Session 5 – Object-Oriented Design

Dao Tran Hoang Chau

Modified from Introduction to Object-Oriented Analysis and Design with UML and Unified Process, Stephen R. Schach

Chapter Overview

- The Design Workflow
- Traditional versus Object-Oriented Design
- Formats of the Attributes
- Allocation of Operations to Classes
- Allocation of Operations: Osbert Oglesby Case Study
- CRC Cards

The Design Workflow

- Recap on analysis workflow
- What and Why in design workflow

Additional tasks

- Make decisions:
  - Programming language
  - Reuse
  - Portability
  - Architecture

Analysis Package -> Subsystems

- Analysis workflow
  - analysis package*:* a set of related classes
- Why decomposition?
  - Easier

Subsystems – Definition

- break up the upcoming implementation workflow into manageable pieces : subsystems

Subsystem – Why ?

- Easier
- Doing in parallel

Subsystem – What ?

- *Software architecture* of an information system includes
  - Component modules
  - How they fit together
  - The allocation of components to subsystems

Architect: identify trade-off in design

- The architect needs to make *trade-offs*
  - functional requirements – nonfunctional requirements – time and budget

- The architect must assist the client by laying out the trade-offs

The architecture is very important

- There is no way to recover from suboptimal architecture
    - must immediately be redesigned

Artifacts of the design workflow

- complete class diagram

Identify class attributes' format

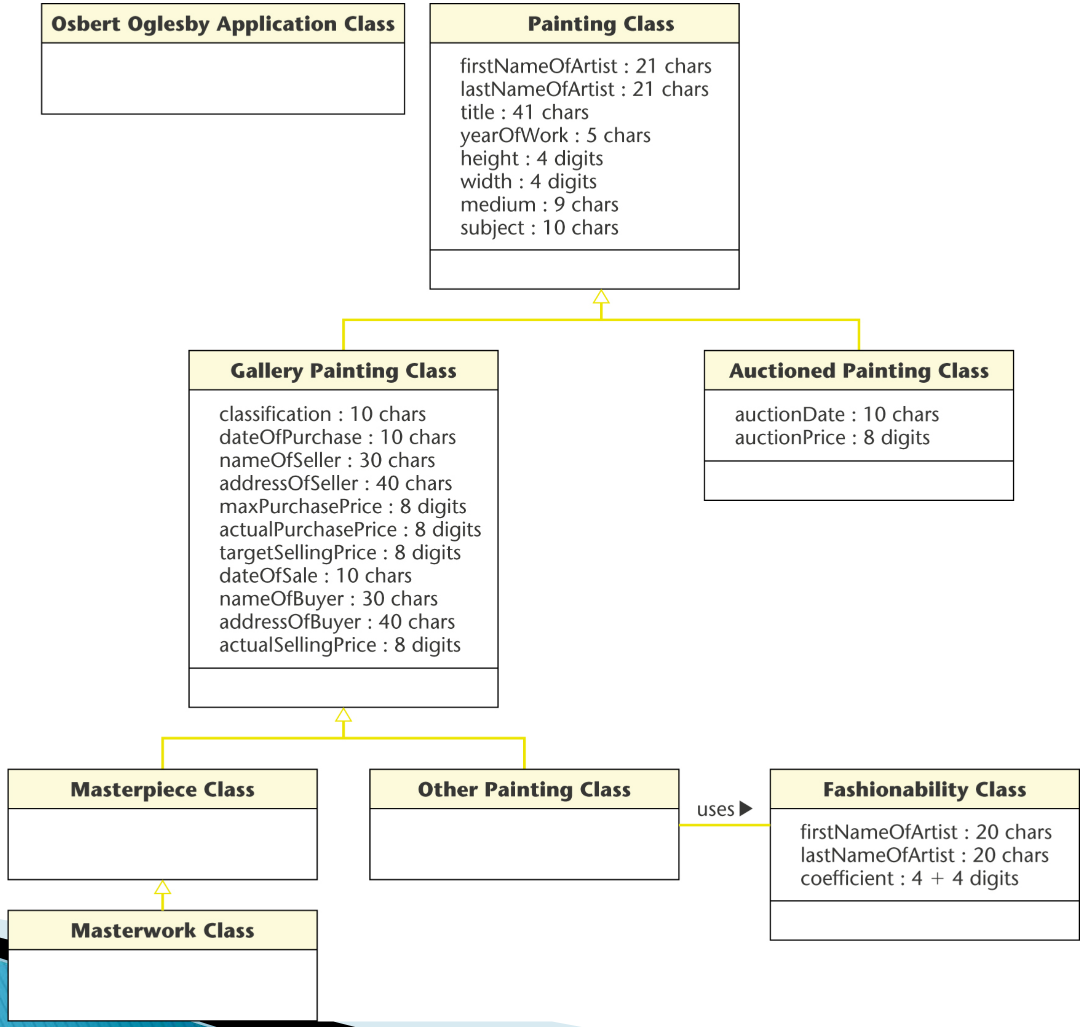
- Exact format of each attribute of the class diagram must be specified
  - Example:
  - A date is usually represented by 10 characters
  - December 3, 1947 is represented as
    - 12/03/1947 in the United States (MM/DD/YYYY format), and
    - 03/12/1947 in Europe (DD/MM/YYYY format)

