**Advanced Java Assessment**

1. Write a program to run the method parallelly and combine results using Concurrency API.

**package** com.maveric;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.concurrent.\*;

**public** **class** ConcurrencyAPIDemo {

**public** **static** **void** main(String[] args) {

**int** no = 4;

ExecutorService executor = Executors.*newFixedThreadPool*(no);

List<Callable<Integer>> tasks = **new** ArrayList<>();

tasks.add(() -> *method1*(2,7));

tasks.add(() -> *method2*(13,8));

tasks.add(() -> *method3*(21,19));

**try** {

List<Future<Integer>> list = executor.invokeAll(tasks);

**int** totalResult = 0;

**for** (Future<Integer> result : list) {

totalResult = totalResult + result.get();

}

System.***out***.println("Combined result: " + totalResult);

} **catch** (InterruptedException | ExecutionException e) {

e.printStackTrace();

} **finally** {

executor.shutdown();

}

}

**private** **static** **int** method1(**int** a, **int** b) {

**return** a\*b;

}

**private** **static** **int** method2(**int** a, **int** b) {

**return** a+b;

}

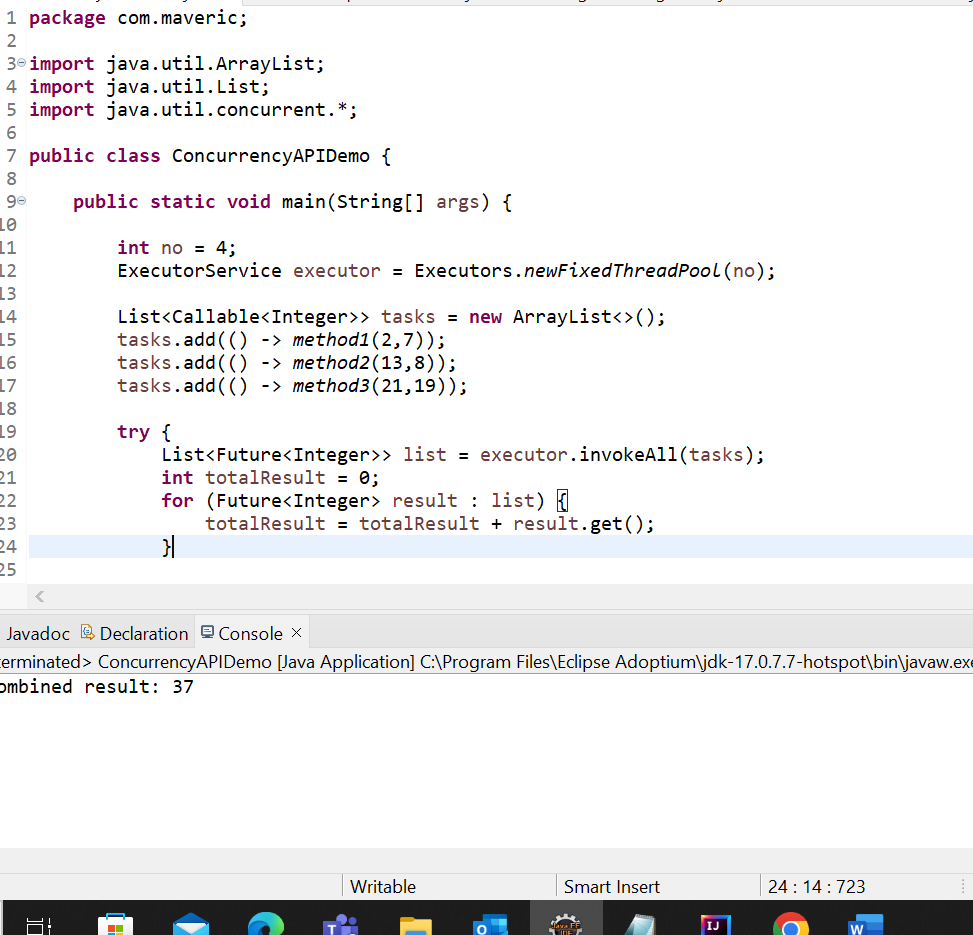
**private** **static** **int** method3(**int** a, **int** b) {

**return** a-b;

}

}

Output:



1. Write a program to filter out all even numbers from a list using lambda expressions.

**package** com.maveric;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** LambdaExpressionDemo {

**public** **static** **void** main(String[] args) {

List<Integer> numbers = Arrays.*asList*(43, 21, 36, 8, 98, 60, 23, 1, 60, 17);

List<Integer> evenNumbers = **new** ArrayList<>();

numbers.forEach(number -> {

**if** (number % 2 == 0) {

evenNumbers.add(number);

}

});

System.***out***.println("The given list is "+ numbers);

System.***out***.println("Even numbers from the list are " + evenNumbers);

}

}

**Output:**

A screenshot of a computer program

Description automatically generated with medium confidence

1. Write a program to convert a list of integers to a list of strings using lambda expressions and the map operation.

**package** com.maveric;

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.stream.Collectors;

**public** **class** IntegertoStringDemo {

**public** **static** **void** main(String[] args) {

List<Integer> integerList = Arrays.*asList*(10, 20, 30, 40, 50);

List<String> stringList = integerList.stream()

.map(number -> number.toString())

.collect(Collectors.*toList*());

System.***out***.println("String List: " + stringList);

}

}

**Output:**

A screenshot of a computer program

Description automatically generated with low confidence

1. Write a program to run the method parallelly and return the value using Executors Future.

**package** com.maveric;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.concurrent.\*;

**public** **class** ExecutorFutureDemo {

**public** **static** **void** main(String[] args) {

**int** no = 4;

ExecutorService executor = Executors.*newFixedThreadPool*(no);

List<Callable<Integer>> tasks = **new** ArrayList<>();

tasks.add(() -> *method1*());

tasks.add(() -> *method2*());

tasks.add(() -> *method3*());

**try** {

List<Future<Integer>> list = executor.invokeAll(tasks);

**int** totalResult = 0;

**for** (Future<Integer> result : list) {

totalResult = totalResult + result.get();

}

System.***out***.println("Combined result: " + totalResult);

} **catch** (InterruptedException | ExecutionException e) {

e.printStackTrace();

} **finally** {

executor.shutdown();

}

}

**private** **static** **int** method1() {

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**return** 10;

}

**private** **static** **int** method2() {

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**return** 50;

}

**private** **static** **int** method3() {

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**return** 100;

}

}

Output:

A screenshot of a computer program

Description automatically generated with medium confidence