Assignment 3: Task 1 and 2

Task1

- **Split lines into words**
- **Map each word to a key-value pair , e.g., <word, 1>**
- **Reduce by key to count frequency for each word, e.g., <word, 12345>**
- **Swap the key-value pairs, e.g., <word, 12345> -> <12345, word >**
- **Sort by key**
- **Swap back the key-value pairs, e.g., <12345, word > -> <word, 12345>**

Task1

```
JavaRDD<String> words = lines.flatMap (new FlatMapFunction<String, String>(){})
// Split lines into words
JavaPairRDD<String, Integer> ones = words.mapToPair(new PairFunction<String, String,
Integer>(){})
// Map each word to key-value pair , e.g., <word, 1>
JavaPairRDD<String, Integer> counts = ones.reduceByKey(new Function2<Integer, Integer,
Integer>(){})
// Reduce by key to count frequency for each word, e.g., <word, 12345>
JavaPairRDD<Integer, String> swappedPair = counts
                        .mapToPair(new PairFunction<Tuple2<String, Integer>, Integer, String>(){})
// Swap the key-value pairs, e.g., <word, 12345> -> <12345, word >
JavaPairRDD<Integer, String> swappedPairSorted = swappedPair.sortByKey(false);
// Sort by key
JavaPairRDD<String, Integer> swappedPairSortedSwapBack = swappedPairSorted
                        .mapToPair(new PairFunction<Tuple2<Integer, String>, String, Integer>(){})
// Swap back the key-value pairs, e.g.,  <12345, word > -> <word, 12345>
```

Task 2

Counting total number of commands:

```
long num_command = lines.filter(new Function<String, Boolean>() {
    @Override
    public Boolean call(String s) throws Exception {
        return s.contains("<Command ");
    }
}).count();
```

Task 2

Counting total number of commands:

```
long num_command = lines.filter(new Function<String, Boolean>() {
    @Override
    public Boolean call(String s) throws Exception {
        return s.contains("<Command ");
    }
}).count();
```

Task 2

Find the largest timestamp

```
JavaPairRDD<Long, Long> timestamps = words.mapToPair(new PairFunction<String,
Long, Long>() {
        @Override
        public Tuple2<Long, Long> call(String s) {
        Pattern p = Pattern.compile("timestamp=\"(\\d*?)\"");
                                Matcher m = p.matcher(s);
        if (matcher.find()) {
                return new Tuple2<Long, Long>(1, Long. parseLong(matcher.group(1)));
        } else {
                return new Tuple2<Long, Long>(1, 0);
        }
    }
});
```

Task 2

Find the largest timestamp

```
JavaPairRDD<Long, Long> counts = timestamps.reduceByKey(new Function2<Long, Long,
Long>() {
	@Override
	public Long call(Long i1, Long i2) {
		if (i1 > i2) {
			return i1;
		} else {
			return i2;
		}
	}
});
```

Task 3 and 4

- Task 3
  - Counting the number of commands in every 15 min interval
- Task 4
  - Counting and ordering all distinct commands.