Identify class attributes' format

- The formats **could be determined during the analysis workflow**
- However, the object-oriented paradigm is iterative
  - Information is added to models **as late as possible**

Formats of Attributes of Osbert Oglesby

- Class diagram with the formats of attributes added



Osbert Oglesby Application Class

Painting Class

firstNameOfArtist : 21 chars
lastNameOfArtist : 21 chars
title : 41 chars
yearOfWork : 5 chars
height : 4 digits
width : 4 digits
medium : 9 chars
subject : 10 chars

Gallery Painting Class

classification : 10 chars
dateOfPurchase : 10 chars
nameOfSeller : 30 chars
addressOfSeller : 40 chars
maxPurchasePrice : 8 digits
actualPurchasePrice : 8 digits
targetSellingPrice : 8 digits
dateOfSale : 10 chars
nameOfBuyer : 30 chars
addressOfBuyer : 40 chars
actualSellingPrice : 8 digits

Auctioned Painting Class

auctionDate : 10 chars
auctionPrice : 8 digits

Masterpiece Class

Other Painting Class

uses ▶

Fashionability Class

firstNameOfArtist : 20 chars
lastNameOfArtist : 20 chars
coefficient : 4 + 4 digits

Masterwork Class

Formats of Attributes of Osbert Oglesby (contd)

- Examples:
  - First name of an artist is up to 20 characters in length, optionally followed by ? if there is uncertainty
    - firstNameOfArtist : 21 chars
  - Title is up to 40 characters in length, optionally with ?
    - title : 41 chars
  - Height and width are measured in centimeters
    - height, width : 4 digits (up to 9999 centimeters, or 99.99 meters)
  - Prices
    - targetSellingPrice, actualSellingPrice, maxPurchasePrice : 8 digits (up to $99,999,999)
  - Dates
    - dateOfPurchase, dateOfSale, auctionDate : 10 chars

Formats of Attributes of Osbert Oglesby (contd)

- Fashionability coefficient
  - This could be a large number or a small number

- The range can be determined only by computing coefficients from a sample of Osbert’s sales
  - High: 985 (Rembrandt van Rijn)
  - Low: 0.064 (Joey T. Dog)

Formats of Attributes of Osbert Oglesby (contd)

- For safety, coefficient is of type 4 + 4 digits

- What is the range ?

Allocation of Operations to Classes (contd)

- Where is the operation in the class diagram ?
- Why do we add it so late
- How do we identify operations ?

Allocation of Operations to Classes (contd)

- Identifying the operations to be allocated to the various classes is easy

- Determining to which class each operation should be allocated is hard

- Three criteria are used
  - Responsibility-driven design
  - Inheritance
  - Polymorphism and dynamic binding (Chapter 20)

Responsibility-Driven Design

- The principle of *responsibility-driven design*
  - If **Class A** sends a message to **Class B** telling it to do something, it is the responsibility of **Class B** to perform the requested operation

