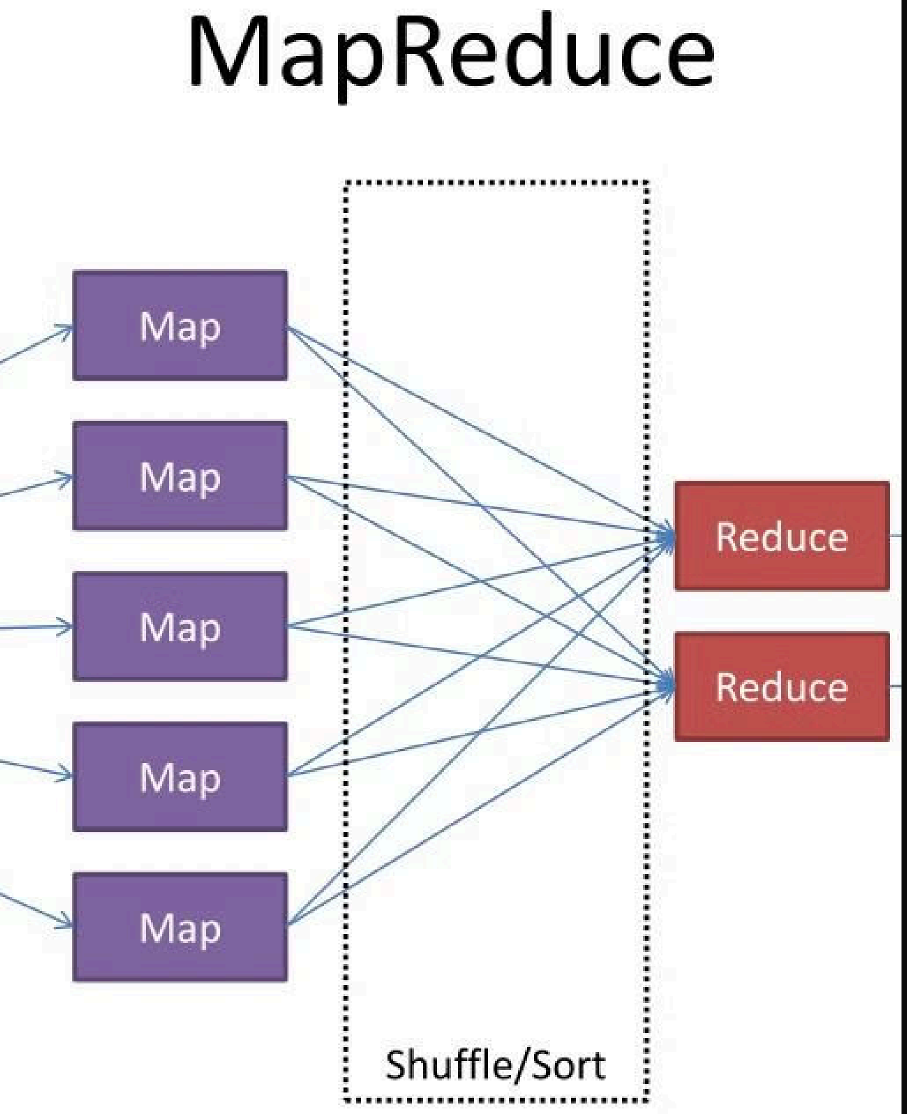
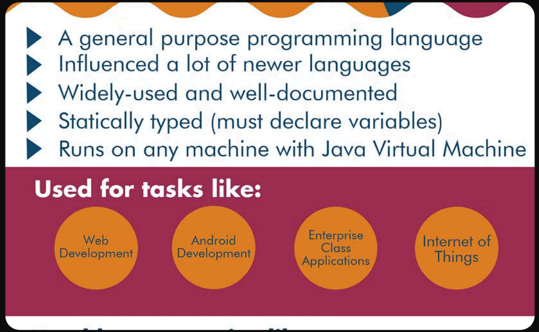


Introduction to MapReduce

An introduction to MapReduce, a programming model for processing and generating big data sets with a parallel, distributed algorithm on a cluster.

