Part 1

Variables

Learning Objectives

- Understand the concept of a variable. You know what variable types, names, and values are.
- Know how to create and use string, integer, floating-point, and boolean variables.

Media and Technology Use Questionnaire

Please respond to the following questionnaire about your media and technology use. With this questionnaire, we collect information about media and technology use, along with student attitudes toward them. Your replies will be used in developing MOOC courses and education research. Open the questionnaire by following the link .

Quiz:
Media and technology use questionnaire

Points:
2/2

Above you find a link to a questionnaire. After you have answered the questionnaire, check the checkbox below and click "submit". You will be awarded two points for answering the questionnaire.

- [x] I have answered the questionnaire

Answered The number of tries is not limited

We've already familiarized ourselves with strings to a degree while dealing with user inputs. Let's turn our attention to learning about other variable *types* commonly used in Java.

A variable can be thought of as a container in which information of a given type can be stored. Examples of these different types include text (`String`), whole numbers (`int`), floating-point numbers (`double`), and whether something is true or false (`boolean`). A value is assigned to a variable using the equals sign (`=`).

```
int months = 12;
```

In the statement above, the value of 12 is assigned to an integer variable called months. The statement could be read as: "the variable months is assigned the value 12".

A variable's value can be joined to a string using the + sign, as seen in the following example.

```
String text = "contains text";
int wholeNumber = 123;
double floatingPoint = 3.141592653;
boolean trueOrFalse = true;

System.out.println("Text variable: " + text);
System.out.println("Integer variable: " + wholeNumber);
System.out.println("Floating-point variable: " + floatingPoint);
System.out.println("Boolean: " + trueOrFalse);
```

Output:

Sample output

```
Text variable: contains text
Integer variable: 123
Floating-point variable: 3.141592653
Booolean: true
```

Programming exercise:

Various Variables

Points
1/1

The exercise template contains a program that prints the following:

Sample output

```
Chicken:
3
Bacon (kg):
5.5
Tractor:
None!

And finally, a summary:
3
5.5
None!
```

Modify the program in the given places so that it outputs the following:

Sample output

```
Chicken:
9000
Bacon (kg):
0.1
Tractor:
Zetor

And finally, a summary:
9000
0.1
Zetor
```

Exercise submission instructions

How to see the solution

Variable names are unique - no two variables can have the same name. The program in the following example is faulty as it attempts to create the variable `pi` twice. The variable is created the first time its declared.

```
public class Example {
    public static void main(String[] args) {
        double pi = 3.14;
        double pi = 3.141592653;

        System.out.println("The value of pi is: " + pi);
    }
}
```

The variable type is stated when the variable is first declared. When a new value is assigned to the variable, the type is no longer declared.

```
int value = 10;
System.out.println(value);
value = 4;
System.out.println(value);
```

Sample output

```
10
4
```

Changing a Value Assigned to a Variable

A variable exists from the moment of its declaration, and its initial value is preserved until another value is assigned to it. You can change a variable's value using a statement that comprises the variable name, an equals sign, and the new value to be assigned. Remember to keep in mind that the variable type is only stated during the initial variable declaration.

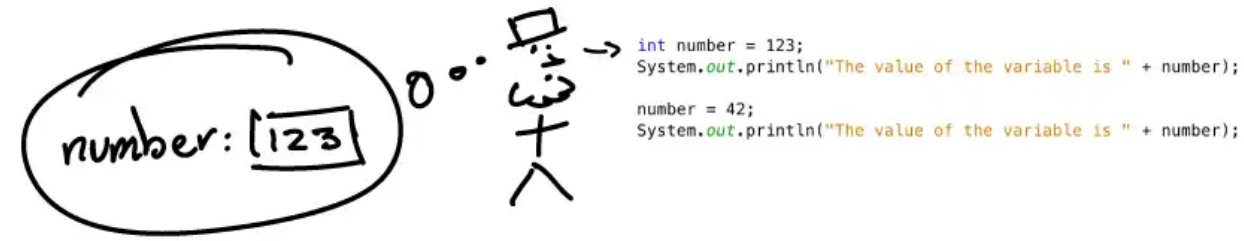
```
int number = 123;
System.out.println("The value of the variable is " + number);

number = 42;
System.out.println("The value of the variable is " + number);
```

Output:

