Exercises module 4 – Data types

4.1 Float and double

1. Start a new project and just enter in the **main** method:

```
float f = 3.14;
```

You will get a compiler error, why? When you write 3.14 the number is interpreted a double and we can't store a double in a float without losing information.

If you would have written this instead it would have worked.

```
double d = 3.14;
```

Try to make the code compile for float as the data type without using **double**.

4.2 Handling money

1. We are working on a financial application and we for some reason need to add 0.1 to the amount 10 times. What is the result?

```
float amount = 0;

amount = amount + 0.1f;
amount = amount + 0.1f;
amount = amount + 0.1f;
amount = amount + 0.1f;
amount = amount + 0.1f;
amount = amount + 0.1f;
amount = amount + 0.1f;
amount = amount + 0.1f;
amount = amount + 0.1f;
amount = amount + 0.1f;

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

(hint, press CTRL+D to duplicate the current row in the editor)

Did you receive 1.0 as output as you might have expected? Why not?

2. Perhaps the reason is that float is not precise enough, let's use **double instead**, and does that help us?

3. The problem is that neither **float** nor **double** can represent **base 10 numbers.** Read this article for more details: http://stackoverflow.com/questions/3730019

Stretch task:

To solve this you can use the **BigDecimal** type or use an external library like: http://javamoney.github.io/

Lookup how **BigDecimal** work and use it instead in the calculation above online at: http://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html

Comparing **double** vs **BigDecimal**

Feature	Double	BigDecimal
Internal base	Base 2	Base 10
Precision	15-16 significant digits	Essentially unlimited
Range	± (~ 10^{-324} to 10^{308})	Essentially unlimited
Speed	Fast, hardware	Slow
Size	8 bytes	Vary

As BigDecimal has an essentially unlimited level of precision, it is much more accurate than a float or double.

The big takeaway here is that **you should never** use float or double to represent money related values.

4.3 Max and min values

1. To print out the maximum and minimum value for an integer variable you can write:

```
System.out.println(Integer.MAX_VALUE);
System.out.println(Integer.MIN_VALUE);
```

What is the output?

2. Read about the differences between **int** and **Integer** here: http://stackoverflow.com/questions/8660691

3. Try to get the max and min values for **byte**, **short**, **long**, **float**, **double**

4. What happens

4.4 Number of bits used to represent a type

1. To print out the number of bits used to represent a type, we can use the SIZE property:

```
System.out.println(Byte.SIZE);
```

What is the output?

Try to get the number of bits needed for **short**, **integer, long**, **float**, **double**

4.5 Type conversion

1. Without writing anything special Java allows us to pass data from smaller types to bigger types like: (implicit conversion)

```
byte x = 100;
int y = x;
long  z = y;
float f = z;
double d = f;

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

(To avoid typing **System.out.println** all the time, you can instead type **sout + TAB**)

Try to do the reverse? Will that work? Can you store a long in an integer?

We can always convert between types as long as don't lose any information or precision. We will look at how to convert from **double-> int** and similar conversions later in the course.

4.6 Overflow

1. Given this program, what is the output? Try to reason about it before you run it:

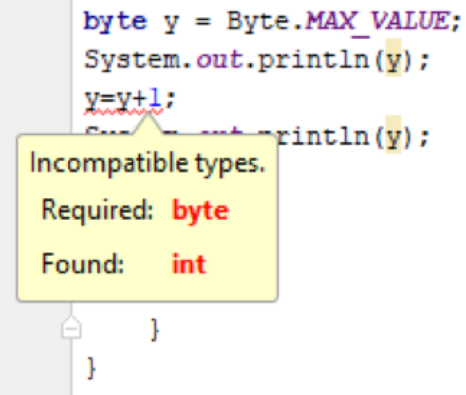
```
int x = Integer.MAX_VALUE;
System.out.println(x);
x=x+1;
System.out.println(x);
```

2. Read more about overflow here: https://en.wikipedia.org/wiki/Integer_overflow

3. Do the same using a byte instead, does that work?

```
byte y = Byte.MAX_VALUE;
System.out.println(y);
y=y+1;
System.out.println(y);
```

4. Why is it complaining about an integer when we are only working with a byte?



5. The problem is that the expression (y+1) is becomes an integer by the compiler. To solve this, we need to convert the integer expression to a byte by doing:

```
y=(byte)(y+1);
```

Run it and explore the output, what is the next value after 127?

Questions and concepts to study further on your own:

- Binary vs octal vs hexadecimal numbers
- Float vs double
- What is IEEE 754
- How much comments should we have? Can we have to many comments? To little?
    - Coding Without Comments
      https://blog.codinghorror.com/coding-without-comments/
    - “Comments are a code smell”
      https://softwareengineering.stackexchange.com/questions/1