Double type --- real numbers

1. `double` type

Java provides the `double` type for working with real (fractional) numbers. It occupies `8 bytes` in memory (twice as many as the `int` type) and can store values in the range from `-1.7*10`^{308} to `+1.7*10`^{308}. For comparison: the `int` type can store a value in the range from `-2*10`^{9} to `+2*10`^{9}.

In real numbers, the fractional part is written after a decimal point. For example, 123.456, or 2.5, or 100.00, or 0.01. When computers deal with such numbers, we call them *floating point* numbers.

By the way, in addition to the `double` type, we also have the `float` type (which is only 4 bytes in size). Its name comes from *floating point*. And the name `double` comes from *double float*. A `double` is twice as large as a `float`: `8 bytes` versus `4`. It is also called a **double precision floating-point number**.

---

2. Creating a `double` type

The double type is used to store real numbers. To create a variable in code that can store real numbers, you need to use a statement like this:

```
double name;
```

Creating a `double` type

Where `name` is the name of the variable. Examples:

Statement	Description
`double price;`	A real `price` variable is created
`double weight;`	A real `weight` variable is created
`double lightSpeed;`	A real `lightSpeed` variable is created

As with the `int` type, you can use shorthand to create multiple `double` variables simultaneously:

```
double name 1, name 2, name 3;
```

Creating multiple `double` variables

And even immediately assign values to them:

```
double name 1 = value 1, name 2 = value 2, name 3 = value 3;
```

Creating and initializing multiple `double` variables

Examples:

Statement	Note
`double price = 5.0;`	The variable stores `5.0`
`double weight = 2;`	The variable stores `2.0`
`double x = 1.0, y = 2.0, z = 3.0;`	

3. Assigning integers and real numbers

It would be bad if integers could be assigned only to `int` variables, and real numbers — only to `double` variables. We want to be able to convert between the two kinds of numbers. And Java provides this ability.

First, both real and integer numbers can be assigned to `double` variables. When assigning integers, they are simply converted to real numbers. Of course, some accuracy may be lost in the process.

Statement	Note
`double price = 5.0;`	The variable stores `5.0`
`double weight = 2;`	The variable stores `2.0`
`int t = 1000;` `double x = t * t;`	The `x` variable stores `1000000.0`

Second, if an integer and a real number are involved in some expression, then the integer is first converted to a real number and only then is the operation with the other real number performed.

Statement	Note
`int t = 1000;` `double x = t * 5.0;`	The `x` variable stores `5000.0`
`System.out.println(5 * 2);`	The number `10` will be displayed on the screen
`System.out.println(5 * 2.0);`	The number `10.0` will be displayed on the screen

And finally, it is also possible to assign real numbers to `int` variables. When we do this, the fractional part of the real number is discarded — the number is rounded down to the nearest whole number.

Additionally, the compiler requires the programmer to explicitly document what is happening (to be sure that other programmers understand that the fractional part is being dropped). In general, such a conversion looks like this in code:

```
integer_variable = (int)(real_number);
```

Assigning a real number to an `int` variable

Examples:

Statement	Note
`int x = (int)(5.5);`	The `x` variable stores `5`
`double a = 5.999;` `int x = (int)(a);`	The `x` variable stores `5`

`double a = 5.999;`
`int b = 2;`
`int x = (int)(a * b);`
	The `x` variable stores `11`

4. Dividing integers and real numbers in Java

When dividing an integer by an integer, the remainder is always discarded. How then can we divide `5` by `2` to get `2.5`?

At first, it seems like the correct option is:

```
double d = 5 / 2;
```

But it's not so simple. The problem here is that the Java machine first calculates the value of `5 / 2` and only then assigns the result to the `d` variable. And the `5 / 2` operation is integer division. That means `d` will contain `2` or, to be more precise, `2.0`

The correct solution is to write **at least one** of the numbers involved in the division as a *real* number (i.e. with a decimal point):

```
double d = 5.0 / 2;
```

```
double d = 5 / 2.0;
```

```
double d = 5.0 / 2.0;
```

In each of expressions, `d` will contain `2.5`

But what if we are working with variables? What if we have code like this?:

```
int a = 5;
int b = 2;
double d = a / b;
```

There is a slick (and obvious) solution here — force the Java machine to convert variables to real numbers by multiplying them by one as a real number (`1.0`)

```
int a = 5;
int b = 2;
double d = a * 1.0 / b;
```

Note that multiplication and division have equal precedence, and are performed from left to right. That means that it matters where we multiply the 1.0.

Examples:

Statement	Order of execution	Result

`int a = 5;`
`int b = 2;`
`double d = 1.0 * a / b;`
	`(1.0 * a) / b;`	`2.5`

`int a = 5;`
`int b = 2;`
`double d = a * 1.0 / b;`
	`(a * 1.0) / b;`	`2.5`

`int a = 5;`
`int b = 2;`
`double d = a / b * 1.0;`
	`(a / b) * 1.0;`	`2.0`