Theory: Integer types and operations

34 minutes 5 / 17 problems solved

Start practicing

§1. Basic information about integer types

Java provides several types which represent integer numbers including positive, negative and zero. In practice, the most used types are `int` and `long`. The first type can store numbers from a smaller range than the second one, but it is often enough (especially, in this topic). You can perform all arithmetic operations (`+`, `-`, `*`, `/`, `%`) with variables of integer types.

Let's look at some examples below.

```
int two = 2;
int ten = 10;

int twelve = two + ten; // 12
int eight = ten - two;  // 8
int twenty = two * ten; // 20
int five = ten / two;   // 5
int zero = ten % two;   // 0, no remainder


int minusTwo = -two; // -2
```

This code demonstrates how to assign values to int variables as well as how to perform arithmetic operations with them. We hope that you already understand all operations well.

To improve the readability of your code, the special underscore character `_` can be used to separate groups of digits within a number.

```
int million = 1_000_000;
```

You may also print a value of an **int** variable:

```
int number = 100;
System.out.println(number); // 100
```

All arithmetic operations work with the `long` type as well.

```
long one = 1L;
long twentyTwo = 22L; // L or l is a literal for longs
long bigNumber = 100_000_000_000L;

long result = bigNumber + twentyTwo - one;
System.out.println(result); // 100000000021
```

If a number ends with the letter `L` or `l` it is considered as `long`, otherwise, it is `int`. We recommended you to use the uppercase letter `L` because the lower case letter `l` is very similar to the digit `1`.

Note, use `long`'s numbers only if it is really necessary (to process big values).

§2. The forms of the assignment operator

Suppose, you want to add some value to a variable. You may write something like this:

```
int n = 10;
n = n + 4; // 14
```

The assignment operator `=` has several forms which combine it with an operation to avoid repeating the variable twice:

Current topic:

- ✓ Integer types and operations — Stage 1

Topic depends on:

- ✓ Scanning the input — Stage 1
- ✓ Naming variables — Stage 1
- ✓ Arithmetic operations — Stage 1

Topic is required for:

- ✓ Floating-point types — Stage 5
- ✓ Relational operators — Stage 2
- ✓ Increment and decrement — Stage 1

Table of contents:

```
int n = 10;
n += 4; // 14
```

As you may see, this form looks more concise. There are a few other possible forms `*=`, `/=`, `%=` and some others.

§3. Reading numbers from the standard input

As a rule, to solve a problem you need to read some data from the outside world, process it, and output the result. The following program reads two numbers from the standard input, adds them, and prints the sum.

```
import java.util.Scanner;

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

        int a = scanner.nextInt();
        int b = scanner.nextInt();


        int sum = a + b;


        System.out.println(sum);

    }

}
```

This simple code uses `Scanner` to read data.

If we know that the input numbers can be quite large, we can read `Long`'s instead of `Int`'s:

```
long a = scanner.nextLong();
long b = scanner.nextLong();

long sum = a + b;
```

No more lines need to be changed in this code.

Now you have enough knowledge to write useful programs that process data. You may use the template above for solving code challenges in this lesson. Try to give meaningful names to variables when solving problems.