SOEN 6011 : SOFTWARE ENGINEERING PROCESSES

SUMMER 2022

ETERNITY

PROBLEM - 4

Error Handling, Debugger and Quality Attributes

Author

Neona Sheetal Pinto

https://www.overleaf.com/project/62ec550934d66385ad8d07ee

Contents

List of Figures

1 PROBLEM 4 - F6: $B(x, y)$

1.1 Error Handling

Error handling is a way of responding and recovering when an unexpected situation occurs. Error handler in a program is triggered when there is any unexpected error condition in the program. Error handling helps the user to perform activities without any crash or screen freeze.

In the project, for the Beta function, if there is any input which is negative, the program using error handling techniques will notify the user by sending an error message asking the user to enter the right inputs in figure 2. On entering wrong format of inputs like strings or other inputs which is not compatible to double values which is in the domain of real numbers, the user is notified to enter the valid inputs. There is also a check to ensure that the user enter correct choices for the startup menu in figure 1.

```
try {
    statements-1
    ...
    numericInputCheck(inputDataString)
    ...
}
catch ( NumberFormatException exception) {
    statements-2
}
```

numericInputCheck(x, y) will provide an error message of type.
NumberFormatException when the input will not be an valid number.
runFunction() will check for both menu options and ensure that the x and y are compatible with double values.

In this project, the error handling is handled by try..catch method in most cases.

1.2 Debugger

1.2.1 Description:

In this project, the debugger used is a built in IntelliJ debugger. A debugger is a tool for finding all the bugs and trying to identify what's happening in the underlying parts of the code. To debug, set breakpoints in the code where you want to pause the program's execution and examine the program's state and behavior. On setting the breakpoints, you are all set to debug. You'll find the results in the Debug tool window in figure 3. [2]

1.2.2 Advantages

- In IntelliJ debugging mode, function's variable can be changed.
- It provides Breakpoints which cane be added easily by pressing on the left side of the code which also increase the efficiency of debugging.