Inheritance – recap

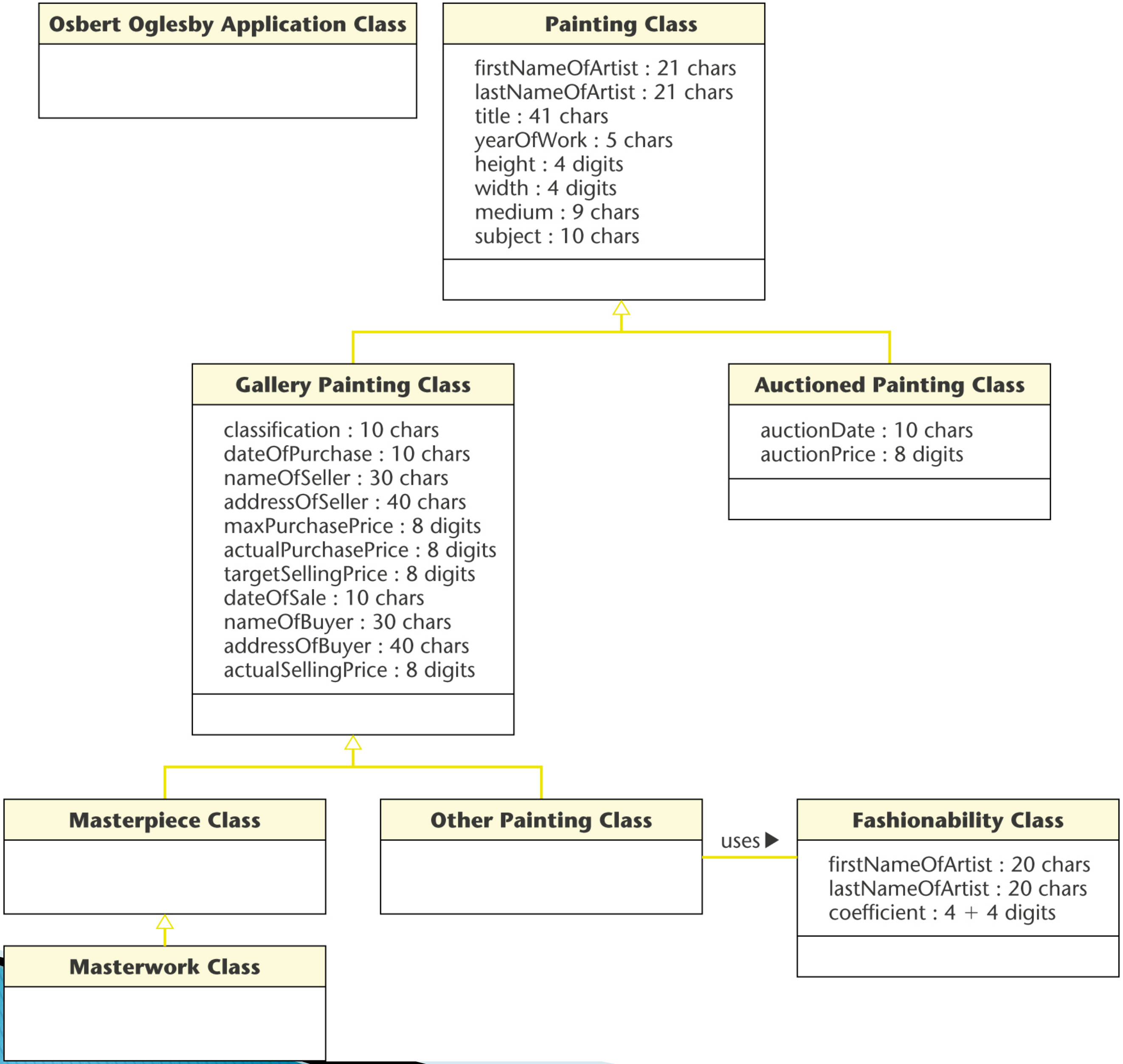
- Suppose an operation is applicable to
  - An instance of a superclass; and to
  - Instances of subclasses of that superclass

- Allocate that operation to the superclass

- Then there is just one version of that operation

- It can be used by
  - Instances of the superclass and by
  - Instances of all its subclasses

