Sequence of Stack Frames for Explaining Banking Exceptions

CS2102

Professor Kathi Fisler

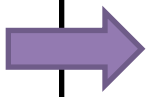
This demonstration accompanies the BankingService example that we have been doing for the last several lectures.

It starts from the point when we decided to throw an exception rather than return null when a named customer is not found.

The lecture notes provide context for the starting point of these slides

```
loginScreen()
```

The Stack



```
public void loginScreen() {
    // prompt for name and password
    System.out.println("Welcome …”);
    System.out.print("Enter username: ");
    String username = keyboard.next();
    System.out.print("Enter password: ");
    int password = keyboard.nextInt();
    try {
      B.login(username,password);
      System.out.println("Login success");
    } catch (CustNotFoundException e) {
      System.out.println(”Try Again");
      this.loginScreen();
    }
}
```

The Current Method

As the user is about to type in their password (having already typed their username)

```
loginScreen()
```

The Stack

```
public void loginScreen() {
    // prompt for name and password
    System.out.println("Welcome …”);
    System.out.print("Enter username: ");
    String username = keyboard.next();
    System.out.print("Enter password: ");
    int password = keyboard.nextInt();
    try {
      B.login(username,password);
      System.out.println("Login success");
    } catch (CustNotFoundException e) {
      System.out.println(”Try Again");
      this.loginScreen();
    }
}
```

The Current Method

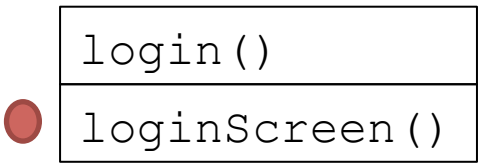
Java reads the typed-in password

The Stack

```
public void loginScreen() {
    // prompt for name and password
    System.out.println("Welcome …”);
    System.out.print("Enter username: ");
    String username = keyboard.next();
    System.out.print("Enter password: ");
    int password = keyboard.nextInt();
    try {
      B.login(username,password);
      System.out.println("Login success");
    } catch (CustNotFoundException e) {
      System.out.println(”Try Again");
      this.loginScreen();
    }
}
```

The Current Method

Java reads sends the username and password to the login method in the BankingService class. Java puts a little marker on the stack to indicate that we are in a try block.



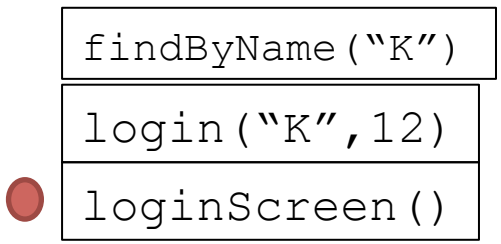
The Stack

```
public void login(String custname,
                  int withPwd) {
   Customer cust =
        customers.findByName(custname);
   cust.tryLogin(withPwd);
}
```

Note this call to tryLogin after the arrow. If findByName throws an exception, this line will be skipped

The Current Method

We record on the stack that we are now in a call to login, and the login code is the current method. The code arrow is at the top of login. We are about to call findByName.

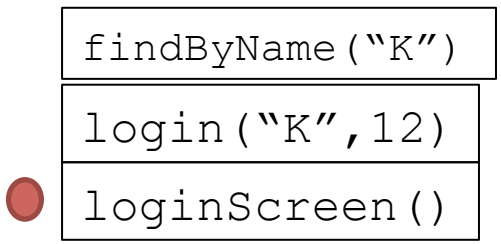


The Stack

```
public Customer findByName(String name) {
  for (Customer cust:customers) {
    if (cust.nameMatches(name))
      return cust;
  }
  throw new CustNotFoundException(name);
}
```

The Current Method

We record on the stack that we are now in a call to findByName, which becomes the current method. The arrow is at the start of the for loop to check the customers.

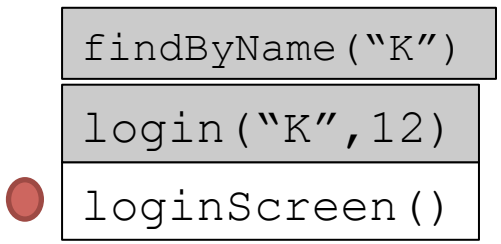


The Stack

```
public Customer findByName(String name) {
  for (Customer cust:customers) {
    if (cust.nameMatches(name))
      return cust;
  }
  throw new CustNotFoundException(name);
}
```

The Current Method

Fast forward – we didn't find the customer, and the arrow is at the point of the exception. Java "throws" the exception.



The Stack

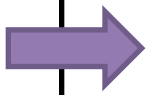
```
public void loginScreen() {
    // prompt for name and password
    System.out.println("Welcome …");
    System.out.print("Enter username: ");
    String username = keyboard.next();
    System.out.print("Enter password: ");
    int password = keyboard.nextInt();
    try {
      B.login(username,password);
      System.out.println("Login success");
    } catch (CustNotFoundException e) {
      System.out.println("Try Again");
      this.loginScreen();
    }
}
```

The Current Method

Java now searches back through the stack for the try marker, ignoring the other method calls that were waiting to finish (this is where the pending call to tryLogin gets skipped). The arrow moves to the start of the catch block.

```
public void loginScreen() {
    // prompt for name and password
    System.out.println("Welcome …”);
    System.out.print("Enter username: ");
    String username = keyboard.next();
    System.out.print("Enter password: ");
    int password = keyboard.nextInt();
    try {
      B.login(username,password);
      System.out.println("Login success");
    } catch (CustNotFoundException e) {
      System.out.println(”Try Again");
      this.loginScreen();
    }
}
```

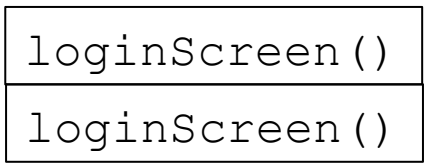
`loginScreen()`



The Stack

The Current Method

The skipped-over methods are removed from the stack, and Java continues running the code within the try block. The marker comes off because we have finished try part of the block



The Stack

```
public void loginScreen() {
    // prompt for name and password
    System.out.println("Welcome …”);
    System.out.print("Enter username: ");
    String username = keyboard.next();
    System.out.print("Enter password: ");
    int password = keyboard.nextInt();
    try {
      B.login(username,password);
      System.out.println("Login success");
    } catch (CustNotFoundException e) {
      System.out.println(”Try Again");
      this.loginScreen();
    }
}
```

The Current Method

The arrow advances to the new call to loginScreen, which now goes on the stack, and the arrow starts again at the top of the loginScreen method. When we get to the try statement, a new marker will go on the stack on the upper loginScreen call.