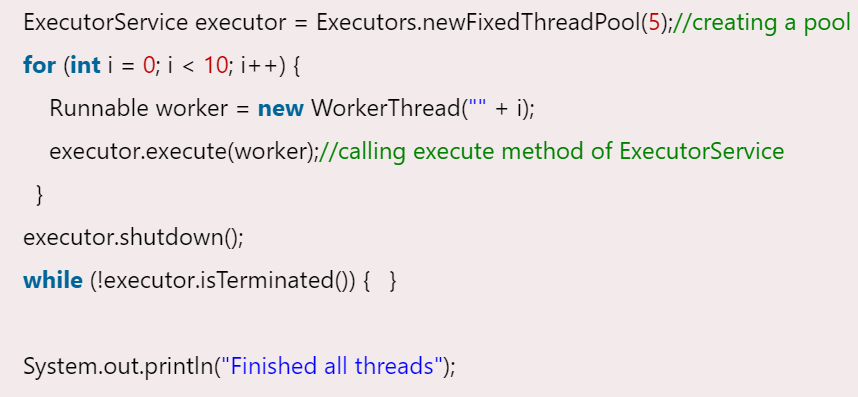
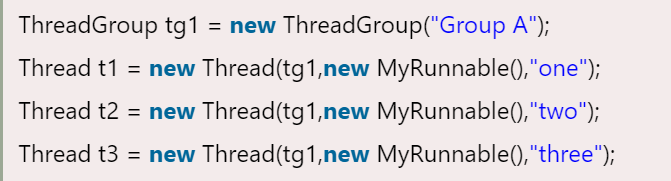
**Java**

1. Constructures
   1. A constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.
   2. Every time an object is created using the new() keyword, at least one constructor is called.
   3. It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.
   4. There are two types of constructors in Java: no-arg constructor, and parameterized constructor.
   5. **Constructor overloading** in Java is a technique of having more than one constructor with different parameter lists.
2. Static keyword
   1. Variable
      1. The static variable gets memory only once in the class area at the time of class loading.
      2. Java static property is shared to all objects.
   2. Method
      1. A static method belongs to the class rather than the object of a class.
      2. A static method can be invoked without the need for creating an instance of a class.
      3. A static method can access static data member and can change the value of it.
      4. The static method cannot use non static data member or call non-static method directly.
      5. this and super cannot be used in static context.
   3. Block
      1. Is used to initialize the static data member.
      2. It is executed before the main method at the time of classloading.
3. This keyword
   1. this can be used to refer current class instance variable.
   2. this can be used to invoke current class method (implicitly)
   3. this() can be used to invoke current class constructor. Call to this() must be the first statement in constructor.
   4. this can be passed as an argument in the method call.
   5. this can be passed as argument in the constructor call.
   6. this can be used to return the current class instance from the method.
4. Inheritance
   1. Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.
   2. Why use inheritance in java?
      1. For Method Overriding (so runtime polymorphism can be achieved).
      2. For Code Reusability.
   3. The extends keyword use for inheritance
   4. Types of inheritance
      1. Single
      2. Multilevel
      3. Hierarchical
      4. Multiple Inheritance is not allowed
5. Super keyword
   1. super can be used to refer immediate parent class instance variable.
   2. super can be used to invoke immediate parent class method.
   3. super() can be used to invoke immediate parent class constructor.
6. Instance initializer block
   1. Instance Initializer block is used to initialize the instance data member. It run each time when object of the class is created.
   2. The instance initializer block is created when instance of the class is created.
   3. The instance initializer block comes in the order in which they appear.
7. Final keyword
   1. Variable
      1. If you make any variable as final, you cannot change the value of final variable(It will be constant).
   2. Method
      1. If you make any method as final, you cannot override it.
   3. Class
      1. If you make any class as final, you cannot extend it.
      2. final method is inherited but you cannot override it.
8. Binding
   1. Static binding
      1. When type of the object is determined at compiled time(by the compiler), it is known as static binding.
   2. Dynamic binding
      1. When type of the object is determined at run-time, it is known as dynamic binding.
9. Abstract Class
   1. A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods
   2. **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
   3. Ways to achieve Abstraction
      1. Abstract class (0 to 100%)
      2. Interface (100%)
   4. It cannot be instantiated.
   5. It can have constructors and static methods also.
   6. It can have final methods which will force the subclass not to change the body of the method.
10. Interface
    1. An interface in Java is a blueprint of a class. It has static constants and abstract methods.
    2. It cannot be instantiated just like the abstract class.
    3. By interface, we can support the functionality of multiple inheritance.
    4. The implements keyword is used for interface
    5. An interface which has no member is known as a **marker** or tagged interface, for example, Serializable, Cloneable, Remote, etc. They are used to provide some essential information to the JVM so that JVM may perform some useful operation.
11. Access Modifier
    1. There are four types of Java access modifiers:
       1. Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
       2. Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
       3. Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
       4. Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.
12. finalize method
    1. finalize is the method in Java which is used to perform clean up processing just before object is garbage collected
13. Error Handling
    1. The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error.
    2. Types of Java Exceptions
       1. Checked Exception : The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions.
       2. Unchecked Exception : The classes that inherit the RuntimeException are known as unchecked exceptions.
       3. Error: Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.
    3. Finally block
       1. Java finally block is a block used to execute important code such as closing the connection, etc.
       2. Java finally block is always executed whether an exception is handled or not.
14. Multithreading
    1. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.
    2. Threads are independent. If there occurs exception in one thread, it doesn't affect other threads. It uses a shared memory area.
    3. Multithreading in Java is a process of executing multiple threads simultaneously.
15. Lifecycle of a Thread
    1. New, Runnable, Running, Non-Runnable(Block) and Terminated
    2. The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.
    3. A thread is in terminated or dead state when its run() method exits.
16. Create Thread
    1. By extending Thread class
    2. By implementing Runnable interface.
17. What if we call run() method directly instead start() method?
    1. Each thread starts in a separate call stack.
    2. Invoking the run() method from main thread, the run() method goes onto the current call stack rather than at the beginning of a new call stack.
18. sleep() method
    1. The sleep() method of Thread class is used to sleep a thread for the specified amount of time.
19. join() method
    1. The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.
20. Priority of a Thread
    1. Default priority of a thread is 5 (NORM\_PRIORITY). The value of MIN\_PRIORITY is 1 and the value of MAX\_PRIORITY is 10.
21. Daemon Thread
    1. Daemon thread in java is a service provider thread that provides services to the user thread. Its life depend on the mercy of user threads i.e. when all the user threads dies, JVM terminates this thread automatically.
22. Thread Pool
    1. a group of fixed size threads are created. A thread from the thread pool is pulled out and assigned a job by the service provider. After completion of the job, thread is contained in the thread pool again.



