Exception Handling for

Master of Science in Information Technology

Software Design and Programming

Joseph DiBiasi, Estell Moore, James McKenna, Anthony Martuscelli

University of Denver College of Professional Studies

9/23/2025

Faculty: Nirav Shah, M.S.

Director: Cathie Wilson, M.S.

Dean: Michael J. McGuire, MLS

Abstract

This portion of our project involved the design and implementation of a new class we had to derive called MyDate. This was to support the manipulation of the already developed class JulianDateConvertor. The development and testing of not only making a new class that derives from JulianDateCovertor, but also doing the testing on this with JUnit proved to be a little challenging with the instructions and templates provided; however as a group we discussed best approaches and learned more about basic object oriented principles through Java by developing this custom class in this assignment.

CONTENTS

Introduction

When looking at the basic templates that were provided for our first assignment and understanding a custom class and a new unit test was required to be developed, this assignment proved to be challenging at the start. However, through team work and asking other team members who had more knowledge and experience with Java this ended up becoming a great learning experience. A big emphasis was required to develop and test our new class and develop error handling to prove that our functionality worked with what we developed. Through the already existing functionality, basic Java principles were required to develop and make this new MyDate class that would require the use of getters, private internal functions that would convert our julian dates and grab integers from our other existing class that was provided by JulianDateConverter. This was a huge first coding assignment as it not only demonstrated the basic object oriented programming concepts that are required for Java, it also showcased other principles for polymorphism and developing tests with error handling that would be harder developed and covered in future classes or from previous experiences. Moving forward however this next section will discuss our focus on Error Handling and why it's important.

Emphasis on Error Handling

When developing software, it’s important to keep track of the functionality of code through testing and proper error handling. If error tracking is insufficient in newly implemented code, then debugging to identify the root cause of the issue is exceedingly difficult and tedious. By implementing exceptions, recovery from failing code becomes easier to trace failures back to their direct causes, thus simplifying the debugging process. However, it’s important to note the

difference between simply catching an exception versus identifying the exception in the code with proper exception types and error messages. This approach not only improves clarity during debugging but also supports the long-term scalability and maintainability of the code as new features are introduced.

Error handling is also important in supporting data integrity and security. If errors are handled appropriately, these exceptions can validate data being processed or stored. For example, when working with databases, it's important to validate the data before being stored to prevent corruption, and preserve reliability (Mucci, Tim, Stryker. 2025.). During validation, additional measures can be incorporated to support data security by encrypting the verified data, and authenticating access to it. Error handling can be implemented to ensure that if a data is trying to be accessed by an unauthorized user, the system would instead reject the request rather than exposing sensitive information.

Cost and Benefit

Does this potentially heavy cost in development time ever outweigh the benefit? It is hard to accept that any serious company would ever disregard the benefits from handling all exceptions properly if they could. That is because it is cheaper to invest the time and effort in resolving errors as soon as they are discovered. A defect headed off by proper planning in the design phase costs a fraction of what it does to fix that same defect after it hits production. In other words, the cost of a bug grows exponentially as the software progresses through the SDLC. (Tamas Czer. 2025.) Defects caught in the design phase are the cheapest to fix, but the

cost scales quickly as the same defect caught in testing is approximately fifteen times more expensive; if this defect makes it all the way to maintenance phase it could be as much as 100 times more expensive. This means that while it is certainly painful to redo a run of regression testing because of a defect found late in the process, the cost is small compared to that defect reaching the maintenance phase. Once it hits the final step in the SDLC, there will be a minimum of the development, testing, and deployment cycles needed to fix the defect but there could be even more involved depending on the nature of the error. There could even be a potential user impact from a defect in production that requires additional development hours to resolve. Handling errors early is crucial to save the business not only monetarily but also in customer perception by providing a smooth user experience.

The argument can be made that there is a difference between handling an exception perfectly and handling an exception properly. Handling an exception perfectly hints at the expectation the system is able to recover from such an error or avoid it in the first place. This is not always possible or even desired. Sometimes if a system encounters an exception the best possible handling is going to be failing on the exception and providing a message that can help in troubleshooting. Cases like this involve less benefit, but there is also less cost in development time. Maybe the savings in cost here can be used to better handle the issue upstream; reducing the likelihood of the exception occurring in the first place. This means that while there are instances error handling may be of less importance; a company that shies away from the cost of error handling entirely does so at their own peril.

Conclusion

Overall, this assignment showed the team that class creation goes well beyond just writing and submitting code. It requires planning, testing, and making sure the code can handle issues when they (inevitably) arise. There were some overwhelming moments during the creation of the new class, but collaboration and communication proved to be our greatest asset. Error handling stood out as one of the key lessons during the week. While it takes added time to add exceptions and correctly test, it saves time and ultimately makes bugs easier to find and fix. In real-world development, correctly handling errors will provide a return on the investment of the time put in. In the end, it will create a smooth experience for users and ultimately save time and money. This week was a good reminder that development isn't about just running code. It's about building something with reliability that can be constantly improved and maintained.

References

Mucci, Tim, and Cole Stryker. 2025. “What Is Data Integrity?” IBM. July 22. https://www.ibm.com/think/topics/data-integrity.

Tamas Cser. 2023. “The Cost of Finding Bugs Later in the SDLC” Functionize. January 5. https://www.functionize.com/blog/the-cost-of-finding-bugs-later-in-the-sdlc.