Question	Option1	Option2	Option3	Option4	Right Answer
Which cannot be thrown by user manually?	Exception	Throwable	Error	RuntimeException	Option3
Which cannot be thrown by user manually?	Exception	Throwable	Error	RuntimeException	Option3
Which must be handled or declared?	Checked	Runtime	Error	AssertionError	Option1
Which statement is correct?	finally executes only if no exception	finally never executes	finally executes always	finally executes only in catch	Option3
Which returns exception description?	printStackTrace	toString	getMessage	getCause	Option3
Which is superclass of RuntimeException?	Throwable	Error	Exception	IOException	Option3
Which is thrown when accessing null object?	NullPointerException	IllegalState	IOException	AssertionError	Option1
FileNotFoundException is subclass of:	IOException	Exception	Error	Throwable	Option1
try-with-resources requires resource to implement:	Serializable	AutoCloseable	Runnable	Comparable	Option2
StackOverflowError occurs due to:	Overflowing array	Deep recursion	Large file	Invalid cast	Option2
Which exception is thrown by new instance of Integer("abc")?	IOException	NumberFormatException	IllegalArgument	RuntimeException	Option2
catch(Exception e) catches:	Only checked	Only unchecked	All exceptions	Errors	Option3
catch(Exception e) catches:	Only checked	Only unchecked	All exceptions	Errors	Option3
catch block order:	Specific first, general later	General first	Alphabetical	Any order	Option1
Which exception is thrown when thread interrupted during sleep?	SleepException	InterruptedException	IllegalStateException	ThreadException	Option2
Which is thrown when class not found?	ClassNotFoundException	ClassCastException	IllegalArgument	NoSuchClassError	Option1
Which exception is thrown by wait() without notify()?	IllegalMonitorStateException	IllegalState	NullPointer	IOException	Option1
RuntimeException is:	Checked	Unchecked	Error	None	Option2
What is true about Exception propagation?	Checked propagate automatically	Unchecked propagate automatically	Errors propagate only	None	Option2
Which Java 8 feature allows us to filter a collection based on a predicate?	Lambda Expression	Default Method	Optional class	Stream API	Option4
Which Java 8 feature allows us to pass behavior as a parameter to a method?	Lambda expressions	Optional class	Stream API	Functional Interface	Option1

What is the output of the following program?
List<String> names = Arrays.asList("ABC", "CAB", "BCA");
String result = names.stream().reduce("", (a, b) -> a + b.charAt(1));
System.out.println(result);
	"ABC"	"ACB"	"BAC"	"CBA"	Option3

Question	Option1	Option2	Option3	Option4	Right Answer
Which is the new method introduced in the String class in Java 8?	offsetByCodePoints()	matches()	intern()	join()	Option4

What will be the output of the following code snippet?
Optional num= Stream.of(9, 5, 8, 7, 4, 9, 2, 11, 10, 3)
        .filter(n -> n > 10)
        .filter(n -> n % 5 == 0)
        .findFirst();
System.out.println(num);
	Optional[8]	Optional.empty	Optional[11]	11	Option2

Which one is the correct syntax for declaring a lambda expression in Java 8?	(param1, param2) => expression	(param1, param2) :: { expression }	(param1, param2) -> expression	(param1, param2) : expression	Option3
Which of the following interface is not a functional interface?	An interface with a single abstract method	An interface with an abstract method and a default method	An interface with an abstract method and an equals() method	An interface with an abstract method and a run() method	Option4
What is the new Date and Time API in Java 8?	A replacement for the java.sql.Date and java.sql.Calendar classes	A new API for working with Optional	A replacement for the java.util.Date and java.util.Calendar classes	A new API for working with Java Stream API	Option3
Which one among the following is the main feature introduced in Java 8?	Annotations	Generics	Lambda expressions	Enumerations	Option3
Which feature out of following is introduced in Java 8?	Strings in switch statement	Improved type inference: the diamond operator <>	StringJoiner Class	Private methods in Interfaces	Option3

What is the output of the following program?
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
int result= numbers.stream()
        .filter(n -> n % 2 == 0)
        .mapToInt(Integer::intValue)
        .sum();
System.out.println(result);
	10	9	6	15	Option3

Mary is a Java developer. She is working in an assignment. She wants to use a predefined functional interface introduced in Java 8 that doesn't take any input and it always returns some object. Which one among the following option should she use to complete her assignment?	Consumer	Pedicate	Supplier	BiConsumer	Option3