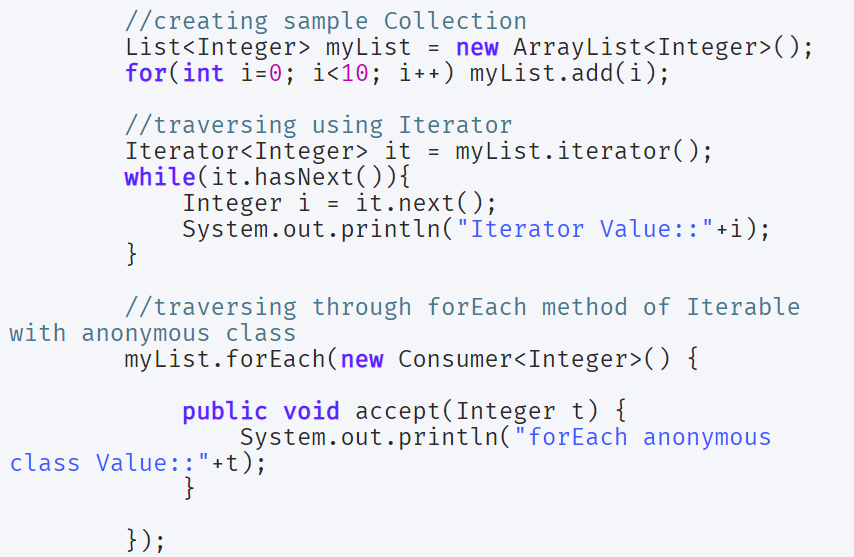
1. ThreadGroup
   1. Java provides a convenient way to group multiple threads in a single object. In such way, we can suspend, resume or interrupt group of threads by a single method call.



