Introduction
Course information

- Petr Hnětynka
  - hnetynka@d3s.mff.cuni.cz
- http://d3s.mff.cuni.cz/~hnetynka/java/

- 2/2 Zk/Z

- exam
  - written test

- "zápočet"
  - practical test in the lab
  - "zápočtový" program
    - "reasonable" size
    - topic till 10. 1. 2020
      - by email
    - homeworks – 225 points (450 max)
  - presence
    - > 3 absences – 315 points
Course information

- Virtual practical for repeated “subscription”
  - and those who do not want to attend

- List of “forbidden” topics for the “započtový” program
  - tick-tack-toe (“piškvorky”)
  - battleships
  - tetris
  - …
  - homeworks for courses like Algorithms, Graphics,…
  - …

- always agree on the topic with a particular teaching assistant

- the practical at Mon 14:00 in English
Literature, links

• Everything about Java
  - http://www.oracle.com/technetwork/java/

• Java tutorial
  - https://docs.oracle.com/javase/tutorial/index.html

• Java Language Specification
  - http://docs.oracle.com/javase/specs/
Java

- object oriented
  - (almost) all is object
- interpreted
  - source code (.java) – compiled to the bytecode
  - bytecode (.class) – interpreted by the virtual machine
    - just-in-time compilation
      - compilation of the bytecode to a native code before/during program execution
- platform independent
  - programs run in the virtual machine

- since Java 9
  - ahead-of-time compilation
History

- 1.0 (1996)
- 1.1 (1997)
  - Inner classes
- Java 2 platform (2000)
  - 1.2, 1.3 – changes in libraries only
- 1.4 (2002)
  - Assert
- 5.0 (2004)
  - Changes in the language
    - generics, annotations,...
- 6 (2006) – changes in libraries only
- 7 (2011) – (small) changes in the language
- 8 (2014) – big changes in the language
  - lambdas,...
- 9 (2017) – changes in the language – modules
- 10 (2018) – changes in the language – loc. var. type inference (var)
- 11 (2018) – changes in libraries (reducing std lib.)
  - long-term support
- 12 (2019) – modified switch (a “preview” feature)
- 13 (2019) – further switch modifications, text blocks (still “preview”)
Java platform

- JSE – standard edition
- JEE – enterprise edition
- JME – micro edition
Obtaining Java

- http://www.oracle.com/technetwork/java/javase/
downloads/index.html
  - JDK
    - compiler, virtual machine, debugger, ...
      - Windows, Linux, Solaris
  - JRE
    - without development tools (i.e. without compiler,...)
      - Windows, Linux, Solaris
    - documentation
  - IDE
    - Netbeans – http://www.netbeans.org/
    - Eclipse – http://www.eclipse.org/
    - IntelliJ IDEA – https://www.jetbrains.com/idea/
  - Ant – like the make program
    - http://ant.apache.org/
  - Maven – „like Ant on Steroids“
    - http://mave.apache.org/
Approx. time-line of the course

- Language
  - classes, primitive types, programming constructions,...

- Basic tools

- Core classes from the std. library
  - threads, collection, I/O,...
Popularity

IEEE Spectrum | Popularity Index

[TIOBE Index](https://www.tiobe.com/tiobe-index/)
JAVA

Language
Comments

• Comment

    /* comment */
    // comment till the end of the line

• "documentation" comments (*javadoc*)

    /** comment */
Objects

- Everything is object
- Object – an instance of a class or array
  - new instances via the operator `new`
- Everything defined in a class
  - i.e. no functions outside classes (e.g. like in C++)
- Working with objects – references
  - no pointers

```java
String s;
String s = new String("hello");
```
StringBuilder s1 =
    new StringBuilder("hello");
StringBuilder s2 = s1;

s1.append(" world");

System.out.println(s2);
    // prints out "hello world"
Primitive types

- Exception – not everything is object
  - variables are not references
  - fixed size, signed only
    int a = 10;

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Min</th>
<th>Max</th>
<th>Wrapper</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Boolean</td>
</tr>
<tr>
<td>char</td>
<td>16-bit</td>
<td>Unicode 0</td>
<td>Unicode $2^{16}-1$</td>
<td>Character</td>
</tr>
<tr>
<td>byte</td>
<td>8-bit</td>
<td>-128</td>
<td>+127</td>
<td>Byte</td>
</tr>
<tr>
<td>short</td>
<td>16-bit</td>
<td>$-2^{15}$</td>
<td>$+2^{15}-1$</td>
<td>Short</td>
</tr>
<tr>
<td>int</td>
<td>32-bit</td>
<td>$-2^{31}$</td>
<td>$+2^{31}-1$</td>
<td>Integer</td>
</tr>
<tr>
<td>long</td>
<td>64-bit</td>
<td>$-2^{63}$</td>
<td>$+2^{63}-1$</td>
<td>Long</td>
</tr>
<tr>
<td>float</td>
<td>32-bit</td>
<td>IEEE754</td>
<td>IEEE754</td>
<td>Float</td>
</tr>
<tr>
<td>double</td>
<td>64-bit</td>
<td>IEEE754</td>
<td>IEEE754</td>
<td>Double</td>
</tr>
</tbody>
</table>
Primitive types – variables

```java
int i1 = 42;
int i2 = i1;

i1 += 1;

System.out.println(i2);
// prints out 42
```
Primitive types