RA by Rohith Daggubati

Basics of Java programming language



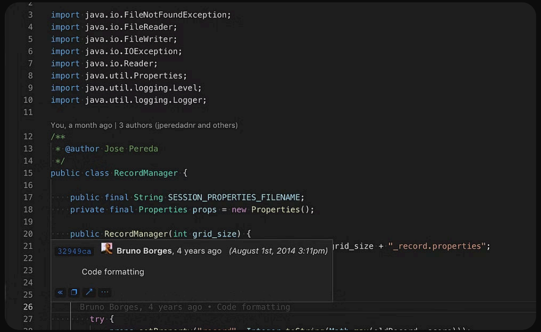
- A general purpose programming language
- Influenced a lot of newer languages
- Widely-used and well-documented
- Statically typed (must declare variables)
- Runs on any machine with Java Virtual Machine

Used for tasks like:

Web Development
Android Development
Enterprise Class Applications
Internet of Things

Variables and Data Types

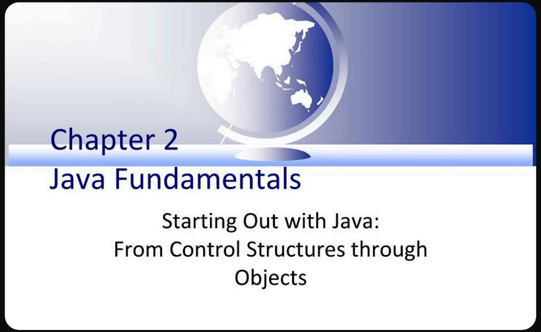
Learn about different types of data like int, float, and string in Java.



```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;

You, a month ago | 3 authors (jperedadnr and others)
/**
 * @author Jose Pereda
 */
public class RecordManager {

    public final String SESSION_PROPERTIES_FILENAME;
    private final Properties props = new Properties();

    public RecordManager(int grid_size) {
32949ca  Bruno Borges, 4 years ago  (August 1st, 2014 3:11pm)      rid_size + "_record.properties";

    Code formatting

        Bruno Borges, 4 years ago • Code formatting
        try {
```

Control Statements

Understand concepts like if, else, switch, and loops in Java.



Chapter 2
Java Fundamentals

Starting Out with Java:
From Control Structures through Objects

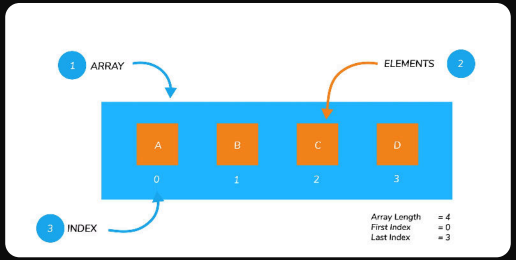
Object-Oriented Programming

Understand classes, objects, inheritance, polymorphism, and encapsulation.

Understanding data structures in Java

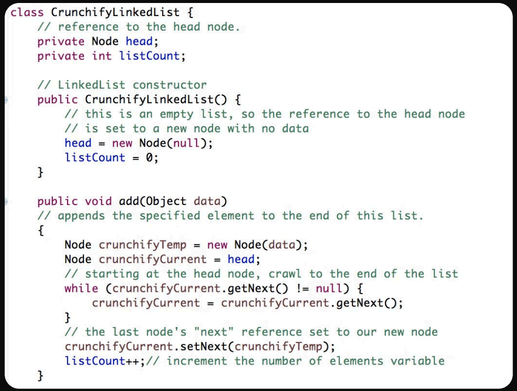
Arrays

Learn about arrays in Java and how to manipulate and iterate through them.



Linked Lists

Understand the concept of linked lists and their implementation in Java.



```
class CrunchifyLinkedList {
    // reference to the head node.
    private Node head;
    private int listCount;

    // LinkedList constructor
    public CrunchifyLinkedList() {
        // this is an empty list, so the reference to the head node
        // is set to a new node with no data
        head = new Node(null);
        listCount = 0;
    }

    public void add(Object data)
    // appends the specified element to the end of this list.
    {
        Node crunchifyTemp = new Node(data);
        Node crunchifyCurrent = head;
        // starting at the head node, crawl to the end of the list
        while (crunchifyCurrent.getNext() != null) {
            crunchifyCurrent = crunchifyCurrent.getNext();
        }
        // the last node's "next" reference set to our new node
        crunchifyCurrent.setNext(crunchifyTemp);
        listCount++;// increment the number of elements variable
    }
```

Stacks and Queues

Understand the working and implementation of stacks and queues in Java.

Working with collections and arrays in Java

1. **ArrayList**

   A dynamic array that can grow or shrink in size.

