Lecture 14 - handout

Recapping MVC and Exceptions (the Banking System)

The idea of MVC is to decouple parts of systems that can be implemented in different ways, so we can swap out different implementations for different needs (e.g., different interfaces, different Customer classes).

MVC also helps illustrate the value of exceptions. Consider this chain of calls that process a login.

```
// in BankingConsole/View
public void loginScreen() {
   // read username & password
   this.B.login(username, password)
   // show menu
}



// in BankingService/Controller
public void login(String cname, String pwd) {
    Customer cust = findCustomer(cname);
    cust.checkPwd(pwd);
}



// in Customer\Model
public void checkPwd(String givenPwd) {
   this.password.equals(givenPwd);
}
```

```
// in BankingConsole/View
public void loginScreen() {
   if (this.B.login(username, password)) {
     // show menu
   } else {
     System.out.println("Login Failed")
     this.loginScreen()
   }
}

// in BankingService/Controller
public boolean login(String cname, String pwd) {
   Customer cust = findCustomer(cname);
   if (cust == null) {
      cust.checkPwd(pwd);
   } else {
      return false;
   }
 }

// in Customer\Model
public boolean checkPwd(String givenPwd) {
   return this.password.equals(givenPwd);
}
```

```
// in BankingConsole/View
public void loginScreen() {
   if (this.B.login(username, password)) {
     // show menu
   } else {
     System.out.println(“Login Failed”)
     this.loginScreen()
   }
}
```

```
// in BankingService/Controller
public boolean login(String cname, String pwd) {
   Customer cust = findCustomer(cname);
   if (cust == null) {
      cust.checkPwd(pwd);
   } else {
      return false;
   }
 }
```

```
// in Customer\Model
public boolean checkPwd(String givenPwd) {
   return this.password.equals(givenPwd);
}
```

```
// in BankingConsole/View
public void loginScreen() {
   // read username & password
   try {
      this.B.login(username, password)
      // show menu
   } catch (CustNotFoundException e) {
      System.out.println(“Unknown username”);
      this.loginScreen();
   } catch (WrongPwdException e) {
      System.out.println(“Wrong password”);
      this.loginScreen();
   }
}
```

```
// in BankingService/Controller
public void login(String cname, String pwd) {
    Customer cust = findCustomer(cname);
    cust.checkPwd(pwd);
}
```

```
// in Customer\Model
public void checkPwd(String givenPwd) {
   if (!(this.password.equals(givenPwd))
     throw new WrongPwdException();
}
```

```
// in BankingConsole/View
public void loginScreen() {
   // read username & password
   try {
      this.B.login(username, password)
      // show menu
   } catch (CustNotFoundException e) {
      System.out.println(“Unknown username”);
      this.loginScreen();
   } catch (WrongPwdException e) {
      System.out.println(“Wrong password”);
      this.loginScreen();
   }
}

// in BankingService/Controller
public void login(String cname, String pwd) {
    Customer cust = findCustomer(cname);
    cust.checkPwd(pwd);
}

// in Customer\Model
public void checkPwd(String givenPwd) {
   if (!(this.password.equals(givenPwd))
     throw new WrongPwdException();
}
```

Improving Runtime of accessing Customers and Accounts

```
// in BankingService
LinkedList<Account> accounts;
LinkedList<Customer> customers;

private Account findAccount(int forAcctNum) throws AcctNotFoundException {
  for (Account acct:accounts) {
     if (acct.numMatches(forAcctNum)) return acct;
  }
  throw new AcctNotFoundException(forAcctNum);
}

private Customer findCustomer(String custname) throws CustNotFoundException {
  for (Customer cust:customers) {
     if (cust.nameMatches(custname)) {
        return cust;
     }
   }
   throw new CustNotFoundException(custname);
}
```

```
// in BankingService
LinkedList<Account> accounts;



private Account findAccount(int forAcctNum) throws AcctNotFoundException {

  for (Account acct:accounts) {
     if (acct.numMatches(forAcctNum)) return acct;
  }




  throw new AcctNotFoundException(forAcctNum);
}
```