- Internal representation of integer types
  - „signed two's-complement integers“
  - example for byte
    - 0 ~ 00000000
    - 127 ~ 01111111
    - -1 ~ 11111111
    - -128 ~ 10000000

- Floating point types
  - allow representation of the NaN value (not-a-number)
    - every comparison of NaNs is false
Autoboxing, autounboxing

- since Java 5
- automated conversion between primitive types and corresponding wrappers

```java
int a = 5;
Integer b = a;    // autoboxing
int c = b;        // autounboxing
```
Arrays

• access checked at run-time

• definitions of arrays
  
  int[] iArray;
  int i2Array[];

• multidimensional array
  
  int[][] iiArray;

• instantiation of arrays – only dynamically
  
  iArray = new int [10];

• array length
  
  iArray.length
Object disposal

- garbage collector
class MyClass {
    /* class body */
}

- class body
  - fields
  - methods
  - inner classes
class MyClass {
    int i;
    float f;
    boolean b;
    String s;
}

...  
MyClass m = new MyClass();
m.i = 5;
m.f = 3.7;
m.b = true;
m.s = new String();
Class: Fields

- Default values
  - boolean – false
  - other primitive types – 0
  - references – null

- Warning
  - local variables are not initialized
  - compilation error
Class: Methods

returnType methodName ( arguments ){
    method body;
}

class MyClass {
    int pow2(int a) {
        return a*a;
    }

    void nothing() {}
}
Class: Methods

• method call

    object.methodName(arguments)

    MyClass m = new MyClass();
    int a = m.pow2(5);

• Arguments passed by value

    class Foo {
        void plusOne(int a) {
            a = a + 1;
        }
        void use() {
            int a = 5;
            plusOne(a);
            System.out.println(a); // 5
        }
    }

    class Bar {
        void appendA(StringBuilder sb) {
            sb.append("A");
        }
        void use() {
            StringBuilder sb =
                new StringBuilder("A");
            appendA(sb);
            System.out.println(sb); // AA
        }
    }
• Since Java 5

```java
enum Planet {
    MERCURY, VENUS, EARTH, MARS,
    JUPITER, SATURN, URANUS, NEPTUNE,
    PLUTO
};

...  
public Planet pl = MARS;
```
Packages

- namespaces
- package
  - a set of classes related in some way
  - like `namespace` in C#, C++

- every class belongs to exactly one package
  - an explicitly specified, or
  - the default unnamed package

- package specification
  ```
  package nameOfPackage;
  ```
Packages

- hierarchical names
  - "reversed" internet domain name of a creator
  - cz.cuni.mff.java.example01
  - org.w3c.dom

- full name of a class
  - `packageName.ClassName`

- class from the same package – "short" name

- classes from another package – full name

- simplified usage by `import`

  ```java
  import packageName.ClassName;
  import packageName.*;
  ```

- `package java.lang` – always imported
Key-word static

- **static fields and methods**
  - not connected with a particular instance (object)
  - "class data", "class methods"

```java
class MyClass {
    static int i;
}

class MyClass2 {
    static void incr() {
        MyClass.i++;
    }
}
```
static import

- since Java 5
- import of static elements
- usage without the class name

```java
import static java.lang.Math.PI;
import static java.lang.Math.tan;
...
tan(PI/4);
```
Local variables visibility

```java
{
    int x = 10;
    // x is visible
    {
        int y = 11;
        // x and y are visible
    }
    // x is visible only
}

{
    int x = 1;
    {
        int x = 2;  // compile-time error
    }
}
```
Classes and files

- every `public` class in a separated file
- the same name as the class + the `.java` extension
- packages ~ directories

```java
package packageName;

import ....;
import ....;

public class ClassName {
    ....
}
```

- non-public classes (without `public`)
  - visible from the same package only
package cz.cuni.mff.java.example01;

public class Hello {
    public static void main(String[] args){
        System.out.println("Hello world!");
    }
}

• save to
  - directory .../cz/cuni/mff/java/example01
  - file Hello.java
Program

