Lambda expressions

1.Sum of Two Integers

ANSWER

```
@FunctionalInterface
interface SumCalculator {
   int sum(int a, int b);
}


public class SumExample {
   public static void main(String[] args) {
      SumCalculator add = (a, b) -> a + b;
      System.out.println("Sum: " + add.sum(10, 20));
   }
}
```

2.Define a functional interface SumCalculator { int sum(int a, int b); } and a lambda expression to sum two integers.

ANSWER

```
import java.util.function.Predicate;


public class StringEmptyCheck {
   public static void main(String[] args) {
      Predicate<String> isEmpty = s -> s.isEmpty();


      System.out.println("Is empty? " + isEmpty.test(""));      // true
      System.out.println("Is empty? " + isEmpty.test("Hello")); // false
   }
}
```

3.Check If a String Is Empty Create a lambda (via a functional interface like Predicate<String>) that returns true if a given string is empty. Predicate<String> isEmpty = s -> s.isEmpty();

ANSWER

```
import java.util.*;

public class FilterEven {
   public static void main(String[] args) {
      List<Integer> numbers = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
      List<Integer> evens = numbers.stream().filter(n -> n % 2 == 0).toList();
      System.out.println("Even numbers: " + evens);
   }
}
```

4.Filter Even or Odd Numbers

ANSWER

```
import java.util.*;

public class FilterOdd {
   public static void main(String[] args) {
      List<Integer> numbers = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
      List<Integer> odds = numbers.stream().filter(n -> n % 2 != 0).toList();
      System.out.println("Odd numbers: " + odds);
   }
}
```

5. Convert Strings to Uppercase/Lowercase

ANSWER

```
import java.util.*;

public class StringCase {
   public static void main(String[] args) {
      List<String> words = Arrays.asList("java", "lambda", "STREAMS");
```

```
        List<String> upper = words.stream().map(String::toUpperCase).toList();
        List<String> lower = words.stream().map(String::toLowerCase).toList();

        System.out.println("Uppercase: " + upper);
        System.out.println("Lowercase: " + lower);
    }
}
```

6. Sort Strings by Length or Alphabetically

ANSWER

```
import java.util.*;

public class SortByLength {
    public static void main(String[] args) {
        List<String> words = Arrays.asList("java", "lambda", "stream");

        List<String> sortedByLength = words.stream()
                .sorted((a, b) -> Integer.compare(a.length(), b.length()))
                .toList();

        System.out.println("Sorted by length: " + sortedByLength);
    }
}
```

7. Aggregate Operations (Sum, Max, Average) on Double Arrays

ANSWER

```
import java.util.*;

public class SortAlphabetically {
    public static void main(String[] args) {
```

```
        List<String> words = Arrays.asList("java", "lambda", "stream");

        List<String> sortedAlpha = words.stream().sorted().toList();

        System.out.println("Sorted alphabetically: " + sortedAlpha);
    }
}
```

8.Create similar lambdas for max/min. 9.Calculate Factorial

ANSWER

```
import java.util.*;

public class AggregateExample {
    public static void main(String[] args) {
        double[] arr = {10.5, 22.3, 3.2, 44.7};

        DoubleSummaryStatistics stats = Arrays.stream(arr).summaryStatistics();

        System.out.println("Sum: " + Arrays.stream(arr).sum());
        System.out.println("Max: " + stats.getMax());
        System.out.println("Min: " + stats.getMin());
        System.out.println("Average: " + stats.getAverage());
    }
}
```

9) Factorial Using Lambda

ANSWER

```
import java.util.function.Function;
import java.util.stream.IntStream;
```

```
public class FactorialExample {
   public static void main(String[] args) {
      Function<Integer, Integer> factorial = n ->
           IntStream.rangeClosed(1, n).reduce(1, (a, b) -> a * b);


      System.out.println("Factorial of 5 = " + factorial.apply(5));
   }
}
```