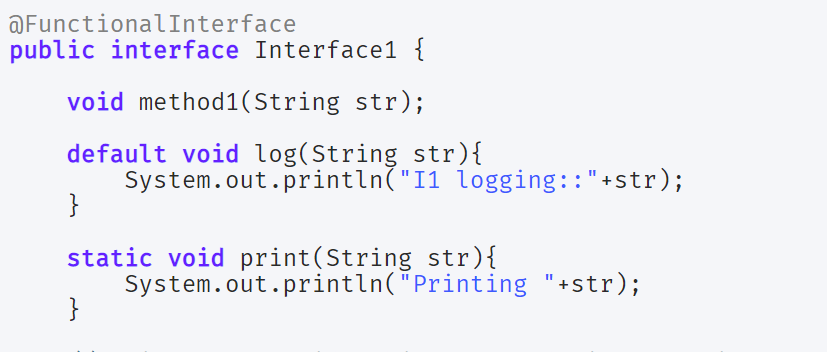
1. Runtime Class
   1. Java Runtime class is used to interact with java runtime environment. Java Runtime class provides methods to execute a process, invoke GC, get total and free memory etc.
2. Synchronization
   1. Synchronization in java is the capability to control the access of multiple threads to any shared resource.
   2. Thread Synchronization
      1. There are two types of thread synchronization mutual exclusive (Synchronized method,Synchronized block and static synchronization) and inter-thread communication.
   3. Concept of Lock in Java
      1. Synchronization is built around an internal entity known as the lock or monitor. Every object has an lock associated with it.
      2. By convention, a thread that needs consistent access to an object's fields has to acquire the object's lock before accessing them, and then release the lock when it's done with them.
   4. synchronized method
      1. Synchronized method is used to lock an object for any shared resource.
   5. Synchronized Block
      1. Synchronized block can be used to perform synchronization on any specific resource of the method.
   6. Static Synchronization
      1. If you make any static method as synchronized, the lock will be on the class not on object.
   7. Deadlock
      1. Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.
3. Inter-thread communication
   1. This is a mechanism in which a thread is paused running in its critical section and another thread is allowed to enter (or lock) in the same critical section to be executed.
   2. .It is implemented by following methods of Object class
      1. wait(): Causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.
      2. notify() : Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened.
      3. notifyAll() : Wakes up all threads that are waiting on this object's monitor.

**Java 8**

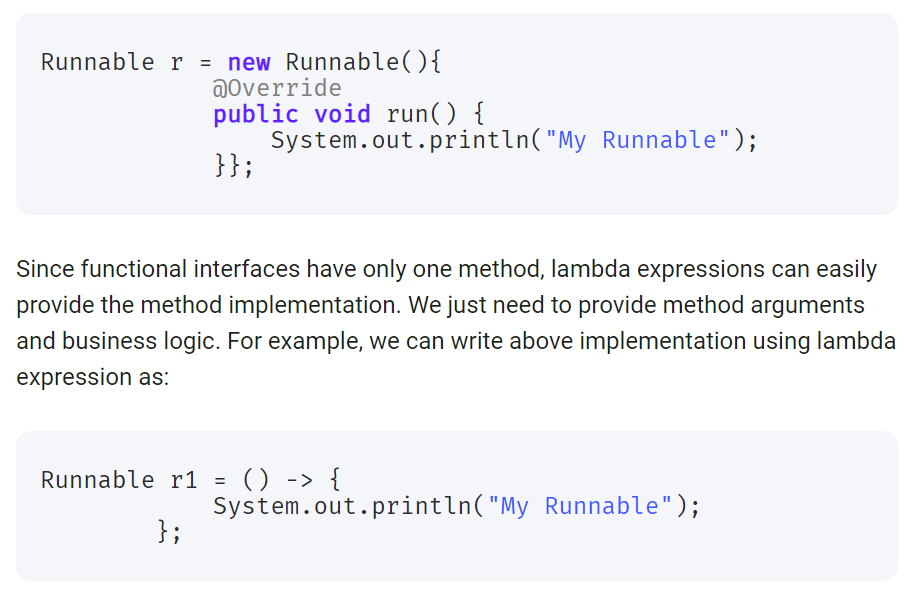
1. forEach() method in Iterable interface
   1. Whenever we need to traverse through a Collection, we need to create an Iterator whose whole purpose is to iterate over, and then we have business logic in a loop for each of the elements in the Collection.
   2. Java 8 has introduced forEach method in java.lang.Iterable interface so that while writing code we focus on business logic. The forEach method takes java.util.function.Consumer object as an argument, so it helps in having our business logic at a separate location that we can reuse.