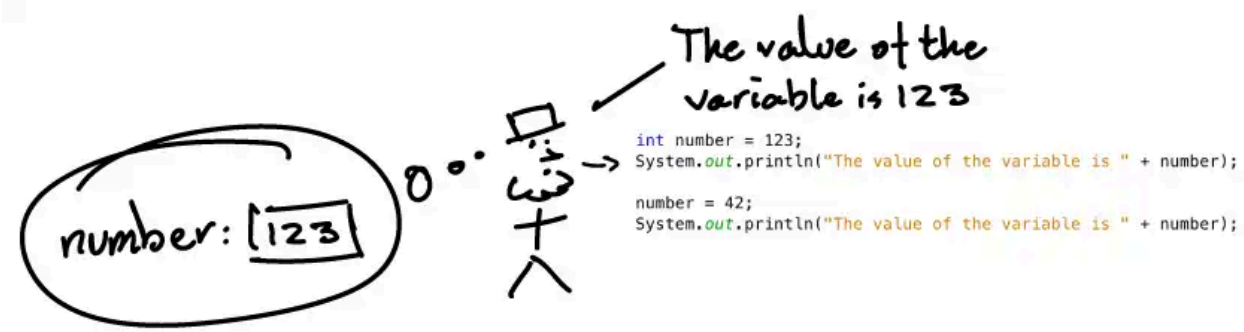
Sample output

```
The value of the variable is 123
The value of the variable is 42
```

Let's look at the preceding program's execution step-by-step. When a variable appears in the program for the first time, i.e., the computer is told both its type (in this case `int`) and its name (in this case `number`), the computer creates a 'named container' for the variable. Then, the value on the right side of the equals sign is copied into this named container.

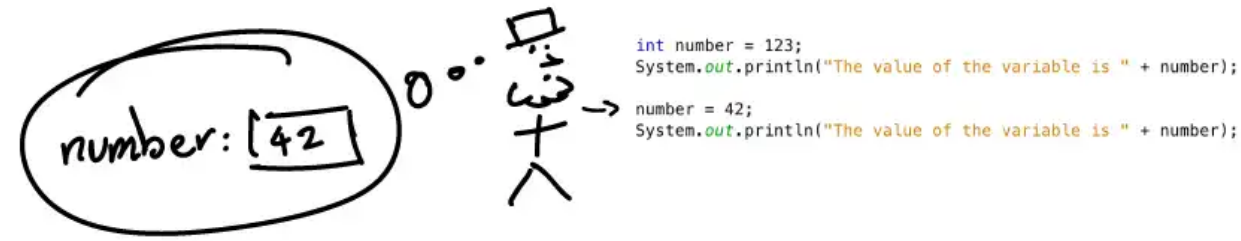


Whenever a variable is referenced by its name in a program — here, we want to print the string "The value of the variable is " followed by the value of the `number` variable — its value is retrieved from a container that has the corresponding name.

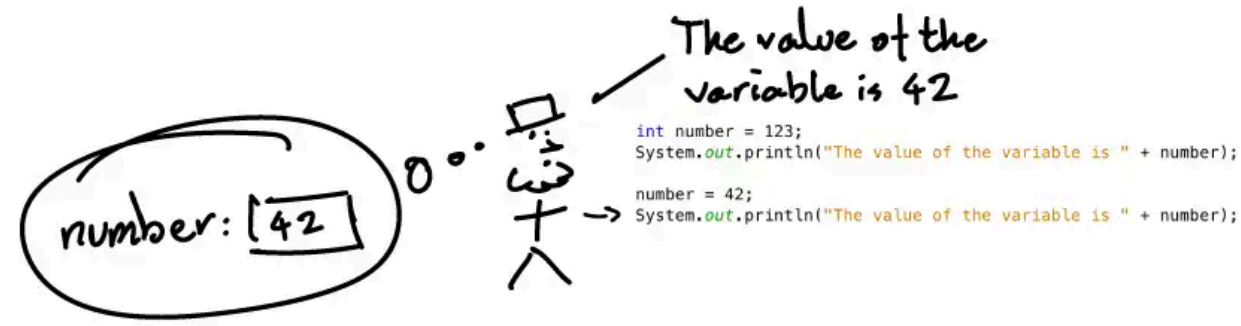


Whenever a value is assigned to a variable (here `number = 42`), a check is run to see whether a container with the given name already exists. If one

does, a new value is copied in the place of the old value, and the old value disappears. If a container of the variable name is not found, the program execution ends in an error message, or it fails to run.



The variable is then referenced again by its name in the program — we again want to print the string "The value of the variable is " followed by the value of the `number` variable. We proceed as normal, retrieving the value of `number` from a container having its name.



At the end of the program, you'll notice that the original value of the variable has vanished. A variable can hold only one value at a time.

