PART I

1) For instance, if we instantiate an object and send it to method which takes that object type as parameter, and instantiate another object from same class in that method and finally try to assign new object to parameter object, input parameter object will assigned to new object in method, not the coming object. But in another method, if we send first object that we instantiate at the beginning to the that method and make for example object.setName, this will change object's name since it is about object's value. Java is always pass by value.

2) I know how immutability can be achieved in java, but I have not used it before. For explanation, it helps us to prevent object content changing after we initialize it once. To make a java class immutable:

   - We need to put "final" keyword in front of our class.

   - We need to define our all data fields as "private" and "final" so that they cannot be changed and accessed directly.

   - We should not create setter methods for that data fields. Getters can be created.

   So, after we define that class, if we instantiate an object from it in another class and print its content, we can see our assigned values. But after those operations, if we decide to make/add another operation and print content of object again, we see that our object remains same as it was originally created.

   After my researches on internet, as for why we use it, i learned that we can use it to achieve thread-safe state and simplicity. And since for example String class is immutable, if a client changes content of a string in our server, other clients would see modified version of that string.

3) While a library is a set of methods that we can use directly instead of writing them, framework is processes our code, so we commit our project to it.

4) It is a memory management system is provided by java to us. We don't need to clean memory (unused objects) like we do in C with this way. We can call out manually by using system.gc() but I have not used it before. In some resources I have read, they say it can cause fatal damages in wrong uses.

5) Memory leak happens when a memory created in heap without deleting it at the end of the work. Despite java has garbage collector, in internet resources, i saw that memory leak can occur in java also. One example is database result sets which we suppose to close it manually.

6) Long term support versions are released every 3 years. Non long term support versions are released every 6 months.

7) Stack is the memory where JVM stores methods and local variables. Also it is thread safe because in each created thread it will have its own stack. Heap is the memory where

JVM stores objects to allocate memory dynamically(in stack, allocated memory is fixed). But reference of that objects are stored in stack.

8) OpenJDK is created by both java community and oracle while OracleJDK is created by only Oracle. OracleJDK requires licence and pricing since OpenJDK does not.

9) @FunctionalInterface is used to restrict that functional interface's abstract method numbers. It can only have 1 abstract method.

10) Some of them are: Consumer, DoubleToIntFunction(also there are similar convertion interfaces), Predicate that we have seen on lecture, BinaryOperator…

PART II

Collection Pipeline is basically takes a sequence, list, set, map etc. as an input and gives an output according to desired operation by processing/filtering specific operations how we want it to be. Later in the paragraph, it basically shows some operations like grouping, filtering, mapping etc. that I write in the code which java equivalent of them.

In alternatives part, it says that we do not have to use collections. We can achive some output by using loops, comprehensions and nested expressions. But I think those are more complex and gives us bigger O(x) output.

Laziness helps us to achieve our goal by shorten the data to be processed so that we don't lose more time to get unnecessary data.

Parallelism helps us to use multi-threading so that we don't have to wait a thread to be processed in order to process another thread. This gives us more performance and time handling opportunity.

It is good to rely on unchanging objects especially in multi-threaded enviroments. In large amounts of data, copying pave the way us to encounter problems. If it becomes a problem then we can use immutable objects.

Debugging helps us to identify and solve problem by going step by step. As a final step we make sure whether problem is solved or not. This can achived by using debugger in IDE's or tests.

TEST RESULTS