

CISC 327 Course Project

Assignment #5

Back end: unit testing

- 1. White box test for account creation transactions**
 - Code section
 - Test cases
 - List of actual test inputs
- 2. White box test for withdraw transactions**
 - Code section
 - Test cases
 - List of actual test inputs
- 3. Test Results and findings**

Company: Canada Trust

Yudong Zhou (20083467)

Yitong Liu (20028039)

Runze Yi (20073329)

Tong Bu (20079649)

White box test for account creation transactions

- Code section

TEST CASE CT1 LINES 130-203

```
130 @SuppressWarnings("resource")
131 public void mergeSummaryFile(String startFileName) throws Exception{
132
133     try{
134         File dir = new File(".");
135         File [] summaryPathList = dir.listFiles(new FilenameFilter() {
136             @Override
137             public boolean accept(File dir, String name) {
138                 return name.startsWith(startFileName);
139             }
140         });
141
142         for(File summaryPath: summaryPathList) {
143             BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(summaryPath)));
144             //data will be each line of a summary transaction file
145             String data = null;
146             data = br.readLine();
147             String dataList[] = new String[5];
148
149             String command = null;
150
151             int number = 0;
152
153             int balance = 0;
154
155             int number2 = 0;
156
157             String name = null;
158
159             while(data != null){
160                 // DataList will split each line of command into different part, CCC AAAA MMMM BBBB NNNN
161                 dataList = data.split(" ");
162                 command = dataList[0];
163                 number = Integer.parseInt(dataList[1]);
164                 balance = Integer.parseInt(dataList[2]);
165                 number2 = Integer.parseInt(dataList[3]);
166                 name = dataList[4];
167
168                 if (this.accountName.containsKey(number)) {
169                     if (this.accountBalance.get(number) < 0) {
170                         throw new Exception("Illegal Balance!");
171                     }
172                 }
173
174             }
175
176             if (this.accountName.containsKey(number2)) {
177                 if (this.accountBalance.get(number2) < 0) {
178                     throw new Exception("Illegal Balance!");
179                 }
180             }
181
182             if (command.equalsIgnoreCase("DEP")){
183                 balance += this.accountBalance.get(number).intValue();
184                 this.accountBalance.put(number, balance);
185             }
186
187         }
188     }
```

```

189     else if (command.equalsIgnoreCase("WDR")){
190         int curMoney = this.accountBalance.get(number).intValue();
191         if (curMoney < balance) {
192             throw new Exception("Illegal Balance!");
193         }
194         balance = curMoney - balance;
195         this.accountBalance.put(number,balance);
196     }
197
198     else if (command.equalsIgnoreCase("NEW")){
199
200         if(this.accountName.containsKey(number)) {
201             throw new Exception("Illegal new account! Account is being used!");
202         }
203
204
205         this.accountBalance.put(number,balance);
206         this.accountName.put(number,name);
207     }

```

TEST CASE CT2 LINES 205-207

```

208
209     else if (command.equalsIgnoreCase("DEL")){
210
211         if (this.accountBalance.get(number) != 0) {
212             throw new Exception("Illegal Balance for delete account!");
213         }
214
215         if (!(name.equalsIgnoreCase(this.accountName.get(number)))) {
216             throw new Exception("Illegal delete account! Account name is not matched!");
217         }
218
219
220         this.accountBalance.remove(number);
221         this.accountName.remove(number);
222     }
223
224     else if (command.equalsIgnoreCase("XFR")){
225
226
227         int pre = this.accountBalance.get(number2);
228
229         if(pre < balance) {
230             throw new Exception("Illegal Balance for transfer!");
231         }
232
233         this.accountBalance.put(number2,pre - balance);
234
235         pre = this.accountBalance.get(number);
236         this.accountBalance.put(number,pre + balance);
237
238     }
239
240     data = br.readLine();
241 }
242 br.close();
243 summaryPath.delete();
244
245 }
246
247 }catch(IOException e){
248     throw new IOException("Illegal IO summary transaction file!");
249 }
250
251 }

```

TEST CASE CT3 LINES 209-251

- Test cases

The table below shows the test cases we designed based on the code section to cover the white box statement testing for creation transaction.

