# Java Programming
# Section 2-4 practice

- You have included exception handling for the create button in the JavaBank application. Do the same for the make transactionbutton. try {if (noAccounts == 0) {displayJTextArea.setText("No Accounts currently created");}else {// get user inputintAccountnum = Integer.parseInt(AccountnumJTextField.getText());intDeposit = Integer.parseInt(DepositJTextField.getText());intWithdraw = Integer.parseInt(WithdrawJTextField.getText());

  for (inti=0; i<noAccounts; i++)

  {

  if ((myAccounts[i].getaccountnum() == Accountnum) && (Deposit>0))

  {

  myAccounts[i].setbalance(myAccounts[i].getBalance()+Deposit);

  displayJTextArea.setText(myAccounts[i].getaccountname() + " " + myAccounts[i].getaccountnum() + " " + myAccounts[i].getBalance());

  }

  if ((myAccounts[i].getaccountnum() == Accountnum) && (Withdraw>0))

  {

  myAccounts[i].setbalance(myAccounts[i].getBalance()-Withdraw);

  displayJTextArea.setText(myAccounts[i].getaccountname() + " " +myAccounts[i].getaccountnum() + " " + myAccounts[i].getBalance());

  }

  }

  }

  }

  catch(NumberFormatException | InputMismatchException e)

  {

  displayJTextArea.setText("");

  JOptionPane.showMessageDialog(null, "Incorrect value.");

  }

  //end catchcatch(Exception e)

```
{
System.out.println(e);
}
//end catchfinally
{
// clear other JTextFields for new dataNameJTextField.setText(" ");
AccountnumJTextField.setText("0");
BalanceJTextField.setText("0");
DepositJTextField.setText("0");
WithdrawJTextField.setText("0");
    }
}
```

- Create an exception class in the JavaBank application called “myException” that accepts a String message as a parameter in its constructor and passes the message to the super class to be printed out when an error message is thrown.

```
publicclassMyExceptionextends Exception
{
public MyException(String message)
{
super(message);
}
}
```

- Update all of the catch(Exceptione) statements in JavaBank.java to create a MyException object named newExc that sends the message "An unhandled error occurred!!" into the object.

- Surround both the method calls for the transaction and create operations in try catch statements displaying the error message in a jOptionPane if a custom exception is thrown.

- To test the custom exception, comment out all other catch statements so that only Exception e is left to handle any run time errors. Enter incorrect data for both the create and transaction functions. Uncomment the other catch statements when you have completed your tests.

## Final program:

```
import javax.swing.*;
```

```
// Main class for JavaBank application
public class JavaBank {

    // Custom exception class
    public static class MyException extends Exception {
        public MyException(String message) {
            super(message);
        }
    }

    // Class to handle account creation
    public static class CreateAccount {
        public void createAccount(String accountNumber, String amountText) throws MyException {
            try {
                if (accountNumber.isEmpty()) {
                    throw new MyException("Account number cannot be empty!");
                }
                double amount = Double.parseDouble(amountText);
                // Logic to create an account using accountNumber and amount
                System.out.println("Account created successfully with account number: " + accountNumber
+ " and amount: " + amount);
            } catch (NumberFormatException e) {
                throw new MyException("Invalid amount entered!");
            } catch (Exception e) {
                throw new MyException("An unhandled error occurred while creating the account!");
            }
        }
    }

    // Class to handle transactions
    public static class MakeTransaction {
        public void makeTransaction(String accountNumber, String amountText) throws MyException {
```

```
        try {
            if (accountNumber.isEmpty()) {
                throw new MyException("Account number cannot be empty!");
            }
            double amount = Double.parseDouble(amountText);
            // Logic to perform a transaction using accountNumber and amount
            System.out.println("Transaction successful for account number: " + accountNumber + " with amount: " + amount);
        } catch (NumberFormatException e) {
            throw new MyException("Invalid amount entered!");
        } catch (Exception e) {
            throw new MyException("An unhandled error occurred while making the transaction!");
        }
    }
}

// Class to manage bank operations
public static class BankOperations {
    private CreateAccount createAccount;
    private MakeTransaction makeTransaction;

    public BankOperations() {
        createAccount = new CreateAccount();
        makeTransaction = new MakeTransaction();
    }

    public void performCreateAccountOperation(String accountNumber, String amountText) {
        try {
            createAccount.createAccount(accountNumber, amountText);
        } catch (MyException newExc) {
            System.out.println("Error: " + newExc.getMessage());
        }
    }
```

```
    public void performMakeTransactionOperation(String accountNumber, String amountText) {
        try {
            makeTransaction.makeTransaction(accountNumber, amountText);
        } catch (MyException newExc) {
            System.out.println("Error: " + newExc.getMessage());
        }
    }
  }

  public static void main(String[] args) {
    BankOperations operations = new BankOperations();

    String accountNumber = "12345"; // Example account number
    String amountText = "100.00"; // Example amount

    operations.performCreateAccountOperation(accountNumber, amountText);
    operations.performMakeTransactionOperation(accountNumber, "50.00");
}
}
```

```
C:\Users\91984\Downloads\java>javac JavaBank.java

C:\Users\91984\Downloads\java>java JavaBank
Account created successfully with account number: 12345 and amount: 100.0
Transaction successful for account number: 12345 with amount: 50.0
```