Exercises Module 20 - Exceptions

Task 1

We need to check that the productId that is given to Movies when creating them is correct. The productId can not be a negative number, but right now there is nothing that stops this from happening. Try to create a Movie with a negative number as productId. Run the program and you will see nothing happens.

Solve this problem by adding a check in the constructor of the Movie class, that throws an IllegalArgumentException if the productId sent in as an input argument is a negative number. Add a message to the Exception so that any code that catches the exception can find out what happened. Try to run the program. What happens? Did the program crash?

Task 2

To prevent the program to crash when an exception is thrown we have to catch the exception. Put the code that can cause an exception to be thrown inside a try-catch statement and handle the exception by catching it. In the catch statement you can just print out the stacktrace by calling the printStackTrace() method on the exception, like this:

```
catch (Exception e) {
    e.printStackTrace();
}
```

If you now create the variables of the movies inside the try block, maybe you now have a problem with adding them to the product list if that code is outside the try block? You might have to declare the variables before the try block and only assign them inside the try block, or add the movies to the product list inside the try block. Find a solution that compiles.

Try running the program. Did it still crash?

Should you put all the lines that creates movies inside one try-catch statement, or have a unique try-catch for every movie that is created? What is the difference when running the program? Try it out!

Notice that there was nothing preventing you to throw the exceptions without also trying to catch the exceptions. That is because the IllegalArgumentException is not a checked exception.

Task 3

Now create a Book with a negative productId (add the productId to the constructor in the Book class if you have been creating books without the productId before). Nothing stops you from having a negative number as productId, but it should. Add a check in the constructor of the Book class that checks the input argument of productId, and this time throw an Exception instead of an IllegalArgumentException.

IntelliJ will tell you that you are throwing an unhandled exception. Since the Exception class is a checked exception it must be handled in some way. You could catch it in the constructor, or declare the constructor to throw an Exception so that code that uses the constructor knows that it might throw an exception and that this has to be handled in the calling code instead. Declare the constructor to throw an exception like this:

```
public Book(long productId, String title, String author, int price) throws Exception {
```

Now everything is fine with the Book class, but what happens in the main method?

Now we need to handle the checked exception in the main method instead. Surround the risky code in the main method in a try-catch statement, catching Exception and handling it by printing out the stacktrace.

If the ChildrensBook class uses the Book constructor you might need to add the

Now the program should run without crashing, and only printing out the stacktraces of books and movies that are being created with a negative productId.

Task 4

We need to create our own Exception to be able to save the incorrect number in the exception when it is thrown. This is so that we can print the incorrect number when catching the exception in the main method.

Create a new class with the name IncorrectProductIdException that extends Exception.

Create an instance variable of type long with the name incorrectNumber, and a constructor that takes a long incorrectNumber and assigns the value to the instance variable.

Override the String toMessage() method and return a message that tells that there is an incorrect number and also adds the incorrectNumber variable to the message.

Now, in the constructor of the Book class, throw the new IncorrectProductIdException instead of an Exception.

Use the new constructor and send the incorrect productId as an input argument to the constructor. There is no need to add a message to the Exception. Also change the signature of the constructor to throw an IncorrectProductIdException instead of an Exception. This makes the calling code only responsible for catching IncorrectProductIdExceptions and not all Exceptions.

In the main method, when creating new books with the constructor of the book with the productId check, surround this code with a try-catch statement and catch the IncorrectProductIdException. In the catch

block, add a System.out.println statement printing ot the message from the exception like this:

```
catch (IncorrectProductIdException e) {
    System.out.println(e.getMessage());
}
```

Try to run the code. Do you see the message from the exception? What do you think of using your own exceptions and having the possibility to add your own variables that you can store in the exception? Where could this be useful?