Introduction to Big Data

ECAP456

Dr. Rajni Bhalla

Associate Professor

Learning Outcomes

After this lecture, you will be able to

- learn what is unit testing
- explore concepts of MRUnit

Introduction

What is Unit Testing?

Unit Testing - Advantages:

- Reduces Defects in the Newly developed
- Reduces Cost of Testing
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build.

Unit Testing Techniques:

Black Box Testing

Unit Testing Techniques:

White Box Testing

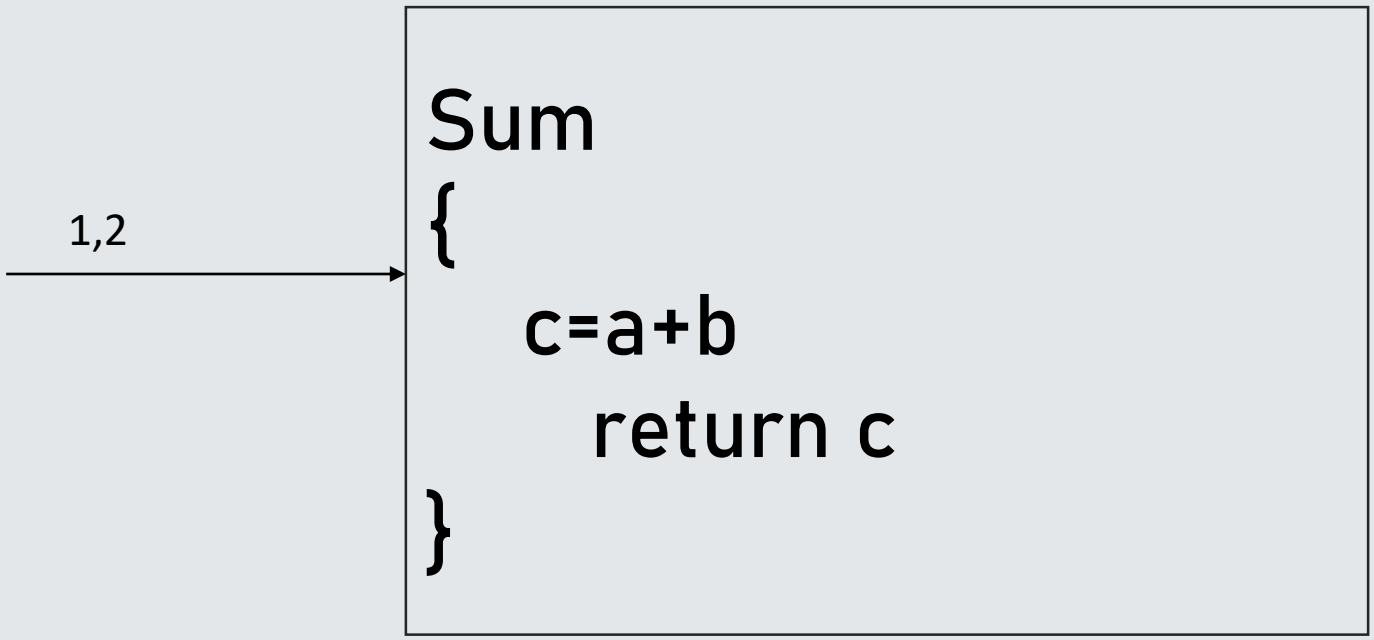
Unit Testing Techniques:

- Black Box Testing
- White Box Testing
- Gray Box Testing

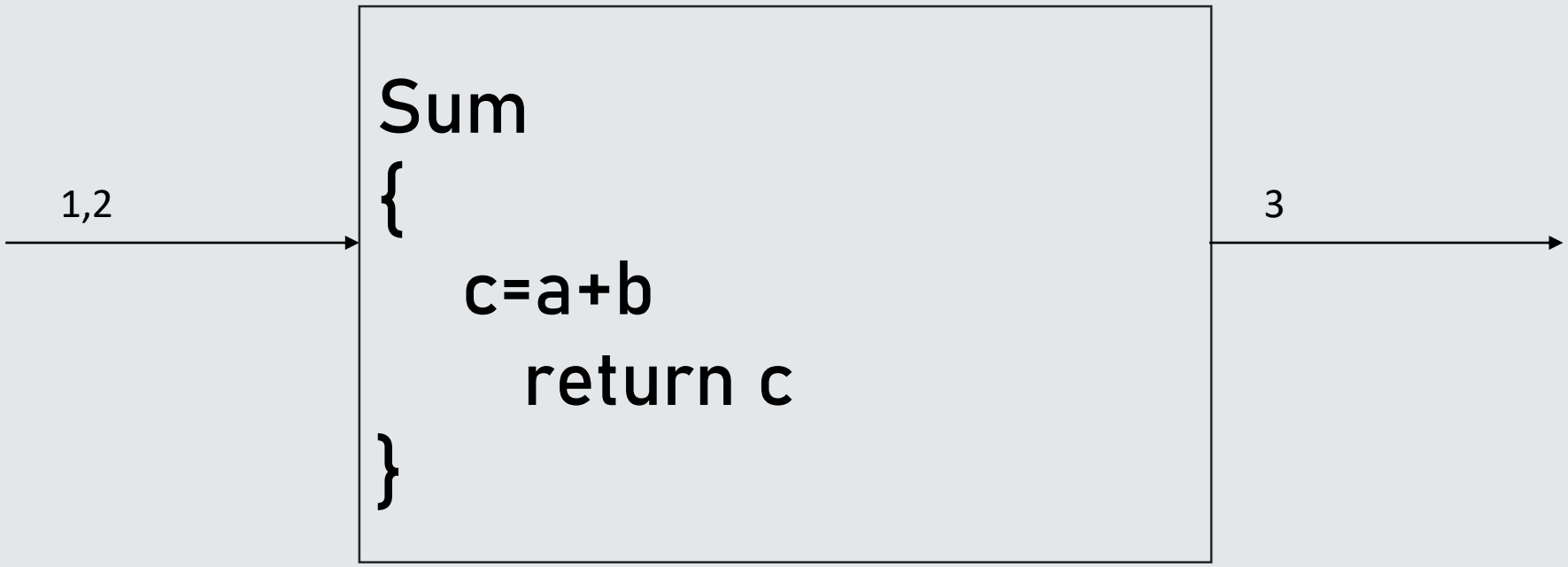
Introduction

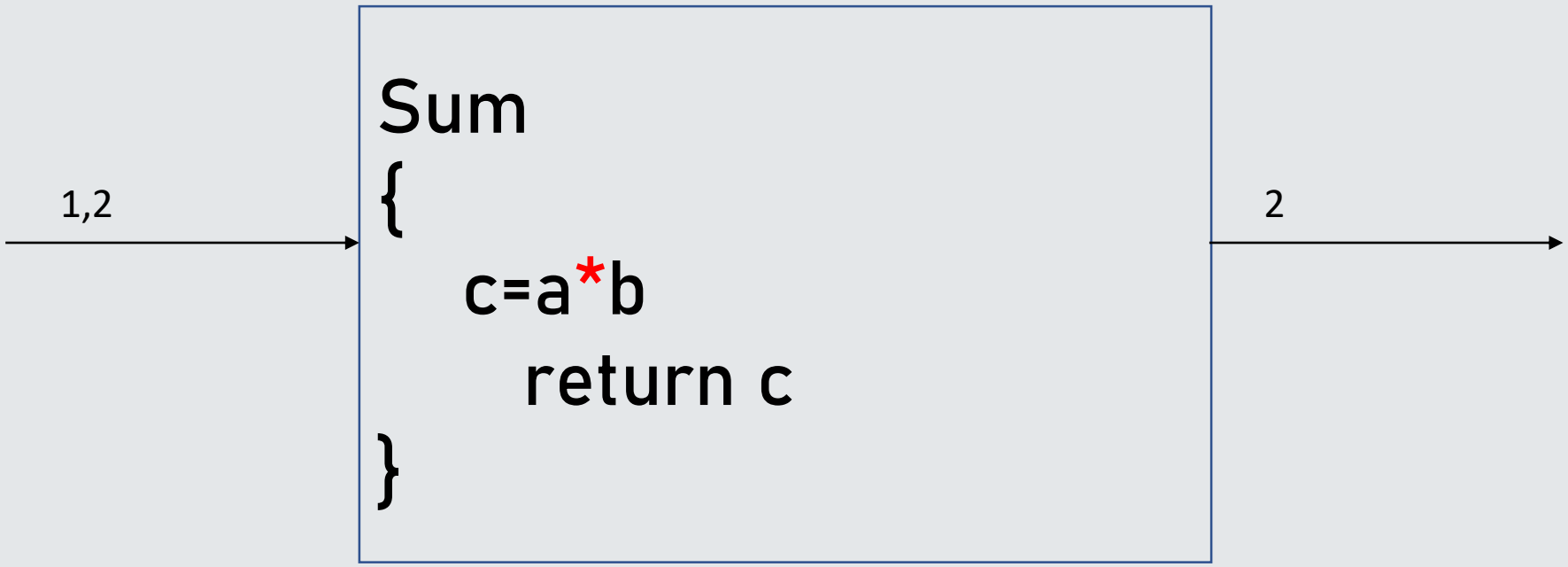
```
Sum
{
    c=a+b
      return c
}
```

Introduction



Introduction




1,2
Sum
{
c=a*b
return c
}
2

Introduction

What is MRUnit ?