The method *mergeSummaryFile(String startFileName)* is the code section that we need to test use white box testing. It reads and merges all summary transaction file, and also checks every transaction. Other functions in the back office class such as *readMasterFile()* and *writeMasterFile(String accountFile)* methods will also involve while running the back office.

The listing of actual test inputs (transactions from the Merged Transaction Summary File) to cover each test case will also be attached below.

Transaction			Account creation transactions	
White box method			Statement	
Statement (lines)	Test Case No.	Test case description	Input files	Output files
130-203	CT1	When an account number is created more than once in one session, the program should throw an exception and stop because of this error.	Merged transaction summary file (createAccount_testfile1) Master Account File	None
205-207	CT2	The create transaction is successfully recorded without any crash. The new account is set with name, number and balance correctly.	Merged transaction summary file (createAccount_testfile2) Master Account File	New Master Account File with the new account New Valid Account List File with the new account
209-251	CT3	The master account file is read successfully, but the transaction summary file does not exist	Transaction summary file does not exist Master Account File	None

- List of actual test inputs

Test Case No CT1:

File Name: createAccount_testfile1:

NEW 1234567 000 0000000 valid123

EOS 0000000 000 0000000 ***

File Name: MasterFile:

8765432 0 jing

7654321 5000 Jax

1234567 931000 Jack

1000000 0 wang

Test Case No CT2:

File Name: createAccount_testfile2:

NEW 1111111 000 0000000 valid123

EOS 0000000 000 0000000 ***

File Name: MasterFile:

8765432 0 jing

7654321 5000 Jax

1234567 931000 Jack

1000000 0 wang

Test Case No CT3:

None exist transaction summary file

File Name: MasterFile:

8765432 0 jing

7654321 5000 Jax

1234567 931000 Jack

1000000 0 wang

White box test for withdraw transactions

- Code section

TEST CASE WT1 BLOCK 131-251

```
130 @SuppressWarnings("resource")
131 public void mergeSummaryFile(String startFileName) throws Exception{
132
133     try{
134         File dir = new File(".");
135         File [] summaryPathList = dir.listFiles(new FilenameFilter() {
136             @Override
137             public boolean accept(File dir, String name) {
138                 return name.startsWith(startFileName);
139             }
140         });
141
142         for(File summaryPath: summaryPathList) {
143             BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(summaryPath)));
144             //data will be each line of a summary transaction file
145             String data = null;
146             data = br.readLine();
147             String dataList[] = new String[5];
148
149             String command = null;
150
151             int number = 0;
152
153             int balance = 0;
154
155             int number2 = 0;
156
157             String name = null;
158
159             while(data != null){
160                 // DataList will split each line of command into different part, CCC AAAA MMMM BBBB NNNN
161                 dataList = data.split(" ");
162                 command = dataList[0];
163                 number = Integer.parseInt(dataList[1]);
164                 balance = Integer.parseInt(dataList[2]);
165                 number2 = Integer.parseInt(dataList[3]);
166                 name = dataList[4];
167
168                 if (this.accountName.containsKey(number)) {
169                     if (this.accountBalance.get(number) < 0) {
170                         throw new Exception("Illegal Balance!");
171                     }
172                 }
173
174                 if (this.accountName.containsKey(number2)) {
175                     if (this.accountBalance.get(number2) < 0) {
176                         throw new Exception("Illegal Balance!");
177                     }
178                 }
179
180                 if (command.equalsIgnoreCase("DEP")){
181                     balance += this.accountBalance.get(number).intValue();
182                     this.accountBalance.put(number, balance);
183                 }
184             }
185         }
186     }
187 }
188
```