Question	Option1	Option2	Option3	Option4	Right Answer
What is the purpose of default method in Java 8?	A method that is automatically called when an object is created	A method that is automatically called when an object is destroyed	A method that is defined in an interface with a default implementation	A method that is called by the garbage collector to free up memory	Option3
What is the method reference feature in Java 8?	A way to create a new object using a constructor reference	A way to call a static method using a method reference	A way to call an instance method using a method reference	All of the above	Option4
What is the Optional class in Java 8?	A class that represents a value which can either be present or absent	A class that represents an immutable collection	A class that represents optional values to stream of elements	A class as an alternative to functional interface	Option1
What is the new feature introduced in Java 8 in the context of concurrent programming?	ExecutorService	CompletableFuture	ThreadLocal	ReentrantLock	Option2
What is the purpose of the forEach() method in Java 8 Streams API?	To apply an operation to each element of a stream	To filter the elements of a stream	To print the elements of a stream	To map the elements of a stream to a different type	Option1
Robert is working in a Java project. In one of his assignment, he is using Stream API of Java 8 where he wants to transform each element to multiple values and also compress the resulting stream. Which of the following method of Stream API will be the best option for him?	map()	filter()	flatMap()	reduce()	Option3
What is the Stream API in Java 8?	A new file I/O stream library	A new concurrency library	A new API for working with collections	A new API for working with databases	Option3
In the context of Java 8 Stream API, assume that you have a requirement to convert a stream into an array or a collection. Which of the following method of the Stream interface will you apply on the stream to achieve that?	toCollection()	toArray()	toList()	collect()	Option4

What is the output of the following code?
Stream<String> stream = Stream.of("hello", "world", "java");
 stream.filter(s -> s.contains("o"))
        .map(String::toUpperCase)
        .forEach(System.out::print);
	HELLO WORLD JAVA	hello world java	HELLOWORLD	HELLO WORLD	Option3

Question	Option1	Option2	Option3	Option4	Right Answer
When should we use the reduce() method in Java 8 streams?	When we wish to apply a transformation to each element of a stream	When we wish to convert a nested stream into a single stream	When we wish to produce one single result from a sequence of elements	When we wish to filter the elements of a stream	Option3

What is the output of the following code?
Stream<Integer> stream = Stream.of(1, 2, 3, 4, 5);
stream.reduce((a, b) -> a + b)
    .ifPresent(System.out::println);
	15	10	6	5	Option1

Which is the mandatory condition to define a functional interface in Java 8?	The interface should have @functionalInterface annotation on top of it	The interface should have only one method	The interface should have only one abstract method	The interface should at least have one method	Option3
What is the purpose of the parallelStream() method in Java 8?	To filter the elements of a stream	To apply an operation to each element of a stream	To create a parallel stream from a sequential stream	To create a sequential stream from a parallel stream	Option3

What is the output of the following code?
List<Integer> list = Arrays.asList(1, 15, 3, 10, 27);
 int sum = list.stream()
        .filter(i -> i % 3 == 0)
        .mapToInt(Integer::intValue)
        .sum();
 System.out.println(sum);
	18	45	30	56	Option2

What is the purpose of the Optional class in Java 8?	To provide optional values to a stream	To store a reference to a value that may or may not be null	To apply a transformation to each element of a stream	To filter the elements of a stream	Option2

What is the output of the following code?
 List<String> list = Arrays.asList("banana", "apple", "cherry");
  list.stream()
    .sorted((s1, s2) -> s1.compareTo(s2))
    .forEach(System.out::println);
	apple, banana, cherry	cherry, banana, apple	banana, apple, cherry	apple, cherry, banana	Option1

Question	**Option1**	**Option2**	**Option3**	**Option4**	**Right Answer**
What is the purpose of the peek() method in Java 8 streams?	To apply an operation to each element of a stream	To filter the elements of a stream	To convert a stream into an array or a collection	To apply a transformation to each element of a stream	Option1

What is the output of the following code?
IntStream.range(1, 6)
        .mapToObj(i -> i + " ")
        .forEach(System.out::print);
	1 2 3 4 5 6	2 3 4 5	1 2 3 4 5	1 2 3 4	Option3

Which of the following is a valid lambda expression in Java 8?	() -> { System.out.println("Hello"); }	(int x) -> { x++; }	(String s, int x) -> { System.out.println(s + x); }	All of the above	Option1
Which Java 8 feature allows us to iterate through a collection using lambda expressions?	Stream API	Method Reference	Optional	Annotation Processing	Option1
Which of the following is not a valid target type for a lambda expression in Java 8?	An interface with a single abstract method	A functional interface	A class with a single abstract method	All of the above are valid target types	Option3
What is the purpose of the Supplier interface in Java 8?	To represent a method that takes no arguments and returns no value	To represent a method that takes one argument and returns a value	To represent a method that takes no arguments and returns a value	To represent a method that takes one argument and returns no value	Option3

What is the output of the following code snippet?
List<String> words = Arrays.asList("Peris", "London", "New York", "Sydney", "Washington");
String result = words.stream()
                    .filter(w -> w.length() > 10)
                    .findFirst()
                    .orElse("none");
System.out.println(result);
	none	Washington	New York	An error is thrown	Option1