Programming Basics
Prof. Dr Silke Lechner-Greite
WiSe 2020 / 2021

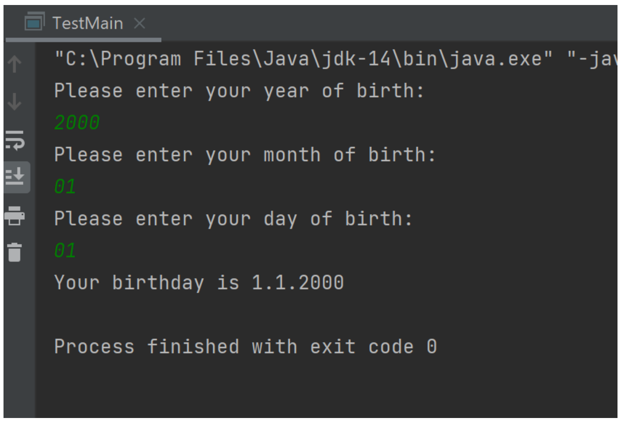
Technische Hochschule Rosenheim

Exercise 12 – Exceptions

Task 1 – Reading from the console

In the following, you implement a programme that asks the user for their year, month and day of birth, and then outputs the date of birth to the console.

a) Implement a `Year` class that has exactly one integer attribute called `year`, which should be initialised by the default constructor with the value 0. Add another appropriate constructor, as well as the `access methods` required for complete data encapsulation.
b) For the Year class, declare a static method `requestBirthYear`() that prompts the user to enter their year of birth, then reads the value entered from the console, and returns the value read in.
(Implementation note: for reading, use the `Scanner` class, as well as the `Integer.parseInt()` method.
c) In the same way as subtasks (a) and (b), implement a `Month` class with the attribute `month`. Similarly, add the related access methods and appropriate constructors. Also declare a static method `requestBirthMonth()` that prompts the user to enter their month of birth, then reads the value entered from the console, and returns the value read in.
d) In the same way as subtasks (a) and (b), implement a `Day` class with the attribute `day`. Similarly, add the related access methods and appropriate constructors. Also declare a static method `requestBirthDay()` that prompts the user to enter their day of birth, then reads the value entered from the console, and returns the value read in.
e) Write a `TestMain` class in which you create an object for each of the classes `Year`, `Month` and `Day`. Use the respective `requestBirth<DateElement>`() method to read from the console the three values that are passed to the constructor as initialisation parameters. A test sequence might look like this if the user enters their data correctly:



```
"C:\Program Files\Java\jdk-14\bin\java.exe" "-ja
Please enter your year of birth:
2000
Please enter your month of birth:
01
Please enter your day of birth:
01
Your birthday is 1.1.2000

Process finished with exit code 0
```

Task 2 – Using predefined exception classes

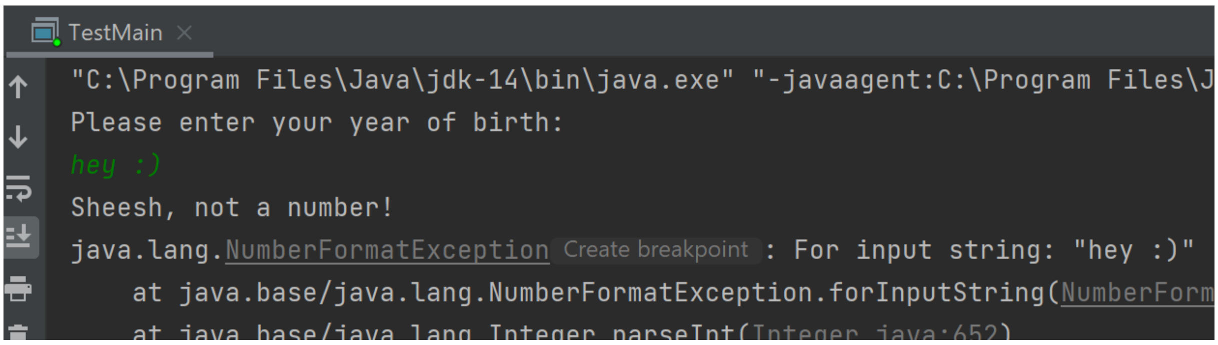
Now modify the `requestBirth<DateElement>()` methods of your classes for each data element, to meet the following requirement:

If the user does not input a valid number for any of the entries, an appropriate error message should be displayed. The user should then be prompted again to enter the desired value. This procedure repeats until the user has successfully entered a valid number for the value sought.

