Homework 1

1- *Pass by Value* : In this way the variable is just copied, the original does not change.
Pass by Reference : In this way the variable changes in memory.

In Java always is used pass by value, the original variable does not change.

2- *Immutability* : It means classes which not changeable its content after creating objects. We create them one time but can not change.

If we want to create immutable classes, we can define with final keyword to class, define with private to all fields, provide to assign the variables in constructor firstly or not create setter methods.

3- In general the aims of both library and framework are same. Both of them is provide us to use codes that developed by others. Frameworks tell us how to use these codes, we are free to using libraries.

4- Space occupied by unused objects is freed up in memory by Garbage Collector.

5- A *Memory Leak* is a situation when there are objects present in the heap that are no longer used, but the garbage collector is unable to remove them from memory and, thus they are unnecessarily maintained.

Memory leak occurs in these situations in Java

- Heavy use of *static* variables
- Forgetting to close a stream or connection
- Not to writing proper overridden methods for *equals()* and *hashCode()* methods
- Creating inner classes that reference outer classes
- Use of finalizers
- Reading huge massive String objects and call intern() on that object
- Using threadlocals

6- New Java versions release in Oracle every 3 years, in Open every 6 months.

7- Stack and heap are logical fields in RAM. Methods, parameters that sent to methods,local primitive types and local references are kept in the Stack. Objects,class variables and instance variables are kept in the Heap.

8- Oracle releases new versions every three years, open releases every six months. Oracle licensed under Oracle binary code license agreement, Open has the GNU General Public License. Oracle has Flight Recorder,Java Mission Control and application class datasharing features, Open has the Font Renderer feature.

9- It is is used to ensure that the functional interface can't have more than one abstract method.

10- Predicate,Lambda expressions,BinaryOperators, BiFunction, ToDoubleBiFunction, ToIntBiFunction, and ToLongBiFunction.

Part – 2

1) In first encounter, the author mentioned some methods that he used in Unix and Ruby.
2) In Defining collection pipeline, Collections are a collection class in oop languages. an OO collection pipeline would use objects.
3) In exploring more pipelines and operations, we can pass functions into pipeline operations either as lambdas or by the name of a defined function. It's often useful to treat hashmaps as lists of key-value pairs.
4) In alternatives, to use collection pipeline takes shorter time, instead of loops.Because of the exact of a comprehension differ from language to language, we should decide when to use them according to situation.
5) In nested operator expressions, one of useful things we can do with collections is manipulate them with set operations. It's often useful to throw a set operation in the middle of a pipeline.Collections are usually lists which are ordered and allow duplicates.We have to look at the particulars of our library to see what means for set operations.
6) In laziness, for a collection pipeline to be lazy, the collection pipeline functions have to be built with laziness in mind. Java and Ruby have some lazy collection implementations. Platforms that take laziness seriously will usually document operations that are unable to preserve laziness.
7) In parallelism, Many of the pipeline operations naturally work well with parallel invocation. Many platforms include the ability to distribute evaluations in parallel. Parallelizing, however, doesn't always boost performance. Sometimes it takes more time to set up the parallel distribution than it you gain the from the parallelism. As with any performance optimization, we should use performance tests to verify whether using a parallelizing operation actually provides any performance improvement.
8) In immutability, collection-pipelines naturally lend themselves to immutable data structures. Start working with the non-mutating operations and only use something else when we have a known performance bottleneck in the pipeline.