Assignment 12/17/2025

Q1. What are the three major categories of exceptions in Java's exception hierarchy? For each category, explain: (1) Whether it must be handled at compile-time, (2) Common examples, (3) Best practices for handling.

- Errors:
  - 1) Not necessarily to be handled at compile-time.
  - 2) OutOfMemoryError / StackOverflowError
  - 3) Do not throw errors, use exceptions instead.
- Checked Exception:
  - 1) yes at compile time.
  - 2) IOException / FileNotFoundException / SQLException / NetworkException
  - 3) Should use try/catch. Do not throw all exceptions, only catch when doing meaningful thing. Also should contain some statements in catch block, even a log.
- Unchecked Exception:
  - 1) No, it's handled at run-time.
  - 2) NullPointerException / NullPointerException
  - 3) Should try to fix root cause, instead of catching them.

Q2. Explain the execution order of try-catch-finally blocks. If both the catch block and finally block contain return statements, which value will be returned? Why is it strongly discouraged to use return statements in finally blocks?

- First do "try". If catch any exceptions, go to "catch" and then "finally"; if no errors, then go to "finally" directly.
- The value inside finally block will be returned.
- It's discouraged to use return in finally because finally must be run at the end, so that if there's return, we may miss some actions or fail to catch errors in "catch" block.

Q3. What is the "catch scope should be from small to large" rule? Why must specific exception types (like `OrderNotFoundException`) be caught before general ones (like `Exception`)? What happens if you violate this rule?

- It means if we have multiple catch, then we should have small (child) exception first and large (parent) after. It is because if the large exception is at first, then the errors would be catched without comparing to the small (more specific) exception.
- Same reason as above
- If we violate the rule, then the large / general exception would be catched, and the small / specific exception wound't be recognized.

Q4. Compare `throw` and `throws`: (1) Where is each used in code? (2) What follows each keyword? (3) Provide one practical example demonstrating both keywords working together in a DAO-Service-Controller architecture.

- 1) "throw" is used to throw a new exception, while "throws" is used when declaring that there might be an exception in the place

- 2) new exception created would follow “throw” keyword, and “throws” must followed by one or more exception classes.
- 3) practical example:
    - DAO throws checked exception (SQLException)
    - Service declares it with throws and wraps into a custom exception
    - Service may also throw runtime exceptions
    - Controller handles service-level exceptions

Q5. What is try-with-resources syntax (introduced in Java 7)? What interface must a class implement to be used with try-with-resources? Explain the execution order when multiple resources are declared.

- Try-with-resources is a Java feature that automatically closes resources when the try block finishes — whether it completes normally or throws an exception.
- java.lang.AutoCloseable
- Created / opened in left-to-right order, and closed in reverse order (right-to-left)

Q6. When creating custom exceptions, how do you decide between extending `Exception` vs extending `RuntimeException`? Provide criteria for each choice and one example scenario for each.

- “Exception” is checked exception, it must be catched/thrown. So when we want the caller to force handle it, we extend “Exception”. For exemple, when we want to have a file parsing issue, we could make our FileParsingException to extend “Exception”, so that the caller could do multiple fixes with the error.
- “RuntimeException” is unchecked exception, so when we want to represent a programming mistake, we extend that. For instance, if we have a class requiring ages, then when the caller input a negative age, we want to throw a “RuntimeException” to inform the caller.

Q7. Explain the two important features of Enum: "Every element is in values" and "Every element is a constructor". How would you implement an Enum with a private constructor that accepts parameters?

- Every element is in values:
- Every element is a constructor: it means every element can directly create the value with a parenthesis, this is calling the provided / default constructor.
- We could add private keyword to enum’s constructor, so that we could not create new enum instance outside the enum definition.

Q8. Describe the popular Enum template pattern (Interface + Enum + Exception). What are its four components? How does using an interface type (`IErrorCode`) allow the exception class to accept multiple different enum types?

- Four Components:
    - An interface A
    - Enums implements the interface A. Enums have private constructor.
    - Enum should have private constructor.

    - An exception can aggregate the interface/enum.
- Because the exception depends on the interface, not the implementation, we could use the exception class to accept different enums.

Q9. Compare the three major Collection interfaces: List, Set, and Queue. For each, explain: (1) Ordering characteristics, (2) Duplicate element handling, (3) Most commonly used implementation class, (4) One typical use case.

- 1) List and Queue have orders, Set doesn't have order.
- 2) List and Queue allow duplicates, Set doesn't have duplicate element.
- 3) List: ArrayList, also has LinkedList or Vector. Set: HashSet. Queue: LinkedList, PriorityQueue.
- 4) List: a list of value where order matters. Set: a collection of unique items. Queue: Processing tasks in order.

Q10. Explain the difference between HashMap and Hashtable. Why is Hashtable considered obsolete? What are the modern alternatives for thread-safe Map implementations?

- HashMap: NOT thread-safe, since methods are not sychronized. It allows null keys / values. Usually faster.
- HashTable: Thread-safe, since methods are sychronized. No null key / value allowed.
- Hashtable is considered obsolete because it locks the entire map for every operation, also there is modern concurrent utility replacing hashtable.
- We can use ConcurrentHashMap to replace HashMap / Hashtable.