Lambda expressions are anonymous functions which are like methods but without the class.

(parameters) -> { lambda expression }

1. Body of Lambda may throw Exceptions
2. Single line lambda
   1. Do not need braces
   2. Do not need an explicit return statement
3. Lambdas with single parameter do not need brackets
4. Lambdas with no parameters must have empty brackets

Java is still strongly, statically typed language.

Type inference means types often need not be explicitly stated.

SAM (Single abstract method) methods can be applied for Lambda expressions.

Lambda expression provides implementation of abstract method.

Static methods can be added to interface.

Consumer<T> - no results returned

Supplier<T> - result is returned.

Function<T,R> - Accepts argument of type T and returns result R.

UnaryOperator<T> - specialized form of Function. Takes inputs of type T and returns result of type T.

BinaryOperator<T> - specialized form of BiFunction. Two arguments and a result of same type T.

Predicate<T> - always returns boolean.

Method References

Format target\_Reference::method\_name

Three kinds of method reference

1. Static method
2. Instance method of an arbitrary type
3. Instance method of an existing object.

|  |  |
| --- | --- |
| 1. Static Method | args -> ClassName.staticMethod(args)  ClassName::staticMethod |
| 1. Instance method of an arbitrary type | (arg0, rest) -> arg0.instanceMethod(rest)  ClassName::instanceMethod |
| 1. Instance method of an existing object | (args) -> expr.instanceMethod(args)  expr::instanceMethod |