任务1

用于处理股票数据，计算每个股票名称出现的次数，并对结果进行排序。以下是编码思路的简要说明：

1. 数据读取：

- `StockCountMapper`类使用`CSVParser`解析CSV格式的股票数据，提取股票名称，并为每个名称生成一个计数（默认为1）。

2. 计数聚合：

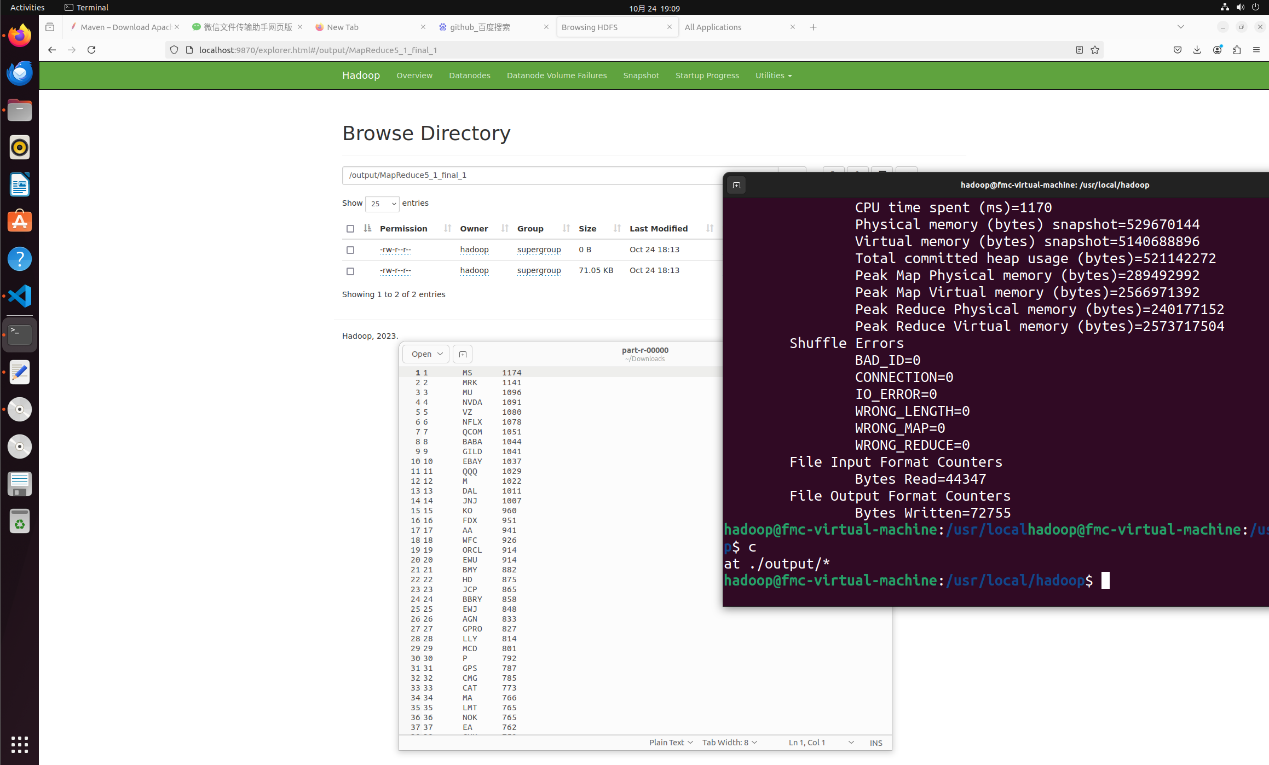
- `StockCountReducer`类接收来自Mapper的键值对，对每个股票名称的计数进行聚合，计算总出现次数。

3. 排序准备：

- `SortCountMapper`类读取股票名称及其计数，将计数作为键，股票名称作为值，为后续排序做准备。

4. 结果排序：

- `SortCountReducer`类接收来自SortCountMapper的排序键值对，生成一个排序后的序列编号，并输出最终的排序结果。



1. **数据读取**：

WordCountMapper类负责读取输入的数据。它使用CSVParser来解析CSV格式的文本数据。对于每一行数据，它提取出股票名称（作为单词处理），并为每个股票名称生成一个默认为1的计数。这个类实际上执行的是单词计数的任务，但在您的模板中，它被描述为StockCountMapper，这可能是一个名称上的混淆。

1. **计数聚合**：

WordCountReducer类接收来自WordCountMapper的输出，对每个股票名称（或单词）的计数进行聚合。它计算每个股票名称的总出现次数，并将这些信息输出。在您的模板中，这个类被描述为StockCountReducer，这同样可能是一个名称上的混淆。

1. **排序准备**：

SortCountMapper类读取WordCountReducer的输出，即每个股票名称及其对应的计数。它将计数作为键（LongWritable），股票名称作为值（Text），为后续的排序操作做准备。这个步骤是为了确保在排序阶段，可以根据计数来进行排序。

1. **结果排序**：

SortCountReducer类接收来自SortCountMapper的排序键值对，并生成一个排序后的序列编号。它输出每个股票名称及其对应的计数，并且每个输出都有一个唯一的序列号。这个类确保了最终的输出是按照股票名称的计数降序排列的。

1. **作业配置与执行**：

WordCountDriver类是MapReduce作业的驱动程序，负责配置和启动上述两个MapReduce作业。第一个作业负责计数，第二个作业负责排序。这个类设置了作业的配置参数，如输入输出路径、Mapper和Reducer类等，并等待作业完成。

