Beaconfire Inc, Home Work, Week1 Day5.
By (Ping) Nalongsone Danddank.
ndanddank@gmail.com
wechatID: ndanddank
Short Answer:

1. What is the Stream API in Java 8? What are the differences between intermediate and terminal operations for Stream?

-> is used to process collections of objects. Is a sequence of objects that supports some methods that can be pipelined to produce the result. It's not a data structure, it just takes input from the Collections, Arrays or I/O channels. They don't change the original data structure, only provide the result as per the pipelined methods.

-> Intermediate operations can be pipelined because lazily executed and returns a stream as a result. But Terminal operations return non-stream values of the result like object or collection or anything. mark the end of the end of the stream.

2. What is the Optional class in Java and why do we need it?

-> is a container object that used to contain not-null object, and used to represent null when it get the absent value. We need it to handle the object or checking null values or NullPointerException.

3. What is an exception and what is an error in Java? What are the differences between them?

-> Exception is an unwanted or unexpected event that occurs during the execution of program at runtime or compile time that we could handle by try catch block code, like arithmeticExceptiion. An error is the serious problem that mostly is thrown by JVM, like in a scenario which is fatal and no way for recover by application program, like OutOf MemeryError.

4. What are the types of exceptions in Java? What are the differences?

-> Checked exception, checked by the compiler at compile time like IOException. Unchecked exception, checked by JVM at run time, like NullPointerException.

5. How can we handle an exception in Java?

-> using try-catch block, "throw/throws" keyword and finally block.

6. What is a custom exception and why do we need to use it?

-> we can custom own exceptiion by extends Exception or RuntimeException to be subcalss of them. And using "throw" keyword to throw exception whenever we want to raise the exception.

7. What is the difference between the *final* and *finally* keywords?

-> "final" is use for let the available be the constrain or memory address on stack cannot be change. If put on the primative type of available, the value cannot modify.

-> “finally” is the code block to handle the situation like, the code in the block will be always executed before the program terminate or when program raise the exception that let program terminate, the code in the finally block is always executed.

8. Is the following code legal? Why?
try{...} finally {...}
-> Yes, because we can use the finally block with try block for process some code that are always executed before program termination or whatever happen.

9. Is there anything wrong with the following exception handler as written?
Will this code compile?
Try {...} catch(Exception e){...} catch(ArithmeticException a) {...}
-> The compiler will tell you that is wrong and tell you to delete “catch(ArithmeticException a)” off or move to above of “catch(Exception e)”. because the Exception class is super class of ArithmeticException, so when a exception occurs the “catch(Exception e)” will always catch it, and not going to ArithmeticException catch block.

13. Match each situation in the first list with an item in the second list.
a. Scenarios
i. int number = “four”; -> iii. compile error

ii. public void recursion() { recursion(); } -> i. Error

iii. A program is reading a stream and reaches the end of the stream marker. -> iv. no exception

iv. You write code to read from a file but misspelled the filename. -> ii. checked exception