Exceptions and the call stack

Friday, October 7, 2022 1:06 PM

```
// 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();
}
```

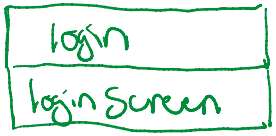
throws WrongPwdException

stop executing

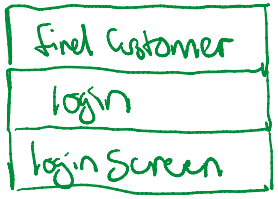
Call stack

1. loginScreen is called

2. loginScreen calls login

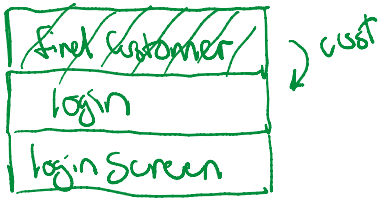


3. login calls findCustomer



4. findCustomer returns cust

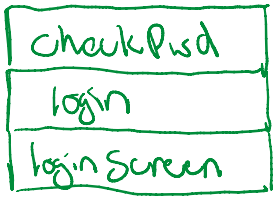
(method is done; no longer on the call stack)



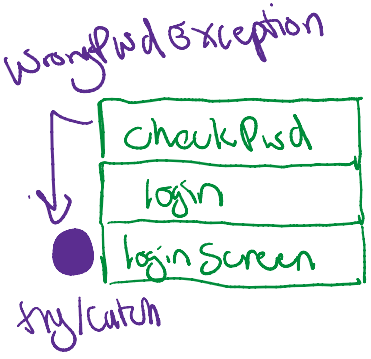
In normal operation, data is passed down the call stack as each method returns

5. login resumes executing;

calls checkPwd



6. checkPwd throws an exception, stops executing, Discards methods on call stack until reaches nearest try/catch to handle that exception



try/catch

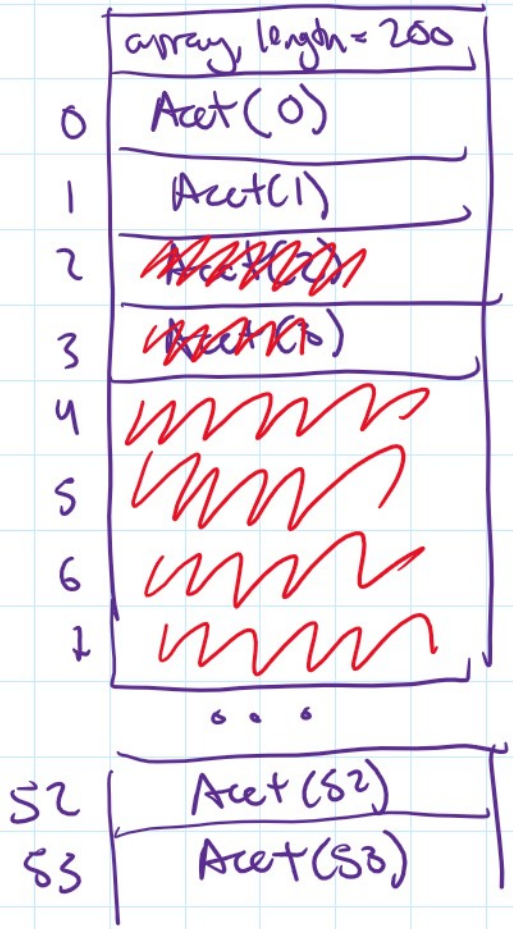
Improving access time

Friday, October 7, 2022 1:07 PM

```
// 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);
}
```



What if we tried to improve the runtime of findAccount by storing all of the accounts in an array, indexed by the account number? This would create constant-time lookup for the accounts!

Some issues with this approach:

- If customers close accounts, lots of unused memory space
- Accounts numbers are not given sequentially in real life
- Cannot apply this same approach to customers, because Strings are not array indices

HashMaps

Friday, October 7, 2022 1:07 PM

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

HashMap<Integer, Account> accounts;

"key" "value"

HashMaps/Dictionaries/HashTables
(all the same thing)
Allow for constant-time lookup of a key matched to a value

```
private Account findAccount(int forAcctNum) throws AcctNotFoundException {

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

  throw new AcctNotFoundException(forAcctNum);
}
```

if (accounts.containsKey(forAcctNum)) {
  return accounts.get(forAcctNum);
}

for customers,
HashMap<String, Customer>
customers;

customers.get(custname)

```
HashMap<String, String> labRooms = new HashMap<>;
labRooms.put("Whitney", "CIT 219");
labRooms.put("Nathan", "CIT 410");
labRooms.put("Raphael", "CIT 219");
labRooms.put("Nathan", "CIT 165"); // cannot have a key point to two values simultaneously!
// (just like each index of an Array just stores one value)
labRooms.get("Whitney"); // returns "CIT 219" // Where's Whitney's lab?
labRooms.get("Nathan"); // returns "CIT 165" // last stored value

HashMap<String, List<String>> multiLabRooms = new HashMap<>();
// a way to map a key to  multiple values -- note that we have to create a whole new HashMap for this
multiLabRooms.put("Whitney", ["CIT 219", "CIT 410"]) // (shorthand for list)
```