Lesson 1A: Variables, class methods and class variables

Mark Eric Dieckmann

January 16, 2023

Revise lectures 2 and 3. Create the directory "Lessons" and then the package "lesson1a" in this directory. Create a new class **Lesson1A** with the method *main(..)* in this package. Test if your IDE works by placing the method call *System.out.println("Lesson 1A");* in the method *main(..)* and run the code.

Underline the text "Lesson 1A" that gets written to the console with "———". Do that in the same call of *System.out.println("Lesson 1A");*

1 Primitive variables, casting operators and constants

First we revise primitive variables (memory requirement known at compile time) and how the "+" operator works in *System.out.println(arg);*

Declare the double precision variable *d1* and write its content to the console. Can you do that? Then initialize it with the value 4.0 and write its content to the console.

Declare and initialize the integer variable *i1* with the value 10.

Modify the *System.out.println(arg);* such that your code writes "The double value is 4.0" to the console. Concatenate the text and the double value in the argument list.

Call *System.out.println(arg);* with the text "The sum of the double value and the integer value is " followed by the sum of *d1* and *i1*. Concatenate the values in the argument list. What happens if you try "text" + d1 + i1? How would you make sure in maths that the addition of both numbers comes first? Try that. What happens now?

Do another call *System.out.println(arg);* where you add up both numbers in the argument list and concatenate that with the text " is the sum of the double value and the integer value." What happens now and why?

Declare another integer *i2* and try assigning the value of *d1* to it. Cast the value from *double* to *int* just like in C++ with *i2=(int) d1;* Try that and write the result to the console with "Result of the cast is: " followed by the value of *i2*.

Declare a boolean variable and initialize it such that its value is true. Write out the result to the console.

We change a variable into a constant by using the keyword *final* before we specify the type of the variable. Write *final double d2 = 2.0;* followed by *d2=3.0;* Does this work? What happens if you declare the constant *d3* and initialize it on the next line? Does that work?

2 Static methods and overloading a method

You know how to convert a *double* into an *int* and vice versa. Implement a method *convert(..)* in the class **Lesson1A** that takes an integer argument *arg*, converts the value of *arg* into a double and returns that value.

Test your method by calling the method with the integer value 1 and writing out the result to the console.

Implement a second method with the same name, which takes a double precision argument *arg*, converts it into an integer value, and returns that value. Do the conversion on the same line of the method where you return the value.

Test your method by calling the method with the double precision value 4.0 and write out the result to the console.

3 A separate class

We want to implement some functions, which deal with integer numbers, double-precision numbers, and booleans.

Create a new class **Manipulations** in the same package as the class **Lesson1A**. This class should not have a method *main(..)*.

Copy and paste the two static methods *convert(..)* from your class **Lesson1A** to the class **Manipulations**.

Test both methods in your class **Manipulations** by calling one with the integer value 1 and the second with the double precision value 4.0.

We used the separate class as a container for static methods. Declare a reference variable in *main()* called *manip* of type **Manipulations**.

Try writing out the content of *manip*. Does that work? Set *manip* to *null* and write out the content of *manip*. What is *null*?

Create an instance of **Manipulations** and copy the result into *manip*. Write out the content of *manip*. What is the number you see on the console or what information does *manip* contain?

Write *manip.* and look at the pop-up menu. Click on the method *toString()*. What information do you get? Click on the methods you implemented. What information do you get for those?

The two methods of **Manipulations** are declared *static*. This declaration implies that they are **class methods**: they exist once for the class and are called with the class name. Introduce a class variable named *returnAnInt* of type *boolean* into **Manipulations**. Set its value to *true*. It should have a global scope for the class and be visible from outside the class. Declare another class variable *value* with a global scope of type double and set its value to 4.0. It should not be visible from outside the class.

Implement the class method *convert()* with return type void and an empty argument list. This method should write values to the console with *System.out.println()*. If the value of *returnAnInt* is true, it should write the return value of the method *convert(..)* with an argument to the console. Otherwise, it should write the value of *value* directly to the console.