Variable's Type Persists

Once a variable's type has been declared, it can no longer be changed. For example, a boolean value cannot be assigned to a variable of the integer type, nor can an integer be assigned to a variable of the boolean type.

```
boolean integerAssignmentWillWork = false;
integerAssignmentWillWork = 42; // Does not work

int value = 10;
integerAssignmentWillWork = value; // Neither does this
```

However, exceptions do exist: an integer can be assigned to a variable of the double type, since Java knows how to convert an integer to a double

during assignment.

```
double floatingPoint = 0.42;
floatingPoint = 1; // Works

int value = 10;
floatingPoint = value; // Also works
```

A floating-point value cannot, however, be assigned to an integer variable. The reason for this is that those who develop the language aim to prevent developers from making errors that lead to a loss of information.

```
int value = 4.2; // Does not work

double floatingPoint = 0.42;
value = floatingPoint; // Neither does this
```

Naming Variables

Naming variables is a fundamental aspect of describing a program. Let's look at two examples.

```
double a = 3.14;
double b = 22.0;
double c = a * b * b;

System.out.println(c);
```

Sample output

```
1519.76
```

```
double pi = 3.14;
double radius = 22.0;
double surfaceArea = pi * radius * radius;

System.out.println(surfaceArea);
```

Sample output

```
1519.76
```

Both of the preceding examples function the same way and output the same result. One of them is, however, much more understandable. The objective here is to compute the surface area of a circle. The value of pi is defined in the first row, the circle's radius in the second, and the surface area calculated in the third.

Variables Express the Program and the Problem to Be Solved

Programming is a problem-solving tool. The aim is to create solutions for any given problem, such as the automation of control in cars. As a problem is approached, the developer decides on the terms used to describe the problem domain. The terminology that is chosen, such as variable names, will serve to describe the problem for anyone who is to work with it in the future.

As you're wording the problem that you're solving, think about the concepts related to that problem and appropriate terms that could be used to describe them. If you find it hard to come up with relevant names, think of the ones that definitely do not describe it. After this, settle on some terminology that you're going to use — the good thing is that you can usually improve on it later on.

Variable naming is limited by certain constraints.

Variable names cannot contain certain special symbols, such as exclamation marks (!). Spaces are also not allowed, since they're used to separate parts of commands. Instead of spaces, the convention in Java is to use a style known as camelCase. **Note!** The first letter of a variable name is always lower-cased:

```
int camelCaseVariable = 7;
```

Numbers can be used within a variable name as long as the name does not begin with a number. Also, a name cannot consists of numbers only.

```
int 7variable = 4; // Not allowed!
int variable7 = 4; // Allowed, but is not a descriptive variable name
```

A variable's name cannot already be in use. These names include, for instance, variables previously defined in the program and commands provided by Java, such as `System.out.print` and `System.out.println`. —>

```
int System.out.print = 4; // Not Allowed!
int System.out.println = 4; // Not Allowed!
```

```
int camelCase = 2;
int camelCase = 5; // Not allowed -- camelCase variable is already in use!
```

Letters containing diacritics (e.g. the letters ä and ö used in Finnish) should also not be used in variable names. You can replace these letters with their non-diacritic equivalents. For example, you should convert ä -> a and ö -> o.

Permissible Variable Names

- lastDayOfMonth = 20
- firstYear = 1952
- name = "Essi"

Impermissible Variable Names

- last day of month = 20
- 1day = 1952
- beware! = 1910
- 1920 = 1

The Type of the Variable Informs of Possible Values

A variable's type is specified when it's initally declared. For example, a variable containing the string "text" is declared with the statement `String`

`string = "text";`, and an integer having the value 42 is declared with the statement `int value = 42;`. —>

A variable's type determines the types of values that can be assigned to it. `String` types hold text, `int` types whole numbers, `double` floating-point numbers, and `boolean` types are either true or false.

As such, the possible values of a given variable type are limited. For example, a string cannot contain an integer, **nor** can a double contain a boolean value.

Type	Example	Accepted values
Whole number, i.e., `int`	`int value = 4;`	An integer can contain whole numbers whose values lie between -2147483648 and 2147483647.
Floating-point number, i.e., `double`	`double value = 4.2;`	Floating-point numbers contain decimal numbers, with the greatest possible value being approximately 2^{1023}. When a decimal number is represented with a floating-point number, the value can be inaccurate as floating-points are incapable of representing all decimal numbers. The reasons behind this are explored in the Computer organization course.
`String`	`String text = "Hi!";`	A string can contain text. Strings are enclosed in quotation marks.
True or false value, i.e., `boolean`	`boolean right = true;`	A boolean contains either the value `true` or `false`.

Reading Different Variable Types from User

In the text-based user interfaces that we've used in our programs, the user's input is always read as a string, since the user writes their input as text. Reading strings from the user has become familiar by this point - we do it using the `nextLine`-command of the Scanner helper method.

```java
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Write text and press enter ");
        String text = scanner.nextLine();
        System.out.println("You wrote " + text);
    }
}
```

Sample output

```
Write text and press enter
text
You wrote text
```

Other input types, such as integers, doubles, and booleans are also read as text. However, they need to be converted to the target variable's type with the help of some tools provided by Java.