2. **HashMap**

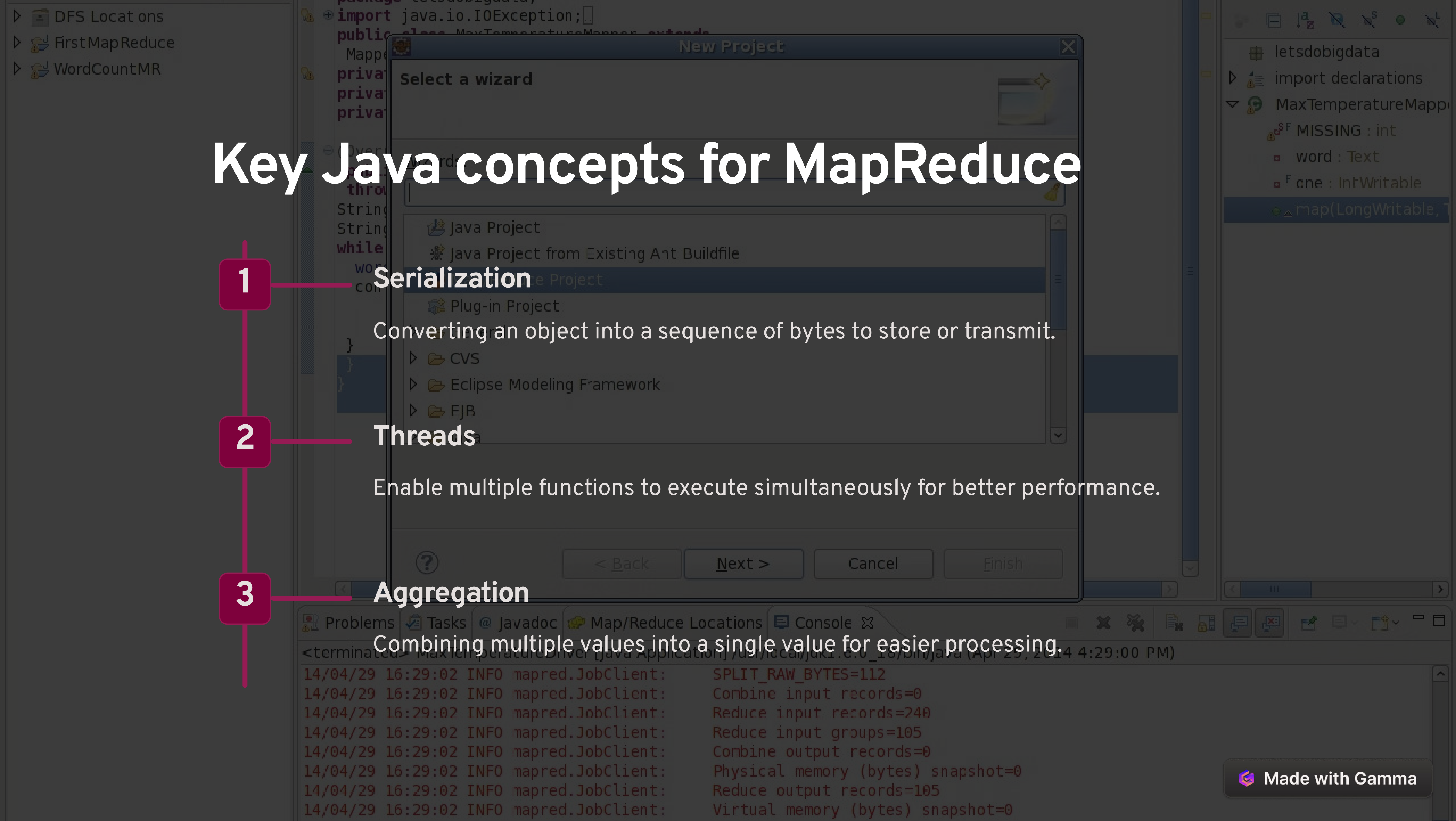
   Key-value pair mapping for fast retrieval of data.

3. **HashSet**

   Collection of unique elements with no duplicate values.

4. **LinkedList**

   A sequence of elements with pointers connecting each element.



```
14/04/29 16:29:02 INFO mapred.JobClient:     SPLIT_RAW_BYTES=112
14/04/29 16:29:02 INFO mapred.JobClient:     Combine input records=0
14/04/29 16:29:02 INFO mapred.JobClient:     Reduce input records=240
14/04/29 16:29:02 INFO mapred.JobClient:     Reduce input groups=105
14/04/29 16:29:02 INFO mapred.JobClient:     Combine output records=0
14/04/29 16:29:02 INFO mapred.JobClient:     Physical memory (bytes) snapshot=0
14/04/29 16:29:02 INFO mapred.JobClient:     Reduce output records=105
14/04/29 16:29:02 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=0
```

Key Java concepts for MapReduce

1. **Serialization**

   Converting an object into a sequence of bytes to store or transmit.