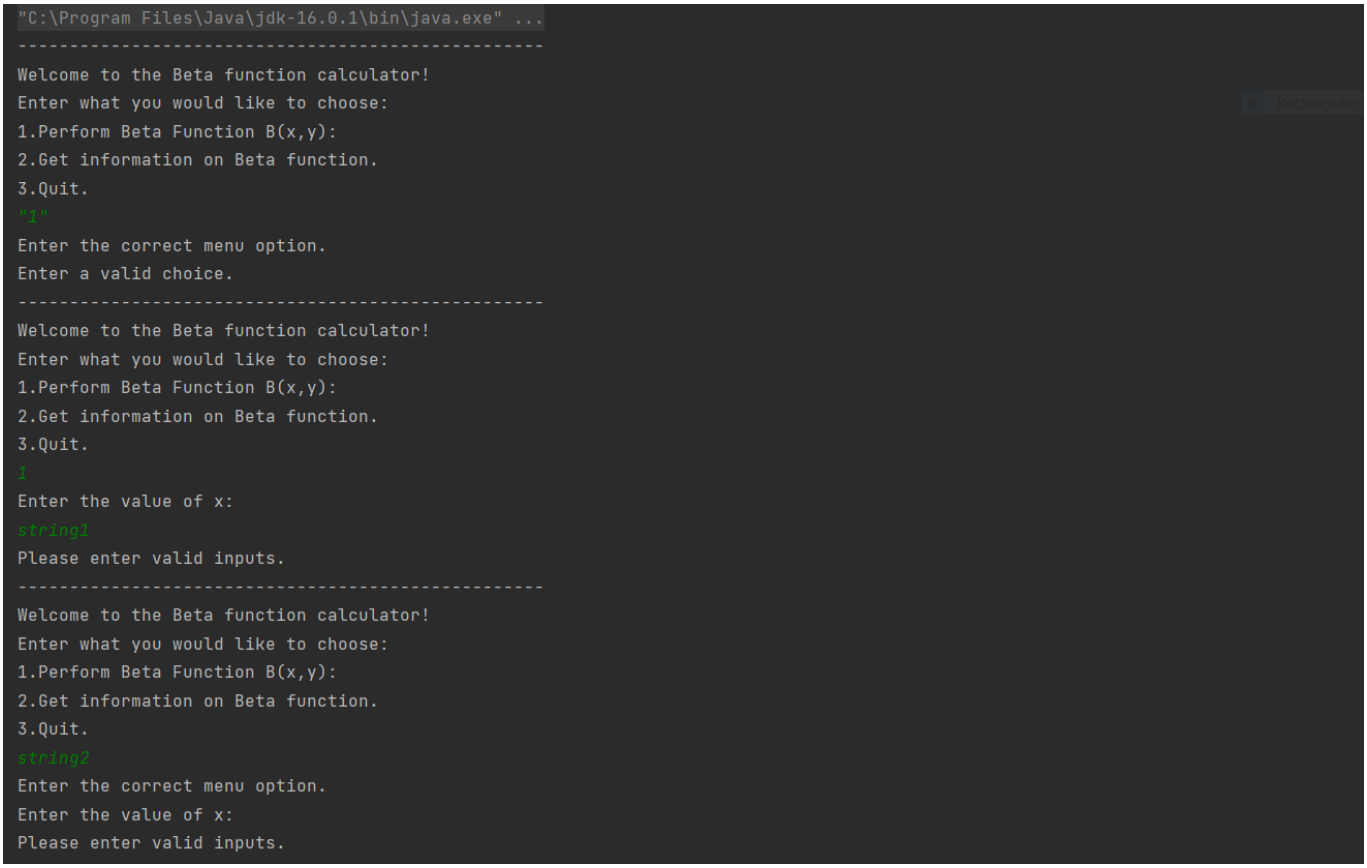

Figure 1: Error Handling for wrong menu option

- By using IntelliJ debugger, one can debug not only in java but also in many other languages.
- When you debug your code, the IDE shows you variable values in the source code next to their usages and lets you change the values. You dont need to refer to only to the debug window for doing the same.
- You can also use the Smart Step-Into action features to debug the specific method call you want.
- The IntelliJ has good debugging support.
- Easy to evaluate indiviual terms of series and check for accuracy and see whats going wrong, and where the approximations are slowly causing the result to become inaccurate.
- Evaluation of mathematical expressions are easier.
- Extremely customized break point features[3].

1.2.3 Disadvantages

- Some programs have multi-threading, which makes it more complicated to control the program.
- The debugger can be slow and can take time to step into functions and jump as compared to other IDE's.
- Makes the debugging of function which have recursion very hard to keep track.
- On using manual suspension instead of breakpoints, the debugger can get frozen and not work as required.
- Steep learning curve for some users
- Debugger window tool can get a bit overwhelming and and inefficient to use with too many options in User interface.

1.3 Programming style

The different programming styles are shown in the mind map above in figure 4. The Eternity project uses the Object-oriented paradigm using its key features like Data abstraction, Encapsulation and modularization. Other good coding practises have been followed throughtout the project

1.4 Quality Attributes

- **Correctness:** Ensuring to use the correct Exponent function to ensure to get the accurate answers in the definite integral. The taylor series for approximating the power function , has a test with different number of iterations ranging from 10 to 100. Based on the tests conducted, the conclusion is that to have minimum difference between expected output and actual output , the number of iterations needed is 13.
- **Efficiency:** As the project eternity, the beta calculator in general, uses many repetitions of mathematical equations which increases the program execution time. As the algorithm mentioned in this paper does not use any complex function structure, it does not show any complexity and can compile quickly.
- **Maintainability:** Usage of javadoc, which provides both readability and maintainability, and can be used for future references. The project keeps in mind the separation of concerns and has kept all the functionalities as modular as possible which helps to be reused or refactored without breaking any code. Unit test cases are also provided to keep existing functions as before.
- **Robustness:** Error handling, which shows appropriate messages on entering wrong inputs, keeps the project from crashing and can ensure smooth experience for user without any interruption.
- **Usability:** The project is in an executable jar file so that other users can use my file without any difficulties. The user need not worry about the implementation details. The Command line interface also provides step by step instructions on performing the Beta calculations.

1.5 Checkstyle

Checkstyle - plug-in for IDE's is used to check the quality of any source code. Checkstyle is mainly used to improve the formation of the code and ensure the code follows good programming practices. It has Sun and Google-style support, and I choose the Google-style support. It can automatically generate the issues of the code. Most parameters and constants were subsequently changed to be final since they are not modified in the function body, lines were split to not exceed too many characters, and an else statement was moved to the same line as the preceding bracket. The class was also given an empty private constructor to ensure it could never be instantiated due to it being a static utility/library class.[4]

1.5.1 Advantage:

- Checkstyle is mainly used to improve the formation of the code. It has a simple installation process. It can automatically generate the issues of the code. It helps the programmer adhere to good programming and coding practises.
- The Checkstyle has good support in case of any errors during installation.
- It automates the process of checking Java code to spare humans of this boring (but important) task.
- This makes it ideal for projects that want to enforce a coding standard.
- Checkstyle is highly configurable and can be made to support almost any coding standard.
- Checkstyle is portable across different IDE's like Eclipse and IntelliJ.

1.5.2 Disadvantage:

- There is less space between the 'error,' 'warnings' and 'description,' 'resource' and 'path,' less space, making it challenging to understand the problem correctly. Can be ambiguous in some situations.
- The only two noticeable annoyances were the inconsistencies in style being labelled as *errors* instead of *warnings*, and that the way the lines with these *errors* were highlighted by Eclipse and/or Checkstyle made it difficult to find issues like an extra space.
- Checkstyle can be very efficient but in the mean time can also suggest code changes which are not of high priority. And hence can be a waste of time reorganising code.

