COMP249 Tutorial 6: Exception Handling and File I/O & Serialization

Ella Noyes

July 28, 2024

1 Exception Handling

We looked at handling IO exceptions last week, now here's what the Java exceptions hierarchy looks like. When using a try-catch statement, you can catch an exception as an exception of its superclass (e.g., you can catch a FileNotFoundException as IOException, or even as an Exception). But we typically try to catch the most specific type of exception that we expect will occur so that we can handle it appropriately.

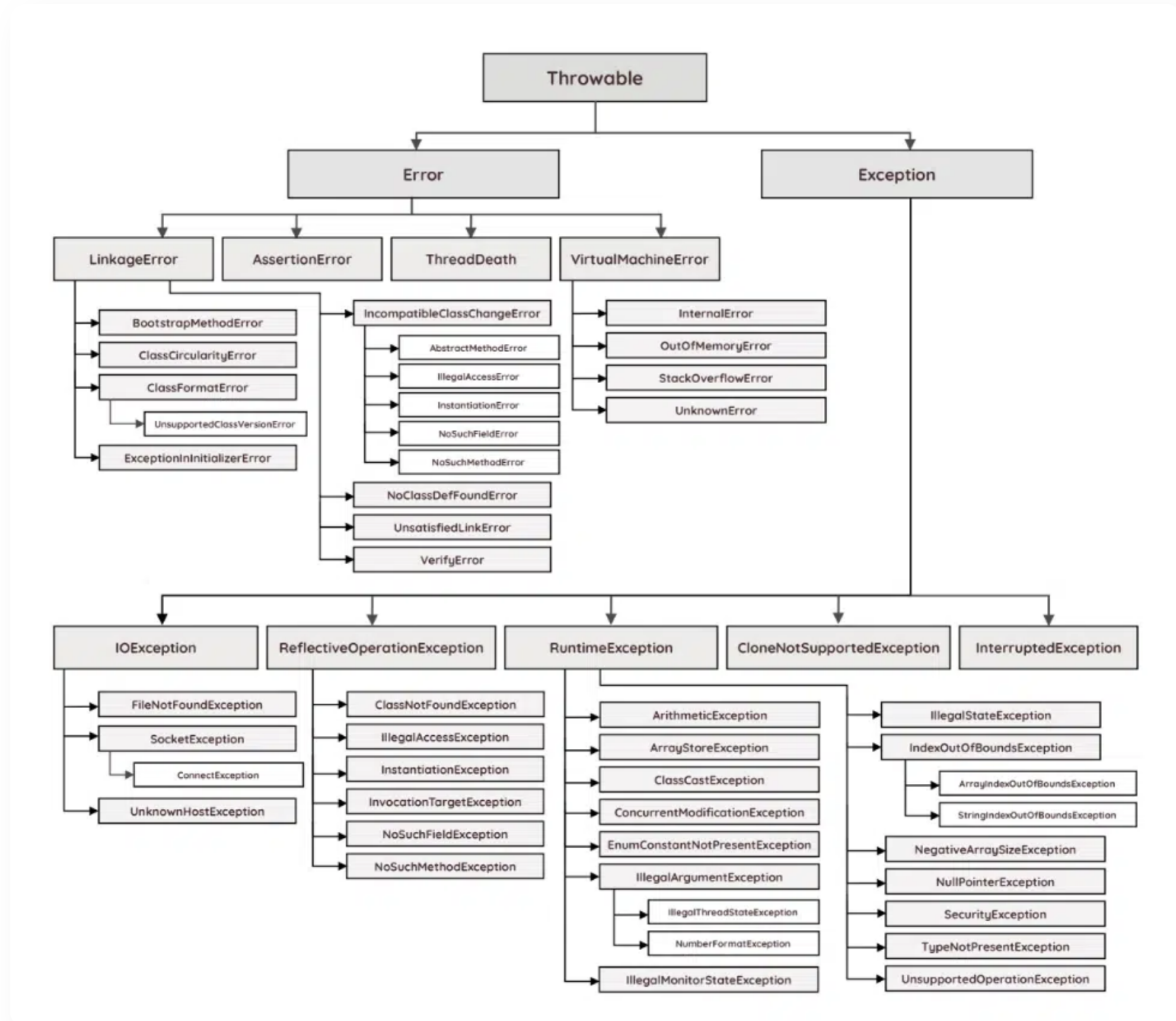

Figure 1: Diagram is from: https://rollbar.com/blog/java-exceptions-hierarchy-explained/

1.1 finally Block

In a try-catch statement, a finally block can be placed after the last catch block. A finally block executes regardless of an exception being thrown and can be used to perform necessary cleanup. Let's see a code example of that.

1.2 User-Defined Exceptions

You can define your own exceptions by extending Exception. We'll see an example of that from Tutorials Point.

