> -- print all object information

eval <expr> -- evaluate expression (same as print)

set <lvalue> = <expr> -- assign new value to field/variable/array element

locals -- print all local variables in current stack frame

classes -- list currently known classes

class <class id> -- show details of named class

methods <class id> -- list a class's methods

fields <class id> -- list a class's fields

threadgroups -- list threadgroups

threadgroup <name> -- set current threadgroup

stop in <class id>.<method>[(argument\_type,...)]

 -- set a breakpoint in a method

stop at <class id>:<line> -- set a breakpoint at a line

clear <class id>.<method>[(argument\_type,...)]

 -- clear a breakpoint in a method

clear <class id>:<line> -- clear a breakpoint at a line

clear -- list breakpoints

catch [uncaught|caught|all] <class id>|<class pattern>

 -- break when specified exception occurs

ignore [uncaught|caught|all] <class id>|<class pattern>

 -- cancel 'catch' for the specified exception

watch [access|all] <class id>.<field name>

 -- watch access/modifications to a field

unwatch [access|all] <class id>.<field name>

 -- discontinue watching access/modifications to a field

trace [go] methods [thread]

 -- trace method entries and exits.

 -- All threads are suspended unless 'go' is specified

trace [go] method exit | exits [thread]

 -- trace the current method's exit, or all methods' exits

 -- All threads are suspended unless 'go' is specified

untrace [methods] -- stop tracing method entrys and/or exits

step -- execute current line

step up -- execute until the current method returns to its caller

stepi -- execute current instruction

next -- step one line (step OVER calls)

cont -- continue execution from breakpoint

list [line number|method] -- print source code

use (or sourcepath) [source file path]

 -- display or change the source path

exclude [<class pattern>, ... | "none"]

 -- do not report step or method events for specified classes

classpath -- print classpath info from target VM

monitor <command> -- execute command each time the program stops

monitor -- list monitors

unmonitor <monitor#> -- delete a monitor

read <filename> -- read and execute a command file

lock <expr> -- print lock info for an object

threadlocks [thread id] -- print lock info for a thread

pop -- pop the stack through and including the current frame

reenter -- same as pop, but current frame is reentered

redefine <class id> <class file name>

 -- redefine the code for a class

disablegc <expr> -- prevent garbage collection of an object

enablegc <expr> -- permit garbage collection of an object

!! -- repeat last command

<n> <command> -- repeat command n times

# <command> -- discard (no-op)

help (or ?) -- list commands

version -- print version information

exit (or quit) -- exit debugger

<class id>: a full class name with package qualifiers

<class pattern>: a class name with a leading or trailing wildcard ('\*')

<thread id>: thread number as reported in the 'threads' command

<expr>: a Java(TM) Programming Language expression.

Most common syntax is supported.

Startup commands can be placed in either "jdb.ini" or ".jdbrc"

in user.home or user.dir

## Ugyanez most Linux alatt

Szövegszerkesztő: $gedit$

$Terminál$ a parancssor neve.

## Kerekítési hiba (A float típus pontatlansága)

//10 db 0.1-et adjunk össze:

float sum**=**0**;**

**for(**int i**=**0**;**i**<**10**;**i**++){**

 sum**+=**0.1**;**

**}**

System**.**out**.**println**(**sum**);**

Az eredmény a várt $1$ helyett $1.0000001$.

## Infinity, Not a number hibák, túlcsordulás, primitív adattípusok használata, változók, értékadás

## Fahrenheit ↔ Celsius konvertáló

public class Celsius**{**

 private static Double convert**(**Double x**){**

 **return** **(**x**-**32**)\***5**/**9**;**

 **}**

 public static void main**(**String**[]** args**){**

 System**.**out**.**println**(**"Converting command line arguments:"**);**

 **for(**int i**=**0**;**i**<**args**.**length**;**i**++){**

 System**.**out**.**println**(**convert**(**Double**.**parseDouble**(**args**[**i**])));**

 **}**

 System**.**out**.**println**(**"Converting console input... (Break with CTRL+C to exit)"**);**

 String s**=**System**.**console**().**readLine**();**

 **while(true){**

 System**.**out**.**println**(**convert**(**Double**.**parseDouble**(**s**)));**

 s**=**System**.**console**().**readLine**();**

 **}**

 **}**

**}**

Gyakorlat vége.