• compilation
  - `javac Hello.java`
  - `creates` `Hello.class`

• execution
  - `java cz.cuni.mff.java.example01.Hello`

• CLASSPATH
  - list of directories, where the compiler/virtual machine looks for classes
    • environment variable `CLASSPATH`
    • arguments `-cp`, `-classpath`
  - examples
    • `/home/petr/java/cz/cuni/mff/java/example01/Hello.class`
    • `java -cp /home/petr/java cz.cuni.mff.java.example01.Hello`
Executing “sources”

- since Java 11
- java HelloWorld.java
Modules – since Java 9

• a module
  − a named collection of classes (and other elements)
  − (a set of packages)

  − declares, which
    • other modules it requires
    • own packages exports

  − the visibility (accessibility) of classes is changed

• module-info.java
  module com.foo.bar {
    requires com.foo.baz;
    exports com.foo.bar.alpha;
    exports com.foo.bar.beta;
  }
Modules – since Java 9

• MODULEPATH
  – similar to CLASSPATH

• modules can be “ignored”
  – without a module specified => a class is in the *unnamed* module
    • requires all other modules
    • exports all of its packages

  – particularly for backward compatibility
Operators: assignment

• Assignment
  
  ```java
  int i;
  int[] array;

  i = 4;
  array[4] = 5;
  4 = i;  // compile-time error
  ```

• Primitive types
  - copying values

• Objects
  - copying references
    • not objects!
Operators: arithmetic

- unary
  - + -
- binary
  - + - * / %
- "short-cuts" for assignment
  - += -= *= /= %=
- increment and decrement
  - prefix and postfix
    - i-- i++ --i ++i
- overflows and underflows are “silent”
  - no exception
Operators: comparison

- **boolean** result
  
  ```java
  ==  !=  all types
  <   >   <=  >=  all primitive except *boolean*
  ```

- test – what is printed out?
  ```java
  Integer i1 = new Integer(2);
  Integer i2 = new Integer(1);
  if (i1 == i2)
      System.out.println("YES");
  else
      System.out.println("NO");
  ```

These constructors are deprecated
Operators: logical

- boolean result
- can be used on boolean only

&& | || !

- short-circuit evaluation
Operators: bitwise

• can be used on **short, int, long, char** and **boolean**

  &    |    ^   ~

• short-cuts

  &=   |=   ^=

• eager evaluation

• type **boolean**
  - considered as 1-bit value
  - operator ~ cannot be used on boolean
Operators: shifts

- can be used on **short, int, long, char**
  - left shift \(<<\)
    - adds zeros to lower bits
  - right shift \(>>\)
    - if number positive – adds zeros
    - if number negative – adds ones
  - unsigned right shift \(>>>\)
    - always adds zeros

- **char, byte, short**
  - first converted to **int**
  - result – always **int**

- **long**
  - result is **long**
Operators: misc

• Ternary operator
  ```java
  int a;
  a = a > 0 ? a : 0;
  ```

• Operator **comma**
  - only in the begging of the **for** cycle

• Operator **+** on **String**
  - concatenates Strings
    - if there is at least one String and only the **+** operators in an expression, then everything is converted to String and concatenated

• Cast
  ```java
  int i = 1;
  long x = (long) i;
  ```

• No sizeof operator
  - no need
### Operators: priority

| unary          | +  -  ++  -- |
|               | *  /  %  +  -  <<  >> |
| comparison     | >  <  >=  <=  ==  != |
| logical and bitwise | &&  | |  &  |  ^  |
| ternary        | ? : |
| assignment     | = (shortcuts +=) |

- In a case of the same priority, expression is evaluated from left
if (boolean-expression)
   statement
else
   statement

• else branch can be ommitted
• statement
  – single statement, or
  – block { ....... }
while, do - while

while (boolean-expression)
  statement

do
  statement
while (boolean-expression);

• cycling while the boolean expression is true
for (initialization; boolean-expression; step) statement

• in the initialization and step, operator comma can be used

for (int i=1, j=1; i<5; i++, j=i*10) {
    ....
}
for (since Java 5)

```java
int[] arr = new int[10];

for (int i: arr) {
    ...
}
```

- arrays, or
- classes with the `iterator`
break, continue

- **break**
  - stops a cycle execution

- **continue**
  - stops the current round of a cycle and starts new one

- **labels** – have meaning only with cycles

```java
label: outer-cycle {
    inner-cycle {
        break;
        continue;
        continue label;
        break label;
    }
}
```
goto

- reserved, but
- not used
switch

int a;
...

switch (a) {
    case 1:
    case 2: System.out.println("1, 2");
            break;
    case 3: System.out.println("3");
            break;
    default: System.out.println("3..");
}

- since Java 7, switch can be used with the String type
- since Java 12, extended switch – “preview” feature only