2 Annexure:

- **Trello Board :** *https://trello.com/eternity119*
- **Code Version Control :** *https://github.com/neonapinto/Scientific_calculator*
- **Overleaf :** *https://www.overleaf.com/project/62ec550934d66385ad8d07ee*

References

[1] Concordia Logo : The logo of Concordia University with transparent background
https://www.pngwing.com/en/free-png-djpzn

[2] IntelliJ Features
https://www.trustradius.com/products/intellij-idea/reviews

[3] JetBeans Features
https://www.jetbrains.com/idea/features/

[4] Checkstyle Features
https://stackoverflow.com/questions/26955766/intellij-idea-checkstyle

```
1.Perform Beta Function B(x,y):
2.Get information on Beta function.
3.Quit.
1
Enter the value of x:
-3
Enter the value of y:
-4
Please enter positive values.
Please enter valid inputs.
--------------------------------------------------
Welcome to the Beta function calculator!
Enter what you would like to choose:
1.Perform Beta Function B(x,y):
2.Get information on Beta function.
3.Quit.
1
Enter the value of x:
0
Enter the value of y:
0
Please enter positive values.
Please enter valid inputs.
--------------------------------------------------
Welcome to the Beta function calculator!
Enter what you would like to choose:
1.Perform Beta Function B(x,y):
2.Get information on Beta function.
3.Quit.
```

Figure 2: Error Handling for wrong inputs

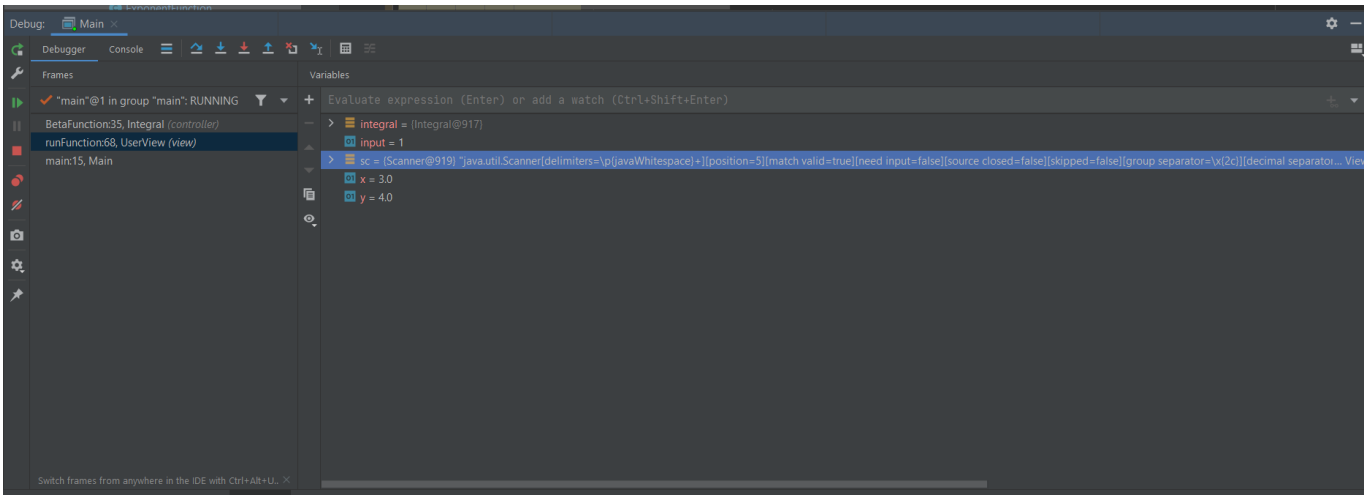

Figure 3: Debugging using IntelliJ debugger

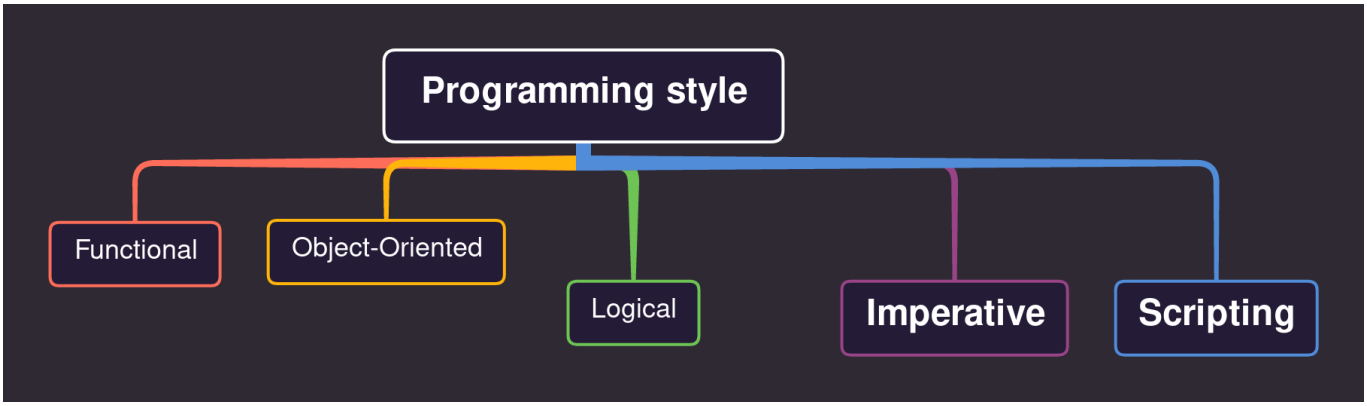

Figure 4: Mind map for Programming style decision