University of Bamberg

Java-Tutorial for ISSS-Students

Chapter 2: Variables and data types

Chapter 2: Variables and data types

1. Structure of variables
2. Representing numbers
3. Representing characters
4. Representing boolean values
5. Relational operators

Structure of variables

- In Java programs data are represented in variables:
  - *dataType variableName* = *value*;
- We differentiate between primitive (like numbers, characters, boolean values) and complex data types
- The name of the variable should be reasonable, written in lower case/ camel case, and start with a letter
  - Allowed components: letters, numbers, $, _
- The value of the variable has to be a valid value within the range of values of the chosen data type

Representing numbers

Data type	Bit	Minimum	Maximum
byte	8-Bit	-128	+127
short	16-Bit	-32728	+32767
int	32-Bit	-2147483648	+2147483647
long	64-Bit	-9223372036854775808	+9223372036854775807
float	32-Bit	$-3.4 * (10^{38})$	$+3.4 * (10^{38})$
double	64-Bit	$-1.7 * (10^{308})$	$+1.7 * (10^{308})$

Arithmetic operators

Operator	Meaning	Short form
+	addition	+= or ++ (add 1)
-	subtraction	-= or -- (subtract 1)
*	multiplication	*=
/	division	/=
%	division with remainder	%=

Representing characters

- Characters are defined in single quotation marks:
  - char s = ‘S’
- You could as well use the decimal encoding …
  - char s = 83
- … or the *Unicode*-notation:
  - char s = ‘\u0053’
- Some characters have a special function in Java and can only be represented via *escape-sequences*

Escape-sequences

Escape-sequence	Meaning
\t	tabulator
\n	new line
\‘	single quotation mark
\“	quotation mark
\\	backslash

Representing boolean values

- There are only two boolean values:
  - boolean b = true;
  - boolean b = false;
- They can be combined into boolean expressions via boolean operators

Boolean operators

Operator	Meaning	Return value
!	negation	the opposite of the original expression
&	AND	*true* if both expressions are *true*
|	OR	*true* if at least one expression is *true*
&&	conditional AND	like AND, stops if the first expression is *false*
||	conditional OR	like OR, stops if the first expression is *true*
^	exclusive OR	*true* if exactly one expression is *true*

Relational operators

Operator	Meaning	Return value
<	less than	*true* if left value < right value
<=	less than or equal to	*true* if left value <= right value
>	greater than	*true* if left value > right value
>=	greater than or equal to	*true* if left value >= right value
==	equal to	*true* if both values are the same
!=	not equal to	*true* if both values are not the same