Lambda expressions are just block of code to be executed later. These are computable functions and the name lambda comes from the Greek letter ^ as used in Principa Mathematica to denote free variables.

Lambdas can’t throw an Exception. Try/Catch it. Or use them for a functional interface that throws an Exception, callable instead of runnable, for e.g.

Functional interfaces are interfaces with a single abstract method. All lambdas are converted to Functional Interfaces. Not all methods on an interface need to be abstract. Methods from Object like equals and toString can be redeclared in an interface as non abstract.

MethodReferences are lambdas but already existing functions. System.out:println where one parameter is passed; Math:pow where two parameters are passed; String::compareToIgnoreCase where two parameters are passed and it’s called on the instance of first parameter and the second parameter is passed as an argument to the method call; this::equals, super::methodName, EnclosingClassInstance::this, EnclosingClassInstance::super when in inner class.

Constructor references – lines.stream(Contribution::new), compiler matches String expiry, String maturity, String strike. Int[]::new where the single argument to the lambda is the length of the array.

Array constructor references are useful to overcome a limitation of Java. It is not possible to construct an array of a generic type T. The expression new T[n] is an error since it would be erased to new Object[n].