# Balance Detector

From the given requirement of the project, I have to create a program that can detect the error in balance calculation of givern transactions record.

We create BalanceDetectorMapper to detect the error in balance calculation in each transaction records that Mapper receiver.

We have Class variable `prev` that stores the previous transaction balance.

In `map` method, the explanation will shown below. 1) We check the header of csv file to ignore it

```
if (value.toString().equals("Date,Description,Deposits,Withdrawls,Balance")) return;
```

2) We extract csv data and store it in a list of String.

```
String[] data = value.toString().split(",(?=([^\"]*\"[^\"]*\")*[^\"]*$)");
```

3) We loop through list of string to replace the comma and double quote with empty string.

```
for (int i = 0; i < data.length; i++) {
      data[i] = data[i].replaceAll(",", "").replaceAll("\"", "");
}
```

4) We check that this record is the first record of this Mapper or not by using condition that the first record will has no previous balance. Then we assign the balance of this record to `prev` variable.

```
if(prev == null) {
    prev = data[data.length - 1];
}
```

5) If it is not the first record, we calculate the balance of this record by using the previous balance as initial balance add by deposit and subtract by withdrawal. And we compare the calculated balance with the recorded balance. We use BigDecimal class for accurate calculation.

```
BigDecimal currentBalance = new BigDecimal(data[data.length - 1]);
BigDecimal depositValue = new BigDecimal(data[data.length - 3]);
BigDecimal withdrawalValue = new BigDecimal(data[data.length - 2]);
BigDecimal initialBalance = new BigDecimal(prev);
BigDecimal calculatedBalance = initialBalance.add(depositValue).subtract(withdrawalValue);
```

6) If the calculated balance is not equal to the recorded balance, we write the record to part file.

```
if (currentBalance.compareTo(calculatedBalance) != 0) {
    outputKey.set("Date: " + data[0] + " Description: " + data[1]);
    outputValue.set("Initial Balance: " + prev + " Deposit: "+ data[data.length - 3] + " Withdrawal: "
    +data[data.length - 2] + " Calculated Balance: " + calculatedBalance.toString() + " Recorded Balance: "
    + data[data.length - 1] + " Difference: " + calculatedBalance.subtract(currentBalance));
    context.write(outputKey, outputValue);
}
```

7) Then, we assign the balance of this record to `prev` variable.

```
prev = data[data.length - 1];
```

In the main method, we create a Configuration object and set the size of input file and create Job object to configure job detail.

```
public class BalanceDetector {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        conf.set("mapreduce.input.fileinputformat.split.maxsize", "48000000");
        conf.set("mapreduce.input.fileinputformat.split.minsize", "24000000");
        Job job = Job.getInstance(conf, "BalanceDetector");
        job.setJarByClass(BalanceDetector.class);
        job.setMapperClass(BalanceDetectorMapper.class);
        job.setNumReduceTasks(0);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

The full source code of BalanceDetector is shown in `BalanceDetector.java` file in compressed attachment.

We need to download hadoop-core-1.2.1.jar from https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-core/1.2.1/hadoop-core-1.2.1.jar.

Then we compile jar file using below command.

```
mkdir balanceDetector
javac -classpath hadoop-core-1.2.1.jar -d balanceDetector BalanceDetector.java
jar -cvf BalanceDetector.jar -C balanceDetector/ . && rm -rf balanceDetector
```

We get `BalanceDetector.jar` file and assign job to hadoop mapreduce using below command.

```
hadoop jar BalanceDetector.jar BalanceDetector input/5000000\ BT\ Records.csv output
```

Since we perform only map task, we get the output in multiple part files. We need to merge the output file using below command.

```
hadoop fs -getmerge -nl output error_balance.txt && hadoop fs -rm -R output
hadoop fs -mkdir output && hadoop fs -put error_balance.txt output/error_balance.txt
rm error_balance.txt
```

We merge all parts in output folder and get the output in `error_balance.txt` file and we remove output folder in HDFS, upload `error_balance.txt` file to HDFS and remove `error_balance.txt` in local file system.