Reading Integers

The `Integer.valueOf` command converts a string to an integer. It takes the string containing the value to be converted as a parameter.

```java
String valueAsString = "42";
int value = Integer.valueOf(valueAsString);

System.out.println(value);
```

Sample output

```
42
```

When using a Scanner object, the reading of the value is usually inserted directly into the type conversion. This happens like so:

```java
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Write a value ");
        int value = Integer.valueOf(scanner.nextLine());
        System.out.println("You wrote " + value);
    }
}
```

Sample output

```
Write a value
42
You wrote 42
```

Programming exercise:

Integer Input

Points 1/1

Write a program that asks the user for a value. The program should then print the value provided by the user.

Here's a couple of examples:

Sample output

```
Give a number:
3
You gave the number 3
```

Sample output

```
Give a number:
42
You gave the number 42
```

Exercise submission instructions

How to see the solution

Test the functionality of your program with non-numerical inputs. The program should break as it doesn't know how to convert these inputs into numbers. We'll learn how to deal with exceptional cases like these in the advanced programming course.

Reading Doubles

The `Double.valueOf` command converts a string to a double. It takes the string containing the value to be converted as a parameter.

```
String valueAsString = "42.42";
double value = Double.valueOf(valueAsString);
System.out.println(value);
```

Sample output

```
42.42
```

As with integers, the reading is nested within the conversion. This is done as follows:

```
import java.util.Scanner;

public class Program {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Write a value ");
        double value = Double.valueOf(scanner.nextLine());
        System.out.println("You wrote " + value);
    }
}
```

Sample output

```
Write a value
1234.2
You wrote 1234.2
```

It's possible to also read an integer variable into a double, in which case the value is converted automatically to type double. The example below demonstrates how the previous program functions when the user inputs an integer.

Sample output

```
Write a value
18
You wrote 18.0
```

Programming exercise:

Double Input

Points 1/1

Write a program that asks the user for a floating-point number using the variable type Double. The program then prints the user's input value.

Example prints for the program can be seen below:

Sample output

```
Give a number:
3.14
You gave the number 3.14
```

Sample output

```
Give a number:
2.718
You gave the number 2.718
```

Exercise submission instructions

How to see the solution

Reading Booleans

The `Integer.valueOf` command converts a string to an integer and the `Double.valueOf` converts it to a floating-point. The `valueOf` command also appears when converting a string to a boolean — this is done with the command `Boolean.valueOf`.

Boolean variables can either have the value `true` or `false`. When converting a string to a boolean, the string must be "true" if we want the boolean value to be `true`. The case is insensitive here: both "true" and "TRue" turn into the boolean value of `true`. All other strings turn into the boolean `false`.

```
import java.util.Scanner;

public class program {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Write a boolean ");
        boolean value = Boolean.valueOf(scanner.nextLine());
        System.out.println("You wrote " + value);
    }
}
```

Sample output

```
Write a boolean
I wont'!
You wrote false
```

Sample output

```
Write a boolean
TRUE
```

You wrote true

Sample output

Write a boolean
true
You wrote true

Programming exercise:
Boolean Input

Points
1/1

Write a program that asks the user for a boolean value. The program should then print the value provided by the user.

Example prints for the program can be seen below.

Sample output

Write something:
santa does not exist
True or false? false

Sample output

Write something:
TRUE
True or false? true

Exercise submission instructions

How to see the solution

Summary

Finally, a summary:

```
import java.util.Scanner;

public class Program {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String text = scanner.nextLine();
        int integer = Integer.valueOf(scanner.nextLine());
        double floatingPoint = Double.valueOf(scanner.nextLine());
        boolean trueOrFalse = Boolean.valueOf(scanner.nextLine());

        // and so on
    }
}
```

Programming exercise:

Different Types of Input

Points 1/1

Write a program that asks the user for a string, an integer, a floating-point number, and a boolean. The program should then print the values given by the user.

Example prints for the program can be seen below.

Sample output

```
Give a string:
bye-bye
Give an integer:
11
Give a double:
4.2
Give a boolean:
true
You gave the string bye-bye
You gave the integer 11
You gave the double 4.2
You gave the boolean true
```

Sample output

```
Give a string:
Oops!
Give an integer:
-4
Give a double:
3200.1
Give a boolean:
false
You gave the string Oops!
You gave the integer -4
You gave the double 3200.1
You gave the boolean false
```

Exercise submission instructions

How to see the solution

You have reached the end of this section! Continue to the next section:

→ 5. Calculating with numbers

Remember to check your points from the ball on the bottom-right corner of the material!

In this part:

5. Calculating with numbers

6. Conditional statements and conditional operation

7. Programming in our society

Source code of the material

This course is created by the Agile Education Research -research group of the University of Helsinki.

Credits and about the material.



HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

MASSIIVISET AVOIMET VERKKOKURSSIT
MASSIVE OPEN ONLINE COURSES · MOOC.FI