JAM

JAM

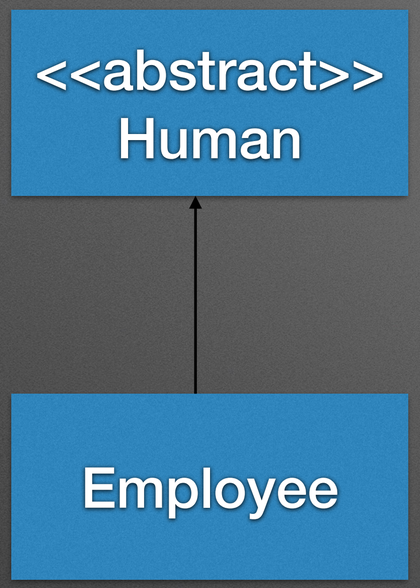
Java Academy Munich

Primitive types
int, float, double, etc.

Munich, 2020 August 14th

JAM

JAM




```
Human me = new Employee();
```

static type

```
Human me = new Employee();
me.getName();
```

dynamic type

What are variables?

Variables are a concept from imperative programming.

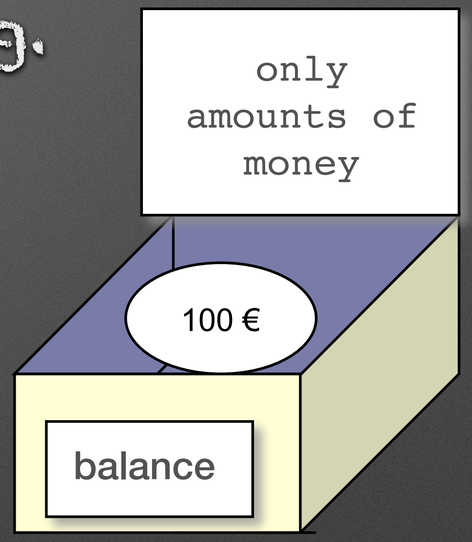
They consist of:

=> a name to address them

=> a value that is stored under this variable name

=> a type defining the possible values that this variable can hold

Treat them like baskets for particular thing.



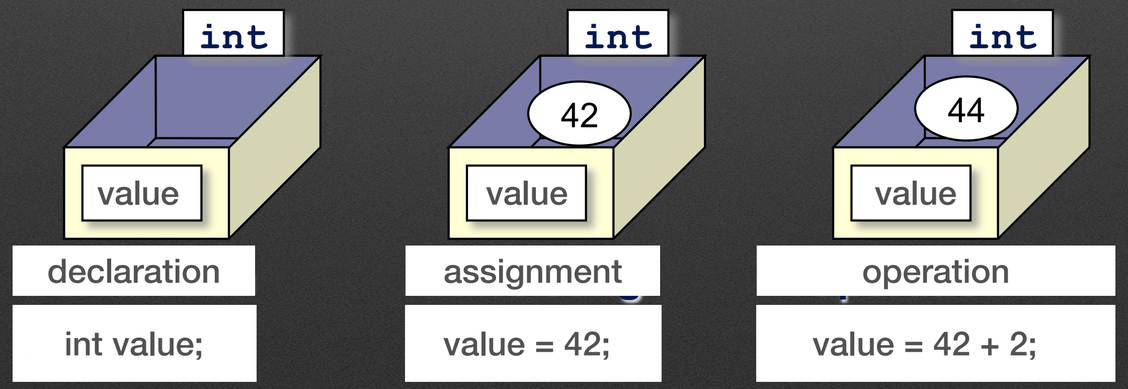
What are variables?

Markers of a variable:

=> The name is just an identifier which will be bound to the typed value.

=> This type determines

- the value set (e.g. int => only integers) and
- the possible operations (e.g. int add, subtract, ...)



primitive type vs reference types

Imperative and object-oriented programming languages usually offer a set of elementary types or basic types: Natural numbers (Integer), real numbers (Float or Real), characters (Character) and truth values (Boolean).

Java has a number of elementary types (basic or primitive types): byte, short, int, long, float, double, boolean and char.

Everything else are objects - even the Strings!

value ranges

value	mem size	range from to
byte	1 Byte	-128 bis +127
short	2 Bytes	-32.768 bis +32.767
int	4 Bytes	-2.147.483.648 bis +2.147.483.647
long	8 Bytes	-9.223.372.036.854.775.808 bis +9.223.372.036.854.775.807
float	4 Bytes	$1{,}40239846*10^{-45}$ bis $3{,}40282347*10^{38}$
double	8 Bytes	$4{,}9406564584124654 4*10^{-324}$ bis $1{,}7976931348623157 0*10^{308}$

assignment: Integers

The value range of the type on the right side must be smaller or equal to that of the type on the left side.

char is interpreted as an unsigned numeric value and therefore matches int or long.

If a char constant is small enough, then it may also be assigned to byte or short.

