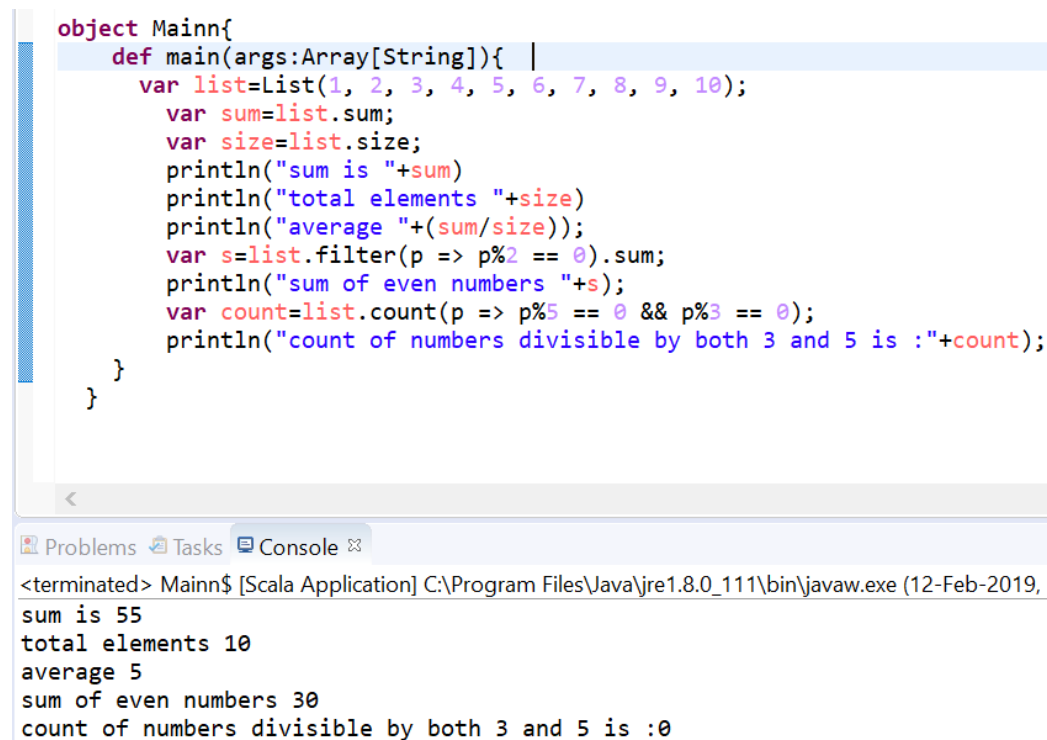


Task 1:

Perform operations on given List of Numbers

```
object Mainn{
  def main(args:Array[String]){
    var list=List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    var sum=list.sum;
    var size=list.size;
    println("sum is "+sum)
    println("total elements "+size)
    println("average "+(sum/size));
    var s=list.filter(p => p%2 == 0).sum;
    println("sum of even numbers "+s);
    var count=list.count(p => p%5 == 0 && p%3 == 0);
    println("count of numbers divisible by both 3 and 5 is :"+count);
  }
}
```



The screenshot shows an IDE with a Scala file. The code defines a `Mainn` object with a `main` method that processes a list of numbers from 1 to 10. The console output shows the results of these operations.

```
object Mainn{
  def main(args:Array[String]){
    var list=List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    var sum=list.sum;
    var size=list.size;
    println("sum is "+sum)
    println("total elements "+size)
    println("average "+(sum/size));
    var s=list.filter(p => p%2 == 0).sum;
    println("sum of even numbers "+s);
    var count=list.count(p => p%5 == 0 && p%3 == 0);
    println("count of numbers divisible by both 3 and 5 is :"+count);
  }
}
```

Console Output:

```
<terminated> Mainn$ [Scala Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (12-Feb-2019,
sum is 55
total elements 10
average 5
sum of even numbers 30
count of numbers divisible by both 3 and 5 is :0
```

Task 2:

a) Limitations of mapreduce

- Issue with Small Files : HDFS lacks the ability to efficiently support the random reading of small files because of its high capacity design
- Slow processing : Data is distributed and processed over the cluster in MapReduce which increases the time and reduces processing speed.

- Support for Batch Processing only : does not process streamed data
- No Delta Iteration : Hadoop does not support cyclic data flow
- Not Easy to use : developers need to hand code for each and every operation which makes it very difficult to work
- Security : Hadoop is missing encryption
- No Abstraction : Hadoop does not have any type of abstraction
- No caching : MapReduce cannot cache the intermediate data in memory for a further requirement which diminishes the performance of Hadoop
- Uncertainty : Hadoop only ensures that data job is complete, but it's unable to guarantee when the job will be complete

b)RDD features

RDD (Resilient Distributed Dataset) is the fundamental data structure of Apache Spark which are an immutable collection of objects which computes on the different node of the cluster.

Features:

- In memory computation
- Lazy evaluation
- Fault tolerance
- Immutability
- Persistence
- Partitioning
- Parallel
- Location stickiness
- Typed

c) RDD operations

Transformation : Transformations are kind of operations which will transform your RDD data from one form to another. And when you apply this operation on any RDD, you will get a new RDD with transformed data (RDDs in Spark are immutable). Operations like map, filter, flatMap are transformations.

Action : Transformations create RDDs from each other, but when we want to work with the actual dataset, at that point action is performed. When the action is triggered after the result, new RDD is not formed like transformation. Thus, actions are RDD operations that give non-RDD values. The values of action are stored to drivers or to the external storage system.