**Team Assignment #3 - Write Up**

Master of Science:

Information and Communications Technology

Michelle Agustin, Hamdi Ali, Kalika Browder, Jake Collins

University of Denver University College

Date: 09/27/2025

Faculty: Nirav Shah, M.S , MBA

Director: Cathie Wilson, M.S

Dean: Michael J. McGuire, MLS

**Team Discussion Summary**

In software design, error handling is crucial because it ensures that programs behave predictably in the face of unexpected inputs or system conditions. Software may abruptly terminate, expose confusing technical errors, or even corrupt data if exceptions are not handled properly. In contrast, when developers anticipate and manage these situations, they help users fix problems by informing them about what went wrong and providing them with a path to correct it. In this way, the software is more reliable, usable, and trustworthy.

As part of our implementation of the MyDate class, our team focused on defensive programming practices. Before converting dates into Julian numbers, the constructors validate inputs to ensure they are realistic. It checks the provided values against the correct ranges for each month, for example, to prevent invalid entries such as "month 15" or "day 32". Instead of producing silent errors or undefined behavior when a user attempts to create a date that doesn't exist, the program responds with an IllegalArgumentException message. As a result, the program is fault-tolerant and user-friendly.

In addition, we recognized that exception handling is not just about catching errors after they occur, but also about anticipating potential problems. A proactive approach aligns with the idea of building resilient software systems that are stable under a variety of conditions. Because of this, our MyDate class combines validation, clear error messages, and graceful recovery.

Meanwhile, our team discussed error handling's tradeoffs. Comprehensive validation and exception logic require additional developer time, increase code complexity, and can have a negative impact on performance. A high-performance environment, such as a gaming engine or real-time monitoring system, may suffer from excessive checks that cause the performance to slow down and negatively impact the user experience. It may be appropriate for developers to handle errors more lightly in those cases. In mission-critical systems, such as healthcare software, financial systems, or aviation control software, the cost of skipping rigorous error handling far outweighs its performance benefits. The safety and reliability of the system must come first in those situations.

Ultimately, our team concluded that error handling should be proportional to the application's context and risk level. An everyday utility such as a date-handling class benefits from thorough validation because it prevents incorrect data and avoids user confusion. For both developers and end users, exception handling contributes directly to software that is more stable, maintainable, and trustworthy. Developers need to weigh the costs and benefits carefully. In high-stakes or production-grade applications thorough error handling is clearly worth it. In performance-sensitive or short-lived projects, excessive handling may create more problems than it solves. The key takeaway is to apply error handling judiciously, aiming for software that is both safe and manageable without unnecessary complexity.

**References**

GeeksforGeeks. 2023. “Exception Handling in Java.” GeeksforGeeks. [Accessed September 25, 2025. https://www.geeksforgeeks.org/exceptions-in-java/](https://www.geeksforgeeks.org/exceptions-in-java/)