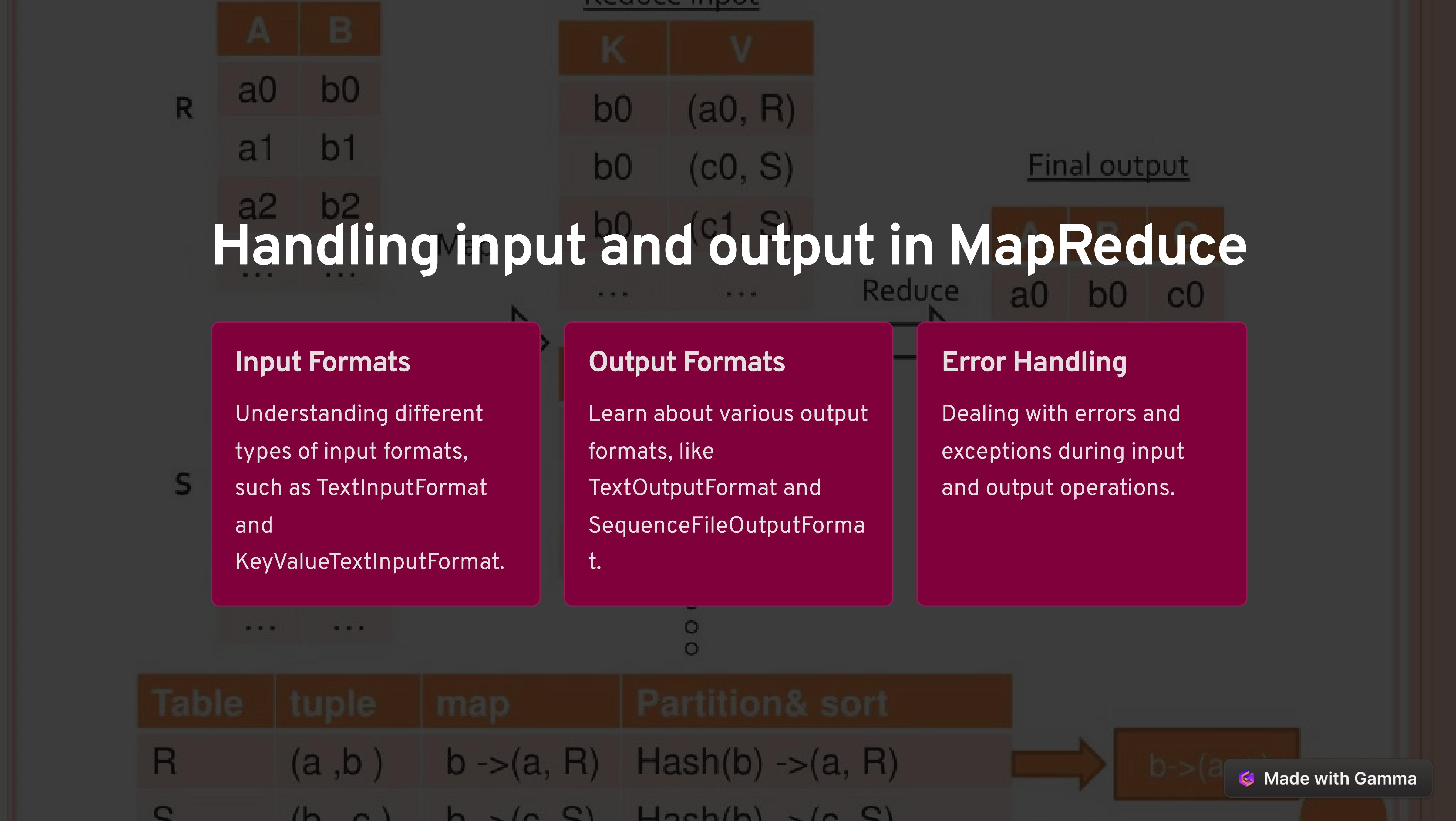
2. **Threads**

   Enable multiple functions to execute simultaneously for better performance.

3. **Aggregation**

   Combining multiple values into a single value for easier processing.

Made with Gamma



Handling input and output in MapReduce

Input Formats

Understanding different types of input formats, such as TextInputFormat and KeyValueTextInputFormat.

Output Formats

Learn about various output formats, like TextOutputFormat and SequenceFileOutputFormat.

Error Handling

Dealing with errors and exceptions during input and output operations.

Implementing MapReduce algorithms in Java

1

Data Preprocessing

Prepare the input data for the MapReduce algorithm.

2

Map Phase

Process and transform the input data into intermediate key-value pairs.

3

Reduce Phase

Aggregate and consolidate the intermediate data into the final output.

4

Optimization

Enhance algorithm efficiency for faster data processing.

Best practices and tips for MapReduce in Java

1

Modularity

Divide the code into smaller, manageable modules for easier debugging.

2

Code Optimization

Ensure efficient resource utilization by optimizing the MapReduce code.

3

Error Handling

Implement robust error handling to prevent data processing failures.