* Why java is not 100% OOPS? 3 reasons
  + Java uses primitives
  + Does not support multiple inheritance
  + Does not support operator overloading
* Java uses pass by value only.
* Collections.sort()
  + Sorts the objects by altering original data
* Stream.sorted()
  + Sorts the copy of the original data, and gives us a new stream output.
* Volatile keyword is used to indicate the java to visit the variable at the main memory bypassing any level of caches.
* Static fields are not serialized during serialization process, only non-static fields are serialized, static field are ignored, because they are part of the class, and not part of the instance.
* If a serializable class contains one non-serializable field, then we’ll encounter NotSerializableException.
* Type erasure is a process referred to the compiler removing the generics during compile time to maintain backward compatibility.
* Java 8 features:
  + Lambda expression
  + Streams API
  + New Date Time API
  + Default and static methods for interfaces
  + Optional class
* Default methods introduced in java 8 for adding more methods to interfaces, without affecting the implemented classes.
* Static methods of interface cannot be overridden by child concrete class.
* Lambda expression can access outside variables, but only final and effective final variables, which does not change its state throughout the program.
* In java, each thread uses its own stack in the stack memory and all shares a single heap memory.
* Visibility problem in java memory model is, changes made by one variable is not immediately or consistently visible for other threads which are executing parallelly, since the threads hold the values in their cache instead of the original memory location.
* NoClassDefFoundError – thrown when JVM failed to locate load a class. (Run time)
* ClassNotFoundException – thrown when the code fails to load a class directly by Class.forName(“MyClass”).
* JIT – Just In Time compilation, used by JVM to convert java byte code to native machine code, which can be run very faster on the processors.
* If an exception is thrown in static initialization block of a class, then the class will be failed to load. And NoClassDefFoundError is thrown if the class is being tried to use.
* HashMap load factor is 0.75. i.e if the 75% of the HM is filled, then the HM will be resized.
* How does resize occur, simple the HM size is doubled.
* How to break a singleton class?
  + Reflection
  + Serialization
  + Cloning
* Java 8 enhancement for HashMap, is they replaced linked list and implemented a balanced tree to hold multiple values in every bucket, to increase the performance.
* Java.util.Optional -> used to avoid null pointer exceptions, gives us a good way of handling the exeception.
* Rules of method overriding
  + Method name should be same
  + Number of parameters, or types or order must be different
  + Return type should be co-variant (same or its child type)
  + Access modifier should not be lesser than the parent method
  + Exception thrown by child should be equal or narrow than parent method