James Penner U58460430: CS665 HW6 – Refactoring

For the first refactoring, I did up extract method on one of the methods found in the class: Transfer Transaction.java.

Before:

```
synchronized (this.fromAccount) {
            if (amount <= this.fromAccount.getCurrentBalance()) {
                Transaction t1 = new WithdrawTransaction(transactionDate,
amount, transactionOwner, fromAccount);
                Transaction t2 = new DepositTransaction(transactionDate,
amount, transactionOwner, toAccount);
            }
        }
```

After:

```
// the following refactoring is the extract method refactoring, and I have
named it extractedTransactions
        extractedTransactions(transactionDate, amount, transactionOwner,
fromAccount, toAccount);

    }

      private void extractedTransactions(Date transactionDate, int amount,
Customer transactionOwner, Account fromAccount,
                  Account toAccount) {
            synchronized (this.fromAccount) {
            if (amount <= this.fromAccount.getCurrentBalance()) {
                Transaction t1 = new WithdrawTransaction(transactionDate,
amount, transactionOwner, fromAccount);
                Transaction t2 = new DepositTransaction(transactionDate,
amount, transactionOwner, toAccount);
            }
        }
      }
```

2nd refactoring: Introduce parameter method. I have used the Bank.java class for this:

Before

```
        bankApp.createAccount(aliceId, AccountType.Checking, //exception
here to do with NullPointerException..don't understand
                "01-001", getDate("June 1, 2022"), 1000);

        String bobId = "022-22-2222";

        bankApp.createCustomer("Bob", bobId, getDate("June 1, 2022"));

        bankApp.createTransaction(TransactionType.Deposit, getDate("June 3,
2022"),
                200, aliceId,  "01-001", null);
```

```
        bankApp.setJointOwner("01-001", "011-11-1111", "022-22-2222",
                getDate("June 6, 2022"));

        bankApp.createTransaction(TransactionType.Withdraw, getDate("June
7, 2022"),
                2000, bobId, "01-001", null);

        String charlieId = "033-33-3333";


        bankApp.createCustomer("Charlie", charlieId,
                getDate("June 7, 2022"));

        bankApp.createAccount(charlieId, AccountType.Savings,
                "02-001",  getDate("June 7, 2022"), 3000);
```

After:

```
        bankApp.createAccount(new CreateAccountParameter(aliceId,
AccountType.Checking, "01-001", getDate("June 1, 2022"), 1000));

        String bobId = "022-22-2222";

        bankApp.createCustomer("Bob", bobId, getDate("June 1, 2022"));

        bankApp.createTransaction(TransactionType.Deposit, getDate("June 3,
2022"),
                200, aliceId,  "01-001", null);

        bankApp.setJointOwner("01-001", "011-11-1111", "022-22-2222",
                getDate("June 6, 2022"));

        bankApp.createTransaction(TransactionType.Withdraw, getDate("June
7, 2022"),
                2000, bobId, "01-001", null);

        String charlieId = "033-33-3333";


        bankApp.createCustomer("Charlie", charlieId,
                getDate("June 7, 2022"));

        bankApp.createAccount(new CreateAccountParameter(charlieId,
AccountType.Savings, "02-001", getDate("June 7, 2022"), 3000));
```

Also this is the new class that was created:

```
package pennerj.cs665.hw2;


import java.util.Date;


import pennerj.cs665.hw2.enumTypes.AccountType;


public class CreateAccountParameter {
```

```
        public String customerId;
        public AccountType accountType;
        public String accountId;
        public Date openDate;
        public int initialAmount;

        public CreateAccountParameter(String customerId, AccountType accountType, String
accountId, Date openDate,
                                int initialAmount) {
                this.customerId = customerId;
                this.accountType = accountType;
                this.accountId = accountId;
                this.openDate = openDate;
                this.initialAmount = initialAmount;
        }
}
```

For 3rd Refactoring am using Extract Interface, and this will be applied on the Main, or: App.java:

Before:

```
public class App {

    public static void main(String[] args) throws ParseException {

        System.out.println("\nSample Outputs ... James Penner\n");

        Bank bankApp = Bank.getBankInstance();

        String aliceId = "011-11-1111";

        bankApp.createCustomer("Alice", aliceId, getDate("June 1, 2022"));

        bankApp.createAccount(new CreateAccountParameter(aliceId,
AccountType.Checking, "01-001", getDate("June 1, 2022"), 1000));

        String bobId = "022-22-2222";

        bankApp.createCustomer("Bob", bobId, getDate("June 1, 2022"));

        bankApp.createTransaction(TransactionType.Deposit, getDate("June 3,
2022"),
                200, aliceId,  "01-001", null);
```

```
        bankApp.setJointOwner("01-001", "011-11-1111", "022-22-2222",
                getDate("June 6, 2022"));

        bankApp.createTransaction(TransactionType.Withdraw, getDate("June
7, 2022"),
                2000, bobId, "01-001", null);

        String charlieId = "033-33-3333";


        bankApp.createCustomer("Charlie", charlieId,
                getDate("June 7, 2022"));

        bankApp.createAccount(new CreateAccountParameter(charlieId,
AccountType.Savings, "02-001", getDate("June 7, 2022"), 3000));

        bankApp.setJointOwner("02-001", charlieId, bobId,
                getDate("June 7, 2022"));


        bankApp.printStatement("022-22-2222", getDate("June 8, 2022"));

        bankApp.createTransaction(TransactionType.Deposit, getDate("June 9,
2022"),
                100, bobId, "01-001", null);


        bankApp.createTransaction(TransactionType.Transfer, getDate("June
9, 2022"),
                700, bobId,  "02-001", "01-001");

        bankApp.createTransaction(TransactionType.Withdraw, getDate("June
10, 2022"),
                2000, bobId, "01-001", null);


        bankApp.printStatement("011-11-1111", getDate("June 15, 2022"));

        bankApp.printStatement("033-33-3333", getDate("June 15, 2022"));

    }

    private static Date getDate(String dateStr) {
        try {
            SimpleDateFormat formatter = new SimpleDateFormat("MMMM dd,
yyyy");
            return formatter.parse(dateStr);
        } catch (ParseException e) {
            throw new RuntimeException("Invalid Date... " +
e.getMessage());
        }
    }
}
```

After:

```
public class App implements GetDate {

    public static void main(String[] args) throws ParseException {
```

```
        System.out.println("\nSample Outputs ... James Penner\n");

        Bank bankApp = Bank.getBankInstance();

        String aliceId = "011-11-1111";

        bankApp.createCustomer("Alice", aliceId, getDate("June 1, 2022"));

        bankApp.createAccount(new CreateAccountParameter(aliceId,
AccountType.Checking, "01-001", getDate("June 1, 2022"), 1000));

        String bobId = "022-22-2222";

        bankApp.createCustomer("Bob", bobId, getDate("June 1, 2022"));

        bankApp.createTransaction(TransactionType.Deposit, getDate("June 3,
2022"),
                200, aliceId,  "01-001", null);

        bankApp.setJointOwner("01-001", "011-11-1111", "022-22-2222",
                getDate("June 6, 2022"));

        bankApp.createTransaction(TransactionType.Withdraw, getDate("June
7, 2022"),
                2000, bobId, "01-001", null);

        String charlieId = "033-33-3333";


        bankApp.createCustomer("Charlie", charlieId,
                getDate("June 7, 2022"));

        bankApp.createAccount(new CreateAccountParameter(charlieId,
AccountType.Savings, "02-001", getDate("June 7, 2022"), 3000));

        bankApp.setJointOwner("02-001", charlieId, bobId,
                getDate("June 7, 2022"));


        bankApp.printStatement("022-22-2222", getDate("June 8, 2022"));

        bankApp.createTransaction(TransactionType.Deposit, getDate("June 9,
2022"),
                100, bobId, "01-001", null);


        bankApp.createTransaction(TransactionType.Transfer, getDate("June
9, 2022"),
                700, bobId,  "02-001", "01-001");

        bankApp.createTransaction(TransactionType.Withdraw, getDate("June
10, 2022"),
                2000, bobId, "01-001", null);


        bankApp.printStatement("011-11-1111", getDate("June 15, 2022"));

        bankApp.printStatement("033-33-3333", getDate("June 15, 2022"));

    }
```

```
    private static Date getDate(String dateStr) {
        try {
            SimpleDateFormat formatter = new SimpleDateFormat("MMMM dd,
yyyy");
            return formatter.parse(dateStr);
        } catch (ParseException e) {
            throw new RuntimeException("Invalid Date... " +
e.getMessage());
        }
    }
}
```

This has also resulted in a new interface: GetDate

For the 4th Refactoring, I have used the Move method refactoring in Account.java:

Before:

```
   private final Customer primaryOwner;
    private Customer jointOwner;

    private AccountStatus accountStatus;

    private Date jointOwnershipDate;
```

and

```
    public void setJointOwner(Customer jointOwner, Date jointOwnershipDate)
{
        this.jointOwner = jointOwner;
        this.jointOwnershipDate = jointOwnershipDate;
        jointOwner.addAccount(this);
    }
```

After:

```
    private final Customer primaryOwner;
    Customer jointOwner;

    private AccountStatus accountStatus;

    Date jointOwnershipDate;
```