2 Serialization

Serialization is the process of turning an data structure or object such that it may be stored or transmitted, then reconstructed to its previous state later.

> When modern computers work with data, it's often held in all manner of complex formats, full of internal relationships and references. When you want to write this data down, whether to share it or to store it for later, you need to find a way to break down those relationships, explain the references, and build a representation of the data that can be read from start to finish. This process is called serialization. [1]

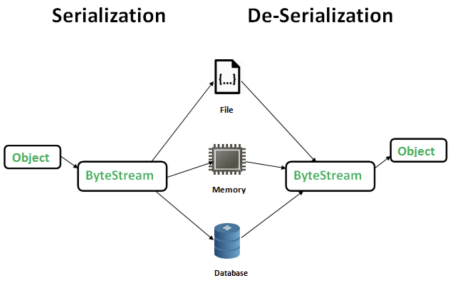

Figure 2: Diagram is from: https://www.geeksforgeeks.org/serializable-interface-in-java/

In Java, you can make objects of a class serializable by having that class implement the Serializable interface. Serializable doesn't have any methods or fields to be implemented, it just marks that class as Serializable. You can then serialize and deserialize objects of that class using an ObjectOutputStream and an ObjectInputStream, respectively.
Let's look at an example of that.

3 Programming Problem: Grocery Store Inventory Management System

These are my ideas combined and formulated by ChatGPT :)

Overview

You are tasked with developing a comprehensive inventory management system for a grocery store. The system must efficiently handle product data, including adding products, saving and loading product information, and managing stock levels. The system should also handle special cases where stock levels fall below a critical threshold.

Requirements

1. **Product Class**:
   - Implement a `Product` class that represents an item in the grocery store.
   - The `Product` class should include the following attributes:
     - `name` (String): The name of the product.
     - `price` (double): The price of the product.
     - `quantity` (int): The quantity of the product available in the inventory.
   - The `Product` class should implement the `Serializable` interface to allow instances to be written to and read from a file.
   - Include a `toString()` method for displaying product details.
2. **Custom Exceptions**:
   - Implement a custom exception class `StockException` that extends `Exception`.
     - `StockException`: This exception should be used to handle scenarios where the stock level of a product falls below a critical threshold. (See how this can be done in my ExceptionHandling example program on Github).
     - The exception should include:
       - `product` (Product): The product that caused this exception
     - Include a constructor to initialize this field
3. **GroceryStore Class**:
   - Implement a `GroceryStore` class that manages a collection of `Product` objects (as an ArrayList or as a large array if you're not familiar with lists).
   - The `GroceryStore` class should have the following fields:
     - `storeName` (String): Self explanatory
     - `inventoryFile` (File): The file that will be used to keep this stock's inventory
     - `productList` (ArrayList or array of Products): A collection of the store's products
     - `CRITICAL_STOCK_THRESHOLD` (int): The minimum acceptable stock for a product (if a product's quantity goes falls this value, then the manager should order more!).
   - The `GroceryStore` class should provide the following functionalities:
     - **Add Product**: Add a new `Product` to the store's inventory.
     - **Check Stock Levels**: Verify if the stock level of any product is below a specified threshold and throw a `StockException` if it is. Handle this exception by incrementing this product's quantity by 50.
     - **Save Products**: Serialize the list of products to a file. Handle any potential `IOException`.
     - **Load Products**: Deserialize the list of products from a file. Handle any potential `IOException` and `ClassNotFoundException`
     - **Print Products**: Print the details of all products currently in the inventory.

4. **Main Application**:
    - Develop a `Main` class with a `main` method to demonstrate the usage of the `GroceryStore` class.
    - Perform the following actions:
        - Add several products to the store's inventory.
        - Check stock levels
        - Save the inventory to a file.
        - Clear the current inventory.
        - Load the inventory from the file.
        - Print the loaded inventory to confirm the data was correctly saved and loaded.
    - Ensure that the application handles and displays any `StockException` that occurs during stock level checks

Constraints

- Assume the file used for saving and loading products is named `products.txt`.
- The program should handle file-related exceptions and provide user-friendly error messages.
- Implement appropriate error handling for stock levels to ensure the program effectively manages inventory thresholds.
- The `Product` class must have appropriate getters and setters for its attributes.

Deliverables

- `Product.java`: The class file implementing `Serializable` for representing a product.
- `StockException.java`: The custom exception class for stock-related issues.
- `GroceryStore.java`: The class responsible for managing the product inventory, including methods for adding, checking stock levels, saving, loading, and printing products.
- `Main.java`: The main class demonstrating the functionality of the `GroceryStore` class and handling exceptions.

References

[1] K. Sitto and M. Presser. *Field Guide to Hadoop: An Introduction to Hadoop, Its Ecosystem, and Aligned Technologies.* O'Reilly, 2015.