What is MRUnit?

- Testing library for MapReduce
- Developed by Cloudera
- Easy integration between MapReduce and standard testing tools (e.g. JUnit)

MRUnit

- Apache is an Open Source library.
- It provides a convenient integration between MapReduce and standard testing libraries
- It doesn't replace Junit
- Knowledge of Hadoop MapReduce and JUnit is required for a better understanding.

MRUnit

- Apache is an Open Source library
- It provides a convenient integration between MapReduce and standard testing libraries.
- It doesn't replace Junit
- Knowledge of Hadoop MapReduce and JUnit is required for a better understanding.

MRUnit

- Apache is an Open Source library
- It provides a convenient integration between MapReduce and standard testing libraries
- It doesn't replace Junit.
- Knowledge of Hadoop MapReduce and JUnit is required for a better understanding.

MRUnit

- Apache is an Open Source library.
- It provides a convenient integration between MapReduce and standard testing libraries.
- It doesn't replace Junit.
- Knowledge of Hadoop MapReduce and JUnit is required for a better understanding.

MRUnit

- The three core classes of MRUnit are the following:

Map Driver

Reducer Driver

MapReduce Driver

MRUnit

MRUnit drivers needed for each specific test purposes.

MRUnit

- In order to add MRUnit to an Hadoop MapReduce project you need to add it as test dependency in the project POM file.

MRUnit

- and of course JUnit should be present as well:

MRUnit

- The MapReduce application under test is the popular word counter (the Hello World of MapReduce). It processes text files and counts how often words occur. Browsing the web you can find hundred of links with this example, but just in case here's the code for the Mapper.

MRUnit

MRUnit

- And the Reducer

MRUnit

- Now let's start implementing a MRUnit test case. It has the same structure as for a JUnit test case. First of all declare the instance variable for the MapReduce test purposes. They include the drivers provided by the MRUnit framework:

MRUnit

MRUnit

- Then create instances of them in the JUnit setUp() method:

MRUnit

- To test the Mapper you need just to provide the inputs and the expected output to the MapperDriver instance and then execute its runTest() method:

MRUnit

- As you can see from the code above, MRUnit supports multiple inputs. firstTestKey and secondTestKey are String variables you can initialize in the setUp() method as well. It is the same process to test the Reducer.

MRUnit

- @Test

```
 public void testWordCountReducer() throws IOException {
        Text firstMapKey = new Text(firstTestKey);
        List<IntWritable> firstMapValues = new ArrayList<IntWritable>();
        firstMapValues.add(new IntWritable(1));
        firstMapValues.add(new IntWritable(1));

        Text secondMapKey = new Text(secondTestKey);
        List<IntWritable> secondMapValues = new
   ArrayList<IntWritable>();
        secondMapValues.add(new IntWritable(1));
```

MRUnit

- ArrayList<IntWritable>();

```
        secondMapValues.add(new IntWritable(1));
        secondMapValues.add(new IntWritable(1));
        secondMapValues.add(new IntWritable(1));

        reduceDriver.withInput(firstMapKey, firstMapValues)
        .withInput(secondMapKey, secondMapValues)
        .withOutput(firstMapKey, new IntWritable(2))
        .withOutput(secondMapKey, new IntWritable(3))
        .runTest();
    }
```

MRUnit

- and the overall MapReduce flow

```
@Test
 public void testWordCountMapReducer() throws IOException {
        mapReduceDriver.withInput(new LongWritable(1), new
Text(firstTestKey))
        .withInput(new LongWritable(2), new Text(firstTestKey))
        .withInput(new LongWritable(3), new Text(secondTestKey))
        .withOutput(new Text(firstTestKey), new IntWritable(2))
        .withOutput(new Text(secondTestKey), new IntWritable(1))
        .runTest();
 }
```

MRUnit

- Just the specific driver to use changes. Any MRUnit test case can be executed the same way as for the JUnit test cases. So you can run all of them together for your applications. When you execute the command

MRUnit

- Maven will run all of the unit tests (JUnit and MRUnit both) available for the given MapReduce application and generate their execution reports.

In Summary, MRUnit...

- Makes testing your Hadoop jobs easier.
- Abstracts away a lot of the boilerplate test setup you need.
- Has it's problems.
- But they are outweighed by the benefits.


That's all for now...