Step 8 – Abstract classes and HashMap

Last time we learned about inheritance. This week we will continue in the same street - this time with abstract classes. And it's time to look at other options when it comes to holding many objects, right? HashMap!

Goals for this step:

- I can use abstract classes, and I understand what that means.
- I understand what aggregation (and composition) entails.
- I know how to use a HashMap.

No additional book chapters this time.

Here is the questionnaire for step 8 where you can tell how it went.

Task 1

Take the solution from the previous step as a starting point. You can choose whether you want to continue working with your own code or the solution proposal (in Canvas).

Let us imagine that the code we have written so far will be used to draw different geometric shapes. We do not care about the drawing itself, but we need objects that represent different figures. These occur in various forms (rectangles, squares and circles). It does not seem appropriate to create Shape objects, does it? Then we do not know what kind of geometric figure we are talking about.

Make sure we can not create Shape type objects.

Task 2

At the same time, we want to make sure that all different subclasses of Shape (current and future) offer methods that provide answers to what area and what circumference the figure has. Therefore, make the necessary changes to ensure this.

Task 3

Make sure all shapes have an id (data type int) that cannot be changed. Make a getter for id.

Task 4

Create 10 objects of different geometric shapes. Put these in a HashMap.

Task 5

Go through all the objects in the HashMap and print information about each object.

Task 6

Pick an object from the HashMap based on a key you know exists. Check that you got the expected object by printing information about the object.

ExtraTask 1

We have several ways to go through the elements of a HashMap. Take a look at this, and test the three ways described there. See if IntelliJ gives you possible recommendations for changes in syntax and try these out.

ExtraTask 2

Print information on all shapes that have an area larger than a certain limit.

ExtraTask 3

Create a method that returns all objects that have a perimeter above a certain size. Here you probably have to do a little research to make it happen in an elegant way. For example, you can try to use stream, which we will not get into in this topic. But fun to learn anyway 😊