Base your local exception handling on the predefined `NumberFormatException` class. In addition to the message to the user, you must also produce a corresponding log of the exception that has occurred to the standard output.

(Implementation note: use the *`printStackTrace`* object method with the call argument *`System.out`*.)

For example, the output of your programme might now look like this:



```
"C:\Program Files\Java\jdk-14\bin\java.exe" "-javaagent:C:\Program Files\J
Please enter your year of birth:
hey :)
Sheesh, not a number!
java.lang.NumberFormatException : For input string: "hey :)"
    at java.base/java.lang.NumberFormatException.forInputString(NumberForm
    at java.base/java.lang.Integer.parseInt(Integer.java:652)
```

Task 3 – Creating user-defined exception classes

Until now, the programme only validates user input in terms of syntax, but not in terms of the values entered. For example, it is possible to enter 42.14.1977 as the date of birth, although this date does not exist at all.

a) First, for the `Month` class, when the month of birth is entered, check whether the month value m read in is 1 <= m <= 12. If this condition is not met, a suitable exception should be triggered. To do so, implement a new simple exception class called `IllegalMonthException`. Catch this exception locally and handle it in the following way: the user should be notified that the value for month must be between 1 and 12. Also, the exception that has occurred should be logged in the same way as in Task 2. In addition, the user should be given the opportunity to correct their incorrect entry.

b) Similarly, for the Day class, when the day of birth is entered, check whether the value read in, taken together with the values already read in for month and year, results in a valid day or not. In order to determine this, the `requestBirthDay()` method must know the values for the month and year of the date of birth. Therefore, extend the signature of the method by two call parameters, as well as all places in your programme where this method is called.
Since this test is relatively time-consuming due to the different lengths of February in leap years and "normal" years, you should put it into a separate method called `validateDay()`. This method is passed all the values needed to verify the correct date composition (value for day, value for year and value for month).

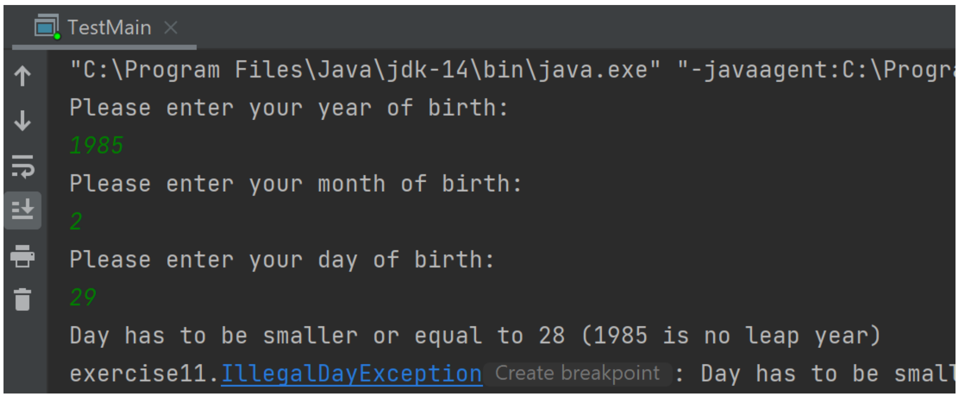
If the day of birth entered, taken together with the month and year values, does not result in a valid date, an appropriate exception should be triggered. To do so, implement a new exception class called `IllegalDayException`, which passes error-specific information. Catch this exception locally and handle it in an appropriate way. When doing so, allow the user to correct their incorrect entry.

Implementation notes:

- You can use the following code to check February (leap year):

```
if (day > 28 && (birthYear % 4 != 0 || (birthYear % 100 == 0 && birthYear % 400!=0)))
    throw new IllegalDayException("The day must be less than or equal to 28.");
```

The output of your programme might look like this:



```
"C:\Program Files\Java\jdk-14\bin\java.exe" "-javaagent:C:\Progr
Please enter your year of birth:
1985
Please enter your month of birth:
2
Please enter your day of birth:
29
Day has to be smaller or equal to 28 (1985 is no leap year)
exercise11.IllegalDayException Create breakpoint : Day has to be smal
```