Lesson 28: `FileManager` and Polymorphism (W08D3)

Balboa High School

Michael Ferraro

October 7, 2015

Do Now

- Create a new project named `Lesson28`
- Download `Product.java` from here and import into `Lesson28`
- Replace instances of the string "`?????`" with appropriate values so that the program works correctly.

Aim

Students will revisit OO ideas, have their first experience with *polymorphism*, and discuss recursion in filesystems to assist with the `FileManager` portion of PS #4b.

Let's Revisit the Bank!

- Create new project `Lesson28`
- Import these sources from here:
  - `BankAccount.java`
  - `BankAccountDriver.java`
  - `CheckingAccount.java`
  - `SavingsAccount.java`
  - . . . **OR** download and import `BankAccount-ALL.jar`
- Find the new method that has been added to each of the object classes and be ready to explain its behavior

The **IS-A** Relationship

- Recall from an earlier lesson the “**IS-A**” relationship

The **IS-A** Relationship

- Recall from an earlier lesson the "**IS-A**" relationship
    - A `CheckingAccount` object **IS-A** `BankAccount`

The **IS-A** Relationship

- Recall from an earlier lesson the "**IS-A**" relationship
    - A `CheckingAccount` object **IS-A** `BankAccount`
    - A `SavingsAccount` object **IS-A** `BankAccount`

The **IS-A** Relationship

- Recall from an earlier lesson the "**IS-A**" relationship
  - A `CheckingAccount` object **IS-A** `BankAccount`
  - A `SavingsAccount` object **IS-A** `BankAccount`
- Since every kind of account we can have is considered a `BankAccount` object, we can create an `ArrayList` to hold `BankAccounts`!

```
ArrayList<BankAccount> listOfAccts =
   new ArrayList<BankAccount>();
```

The **IS-A** Relationship

- Recall from an earlier lesson the "**IS-A**" relationship
  - A `CheckingAccount` object **IS-A** `BankAccount`
  - A `SavingsAccount` object **IS-A** `BankAccount`
- Since every kind of account we can have is considered a `BankAccount` object, we can create an `ArrayList` to hold `BankAccount`s!

```
ArrayList<BankAccount> listOfAccts =
   new ArrayList<BankAccount>();
```

- Look through the driver class and predict its new behavior

Superb Customer Service

- When iterating through the objects contained in `ArrayList<BankAccount> listOfAccounts` using the `for-each()` loop, Java encounters `BankAccount` objects.

Superb Customer Service

- When iterating through the objects contained in `ArrayList<BankAccount> listOfAccounts` using the `for-each()` loop, Java encounters `BankAccount` objects.
- Each such object might be a...

Superb Customer Service

- When iterating through the objects contained in `ArrayList<BankAccount> listOfAccounts` using the `for-each()` loop, Java encounters `BankAccount` objects.
- Each such object might be a...
  - `CheckingAccount`,

Superb Customer Service

- When iterating through the objects contained in `ArrayList<BankAccount> listOfAccounts` using the `for-each()` loop, Java encounters `BankAccount` objects.
- Each such object might be a...
  - `CheckingAccount`,
  - `SavingsAccount`, or

Superb Customer Service

- When iterating through the objects contained in `ArrayList<BankAccount> listOfAccounts` using the `for-each()` loop, Java encounters `BankAccount` objects.
- Each such object might be a...
    - `CheckingAccount`,
    - `SavingsAccount`, or
    - `BankAccount`.

Superb Customer Service

- When iterating through the objects contained in `ArrayList<BankAccount> listOfAccounts` using the `for-each()` loop, Java encounters `BankAccount` objects.
- Each such object might be a...
  - `CheckingAccount`,
  - `SavingsAccount`, or
  - `BankAccount`.
- **Question:** So when Java works with one of these objects, how does it decide which class' `getAccountSummary()` method to call?

Superb Customer Service

- When iterating through the objects contained in `ArrayList<BankAccount> listOfAccounts` using the `for-each()` loop, Java encounters `BankAccount` objects.
- Each such object might be a. . .
  - `CheckingAccount`,
  - `SavingsAccount`, or
  - `BankAccount`.
- **Question:** So when Java works with one of these objects, how does it decide which class' `getAccountSummary()` method to call?
- **Answer:** Java will always call the most specific version available.