You cannot assign integer types to char variables, because the sign could be lost.

```
int   i;
short s;
char  c;
```

```
i = s;  // ok, short is smaller than int
s = i;  // fail, short is too small
i = c;  // ok, int is large enough
c = s;  // fail, sign will get lost
s = 'a'// ok. value (97 for ‚a') matches
```

assignment: float values

The value range of the type on the right side must be smaller or equal to that of the type on the left side.

Therefore a float can be assigned to a double.

Since the value range of integer types is smaller than that of float, the assignment is possible.

Floating point types cannot be assigned to integer types because the value range is too small.

Float constants are written with an f as postfix, otherwise the type double is assumed!

```
int     i=3;
float   f=3.14f;
double  d=3.1415;
```

```
f = i;   // ok, int is smaller than float
d = f;   // ok, float is smaller than double
f = i+f;// ok, implicite type cast and float used
i = i+f;// fail, float will not be converted to int
```

assignment: chars / Character

Characters are stored as 16-bit Unicode. The ASCII characters are a subset of Unicode.

You can specify characters as constants in single quotes.

Special characters can be specified via their character code or via predefined escape sequences.

```
char c = 'a';
c = '\u0027'; // Unicode for quote sign
c = '\''; // escape character for quote
c = 65;   // converted code for A
```

some escape sequences

- `\\` – \
- `\n` – new line
- `\'` – '
- `\"` – "
- `\t` – tab

assignment: Boolean

Booleans cannot be assigned to any other type (and vice versa).

There are two predefined boolean constants: true and false.

```
boolean done = false;
boolean isWeekend = true;

isWeekend = dayIndex >= 6;
```

Operators

Caution: Depending on the type of operand, the operator can have a different meaning!

Overview:

arithmetic unary	++ -- + -
arithmetic binary	* / + - % (Modulo)
logically unary	!
logically binary	&& (AND) || (OR) ^ (XOR) & |
bitwise binary	& | ^ << >> >>> (bit-shift)
bitwise unary	~ (bitwise complement)
Comparison	< <= > >= == !=
ternary	<booleanexpession> ? <exp> : <exp>
assignment	= += -= *= /= %= <<= >>= >>>= &= ^= |=

Basic arithmetic operations

Java defines the basic arithmetic operations for all basic numerical types.

The modulo operator returns the integer division remainder.

The usual operator precedents apply (multiplication before addition).

*	multiplication
/	division
%	modulo
+	addition
-	subtraction

```
a = b + 32;
n = (a + 1) * (b + 1);

anteil = menge / leute;
rest = menge % leute;
```

Increment / Decrement

For lazy writers

Increment and decrement are taken from C and are a short form of x=x+1 or x=x-1.

With the pre-versions, the value is first changed and then delivered as the result.

In the post versions, the old value is used in the printout and then the value is changed

`n++`	post increment
`n--`	post decrement
`++n`	pre increment
`--n`	pre decrement

```
int x = 4;
int y;
y = x++;
y = ++x;
y = x-- * --x;
```

Caution:

Using pre and post increment or decrement operators might reduce the maintainability od the code!

Comparisons and assignment

Comparisons always have a boolean value as a result.

The operands must be comparable.

Assignments can be combined with other operators.

Assignments have a value and can be used as part of an expression

=	assignment
==	equal
!=	not equal
<=	less or equal
>=	greater or equal
<	less than
>	greater than

```
int i;
int j = i = 5;
boolean b = i == 5;
i += 5; // same as i = i + 5;
```

Caution:

Do not mixup assignment (=) and comparison (==)!

Logical expressions

Operators for NOT, AND, OR and XOR are available.

The operators | and & evaluate the expression completely even if the result is already fixed.

The commonly used operators || and && allow a short-circuit evaluation. The expression is calculated only as far as necessary

!	logical NOT
|	logical OR
&	logical AND
^	logical XOR
||	conditional OR
&&	conditional AND

```
getup = !(tired || inVacation);
fertig = gefunden || x++ > 5;
fertig = gefunden | x++ > 5;
```

Caution:

The use of logical NOT or logical AND might have side effects!

ternary expressions

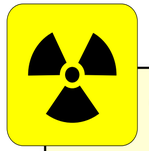
case discrimination in expression

Ternary expressions are similar to if else control structures, but have a result value.

The type of the result is determined by the type of the alternative expressions.

```
<bool-exp>
? <exp>
: <exp>;
```

```
String result = a % 2 > 0 ? "ungerade" : "gerade";
result = "Guten " +
  zeit < 12
  ? "Morgen"
  : zeit < 18
    ? "Nachmittag"
    : "Abend";
```



Caution:

The use of ternary expressions can make the code difficult to read.

Initialization of variables

default values

If a variable is not explicitly assigned a value when it is declared, it receives a standardized initial value.

Default values are 0, 0.0, false and '\u0000

constants

To make the variable unchangeable, use the keyword final.

Constants must be initialized.

Attributes can also be declared as constants.

```
final int i = 10;
final double pi = 3.1415;
final char a = 'a';
```