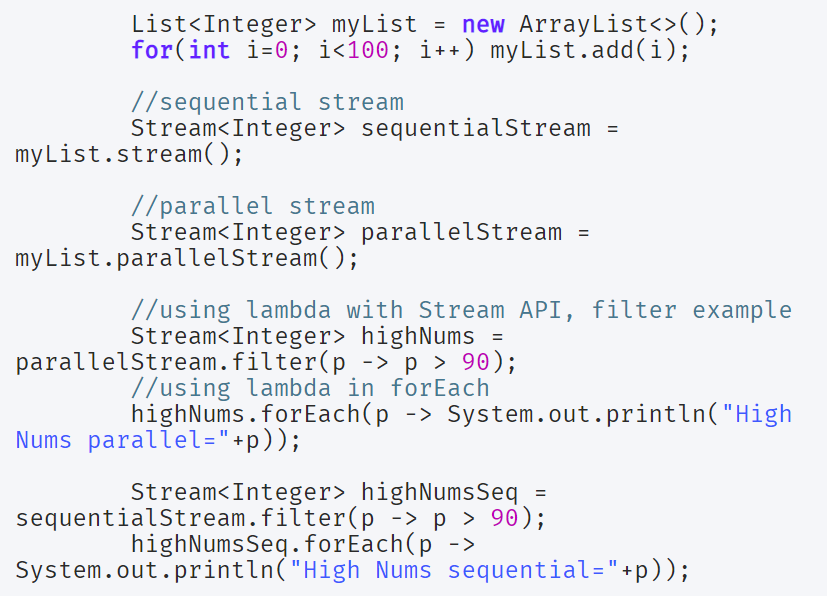
1. default and static methods in Interfaces
   1. Java 8, interfaces are enhanced to have a method with implementation. We can use default and static keyword to create interfaces with method implementation.



1. Functional Interfaces and Lambda Expressions
   1. Functional interfaces are a new concept introduced in Java 8. An interface with exactly one abstract method becomes a Functional Interface. We don’t need to use @FunctionalInterface annotation to mark an interface as a Functional Interface.
   2. java.lang.Runnable with a single abstract method run() is a great example of a functional interface.
   3. One of the major benefits of the functional interface is the possibility to use lambda expressions to instantiate them.
   4. A new package java.util.function has been added with bunch of functional interfaces to provide target types for lambda expressions and method references.
   5. Some of the useful java 8 functional interfaces are Consumer, Supplier, Function and Predicate.



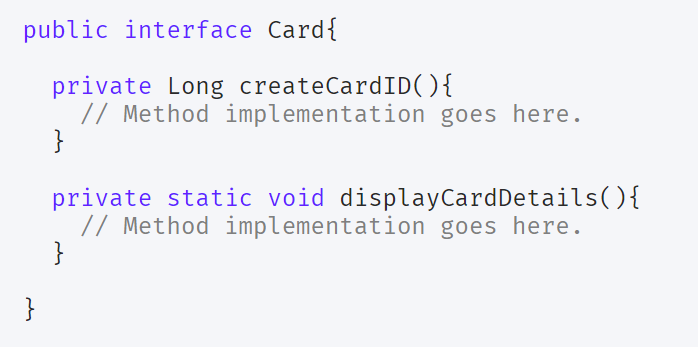
1. Java Stream API for Bulk Data Operations on Collections
   1. A new java.util.stream has been added in Java 8 to perform filter/map/reduce like operations with the collection. Stream API will allow sequential as well as parallel execution.
   2. Collection interface has been extended with stream() and parallelStream() default methods to get the Stream for sequential and parallel execution.



1. Collection API improvements
   1. We have already seen forEach() method and Stream API for collections.
   2. Some new methods added in Collection API are:
      1. Iterator default method forEachRemaining(Consumer action) to perform the given action for each remaining element until all elements have been processed or the action throws an exception.
      2. Collection default method removeIf(Predicate filter) to remove all of the elements of this collection that satisfy the given predicate.
      3. Collection spliterator() method returning Spliterator instance that can be used to traverse elements sequentially or parallel.
      4. Map replaceAll(), compute(), merge() methods.
2. Concurrency API improvements
3. Java IO improvements
4. Java Time API

**Java 9**

1. Private methods in Interfaces
   1. In Java 8, we can provide method implementation in Interfaces using Default and Static methods. However we cannot create private methods in Interfaces.
   2. From Java SE 9 onwards, we can write private and private static methods too in an interface using a ‘private’ keyword.



1. Factory Methods for Immutable List, Set, Map and Map.Entry
   1. introduced some convenient factory methods to create Immutable List, Set, Map and Map.Entry objects.
   2. These utility methods are used to create empty or non-empty Collection objects.