Superb Customer Service

- When Java retrieves object `chkgAcct1`, it understands that it's a `BankAccount`; but more specifically, it's an object of type `CheckingAccount`

Superb Customer Service

- When Java retrieves object `chkgAcct1`, it understands that it's a `BankAccount`; but more specifically, it's an object of type `CheckingAccount`
- *Java's inner voice:*

Superb Customer Service

- When Java retrieves object `chkgAcct1`, it understands that it's a `BankAccount`; but more specifically, it's an object of type `CheckingAccount`
- *Java's inner voice:*
  - if there's an implementation of `getAccountSummary()` in `CheckingAccount`, let's use that one

Superb Customer Service

- When Java retrieves object `chkgAcct1`, it understands that it's a `BankAccount`; but more specifically, it's an object of type `CheckingAccount`
- *Java's inner voice:*
    - if there's an implementation of `getAccountSummary()` in `CheckingAccount`, let's use that one
    - else, fall back to the inherited `getAccountSummary()` from superclass `BankAccount`

Superb Customer Service

- When Java retrieves object `chkgAcct1`, it understands that it's a `BankAccount`; but more specifically, it's an object of type `CheckingAccount`
- *Java's inner voice:*
    - if there's an implementation of `getAccountSummary()` in `CheckingAccount`, let's use that one
    - else, fall back to the inherited `getAccountSummary()` from superclass `BankAccount`
- Do the same for the remaining elements in the `ArrayList`. . .

Polymorphism

- **POLY** = *many*

Polymorphism

- **POLY** = *many*
- **MORPH** = *shape* or *form*

Polymorphism

- **POLY** = *many*
- **MORPH** = *shape* or *form*
- **Polymorphism:** In the OO world, this is the ability for a programming language to treat objects as members of a superclass (e.g., handle `CheckingAccounts` and `SavingsAccounts` as `BankAccounts`), but to also interact with these objects using the most specific methods available.

Polymorphism

- **POLY** = *many*
- **MORPH** = *shape* or *form*
- **Polymorphism:** In the OO world, this is the ability for a programming language to treat objects as members of a superclass (e.g., handle `CheckingAccounts` and `SavingsAccounts` as `BankAccounts`), but to also interact with these objects using the most specific methods available.
- You'll see another example of polymorphism in the `FileManager` section of PS #4b

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]

[1] Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a <u>directory</u>?

[1] Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
    - …aka *folder*, the more modern metaphor

[1] Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
  - ...aka *folder*, the more modern metaphor
  - A logical storage unit or container for:

[1] Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
    - ...aka *folder*, the more modern metaphor
    - A logical storage unit or container for:
        - files (logical storage units in their own right) and/or

[1] Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
    - ...aka *folder*, the more modern metaphor
    - A logical storage unit or container for:
        - files (logical storage units in their own right) and/or
        - more directories!

[1]Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
    - ...aka *folder*, the more modern metaphor
    - A logical storage unit or container for:
        - files (logical storage units in their own right) and/or
        - more directories!
- Directories within directories,

[1] Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
    - . . . aka *folder*, the more modern metaphor
    - A logical storage unit or container for:
        - files (logical storage units in their own right) and/or
        - more directories!
- Directories within directories,
  within directories,

[1] Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a <u>directory</u>?
    - ...aka *folder*, the more modern metaphor
    - A logical storage unit or container for:
        - files (logical storage units in their own right) and/or
        - more directories!
- Directories within directories,

  within directories,

  within directories,

[1]Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
    - …aka *folder*, the more modern metaphor
    - A logical storage unit or container for:
        - files (logical storage units in their own right) and/or
        - more directories!
- Directories within directories,
  within directories,
  within directories,
  within …

[1]Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
  - . . . aka *folder*, the more modern metaphor
  - A logical storage unit or container for:
    - files (logical storage units in their own right) and/or
    - more directories!
- Directories within directories,
  within directories,
  within directories,
  within . . .

  What does that seem to suggest?

[1]Make sure you read the additional details in the problem set.

Litvin's `FileManager` Lab

- Make sure you do the required reading for §7 of PS #4b, Litvin §13.4[1]
- What is a directory?
  - ...aka *folder*, the more modern metaphor
  - A logical storage unit or container for:
    - files (logical storage units in their own right) and/or
    - more directories!
- Directories within directories,
  within directories,
  within directories,
  within ...

What does that seem to suggest? **RECURSION!**

[1]Make sure you read the additional details in the problem set.

HW

Work on §7 of PS #4b, making sure to read the relevant parts of the problem set **and** the textbook.