TEST CASE WT2 BLOCK 189-197

```
189     else if (command.equalsIgnoreCase("WDR")){
190         int curMoney = this.accountBalance.get(number).intValue();
191         if (curMoney < balance) {
192             throw new Exception("Illegal Balance!");
193         }
194         balance = curMoney - balance;
195         this.accountBalance.put(number, balance);
196     }
197 }
```

TEST CASE WT3 BLOCK 191-193

```
199     else if (command.equalsIgnoreCase("NEW")){
200
201         if(this.accountName.containsKey(number)) {
202             throw new Exception("Illegal new account! Account is being used!");
203         }
204
205         this.accountBalance.put(number, balance);
206         this.accountName.put(number, name);
207     }
208
209     else if (command.equalsIgnoreCase("DEL")){
210
211         if (this.accountBalance.get(number) != 0) {
212             throw new Exception("Illegal Balance for delete account!");
213         }
214
215         if (!(name.equalsIgnoreCase(this.accountName.get(number)))) {
216             throw new Exception("Illegal delete account! Account name is not matched!");
217         }
218
219
220         this.accountBalance.remove(number);
221         this.accountName.remove(number);
222     }
223
224     else if (command.equalsIgnoreCase("XFR")){
225
226
227         int pre = this.accountBalance.get(number2);
228
229         if(pre < balance) {
230             throw new Exception("Illegal Balance for transfer!");
231         }
232
233         this.accountBalance.put(number2, pre - balance);
234
235         pre = this.accountBalance.get(number);
236         this.accountBalance.put(number, pre + balance);
237
238     }
239
240     data = br.readLine();
241 }
242 br.close();
243 summaryPath.delete();
244
245 }
246 }catch(IOException e){
247     throw new IOException("Illegal IO summary transaction file!");
248 }
249
250
251 }
```

- Test cases

The table below shows the test cases we designed based on the code section to cover the white box statement testing for creation transaction.

The method *mergeSummaryFile(String startFileName)* is the code section that we need to test use white box testing. It reads and merges all summary transaction file, and also checks every transaction. Other functions in the back office class such as *readMasterFile()* and *writeMasterFile(String accountFile)* methods will also involve while running the back office.

The listing of actual test inputs (transactions from the Merged Transaction Summary File) to cover each test case will also be attached below.

Transaction			Withdraw transactions	
White box method			Block	
Block (lines)	Test Case No.	Test case description	Input files	Output files
131-251	WT1	The master account file is read successfully, but the transaction summary file does not exist	Transaction summary file does not exist Master Account File	None
189-197	WT2	The withdraw transaction is successfully recorded without any crash. The new account balance is set.	Merged transaction summary file (withdraw_testfile2) Master Account File	New Master Account File with the update balance New Valid Account List File
191-193	WT3	When the users withdraw more than they have, the program will recognize the negative balance and stop with an exception because of this error	Merged transaction summary file (withdraw_testfile3) Master Account File	None

- List of actual test inputs

Test Case No WT1:

File Name: withdraw_testfile1:

WDR 1234567 1000 0000000 Jack

EOS 0000000 000 0000000 ***

File Name: MasterFile:

8765432 0 jing

7654321 5000 Jax

1234567 931000 Jack

1000000 0 wang

Test Case No WT2:

File Name: withdraw_testfile2:

WDR 7654321 10000 0000000 Jax

EOS 0000000 000 0000000 ***

File Name: MasterFile:

8765432 0 jing

7654321 5000 Jax

1234567 931000 Jack

1000000 0 wang

Test Case No WT3:

None exist transaction summary file

File Name: MasterFile:

8765432 0 jing

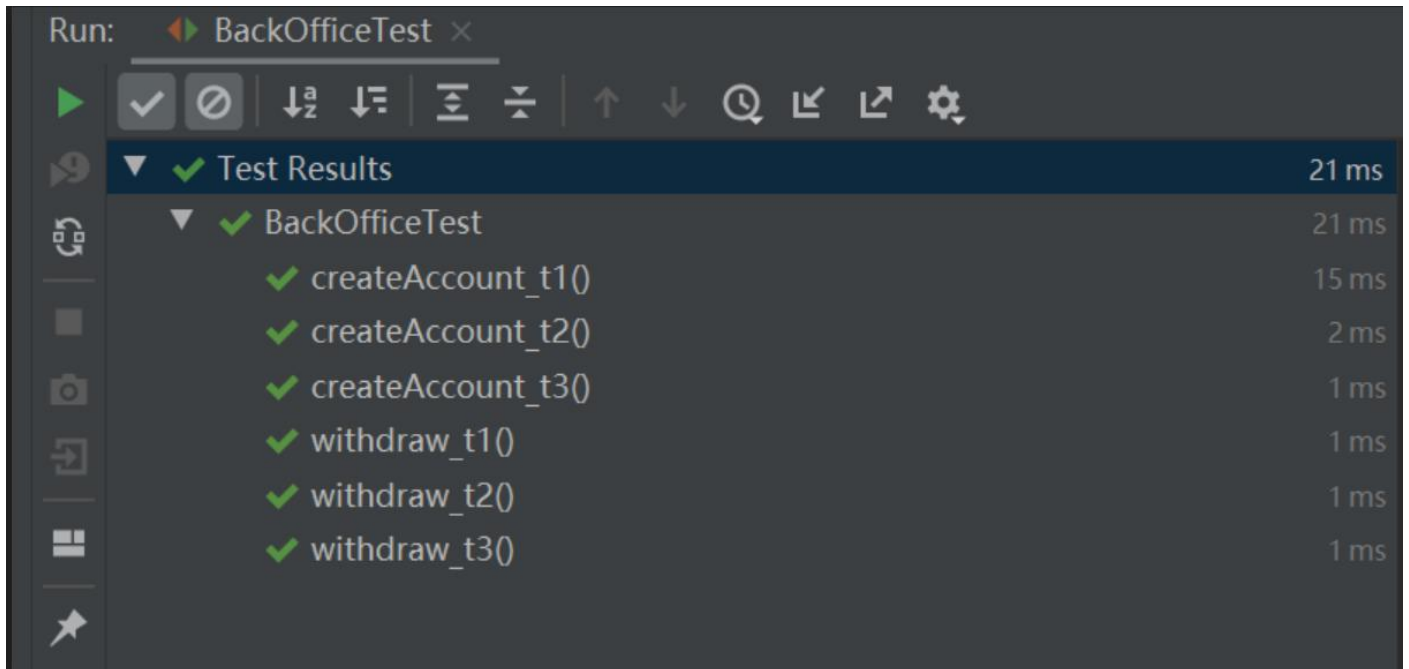
7654321 5000 Jax

1234567 931000 Jack

1000000 0 wang

- Test Results and findings

When we first test our cases, we find that the catch exception in the *mergeSummaryFile(String startFileName)* method does not catch the exception, thus the test case CT3 and the WT3 failed. After we solve the problem by changing the way of the exception, all of our test cases passed. Moreover, by comparing the master account file with the constraints, we realized that all information stored in that file should be descending order by account number instead of increasing order, we fixed this problem.



Console output

create account test case 1
Illegal new account! Account is being used!

create account test case 2
valid123
0

create account test case 3
Illegal IO summary transaction file!

withdraw test case 1
Illegal IO summary transaction file!

withdraw test case 2
930000

withdraw test case 3
Illegal Balance!

Process finished with exit code 0

Contribution

Yudong Zhou: program code and list all functionalities (20 hrs)

Yitong Liu: analysis and write the code of test cases (20 hrs)

Runze Yi: communication and work the code, check the code (20 hrs)

Tong Bu: Report writer and check report with others and also list white box test method (20 hrs)