Allocation of Operations: Osbert Oglesby

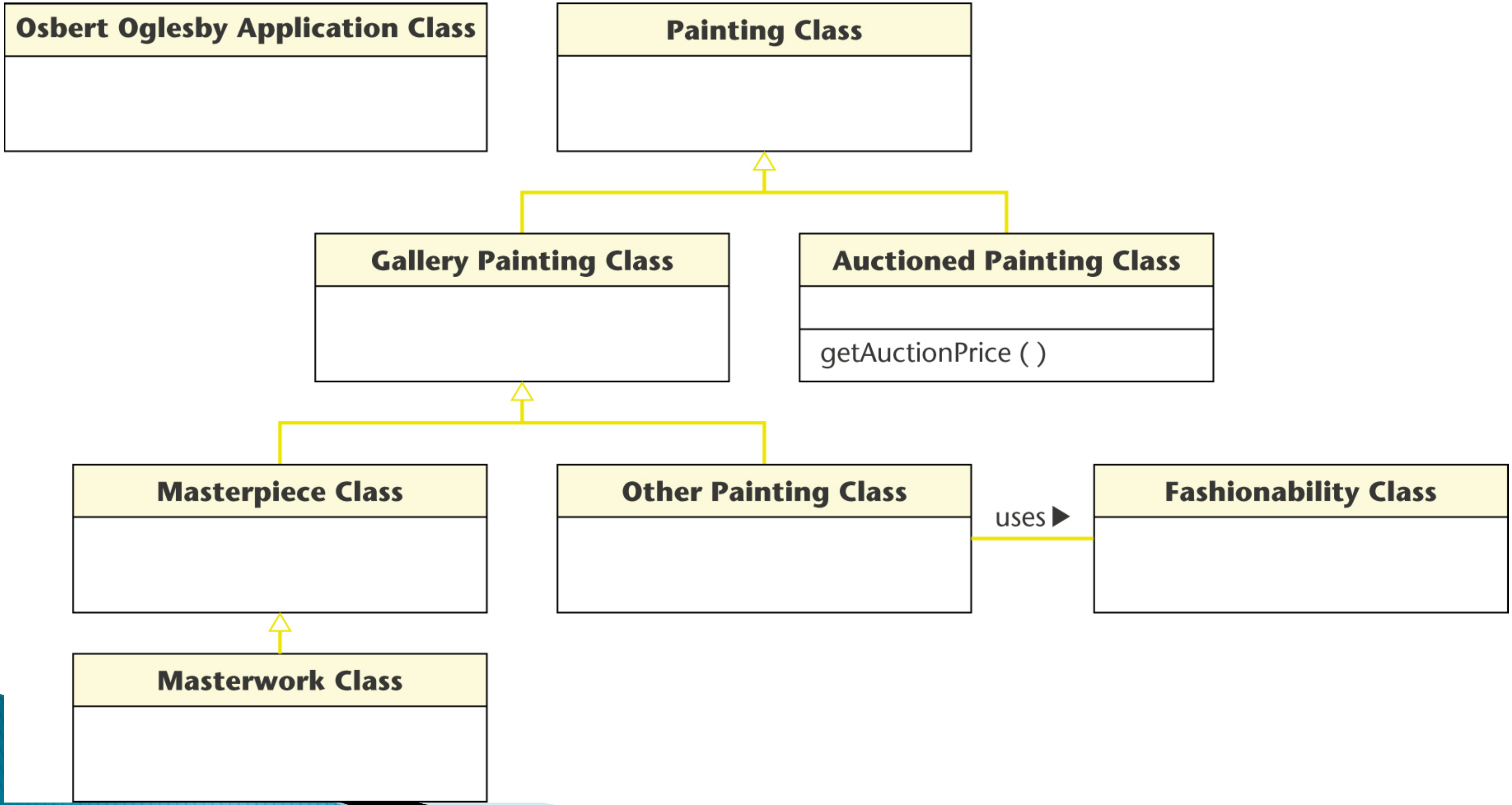

Osbert Oglesby Application Class
Painting Class
firstNameOfArtist : 21 chars
lastNameOfArtist : 21 chars
title : 41 chars
yearOfWork : 5 chars
height : 4 digits
width : 4 digits
medium : 9 chars
subject : 10 chars
Gallery Painting Class
classification : 10 chars
dateOfPurchase : 10 chars
nameOfSeller : 30 chars
addressOfSeller : 40 chars
maxPurchasePrice : 8 digits
actualPurchasePrice : 8 digits
targetSellingPrice : 8 digits
dateOfSale : 10 chars
nameOfBuyer : 30 chars
addressOfBuyer : 40 chars
actualSellingPrice : 8 digits
Auctioned Painting Class
auctionDate : 10 chars
auctionPrice : 8 digits
Masterpiece Class
Other Painting Class
uses ▶
Fashionability Class
firstNameOfArtist : 20 chars
lastNameOfArtist : 20 chars
coefficient : 4 + 4 digits
Masterwork Class

Responsibility-Driven Design: Osbert Oglesby

- Consider operation *getAuctionPrice()*, where should we put it ?

Responsibility-Driven Design: Osbert Oglesby

- Allocation of getAuctionPrice



Set and get methods

- What is set and get method ?

Inheritance: Osbert Oglesby Case Study

- Consider operations of all types of paintings
  - `setTitle` and
  - `getTitle`
- Where should we put these methods?

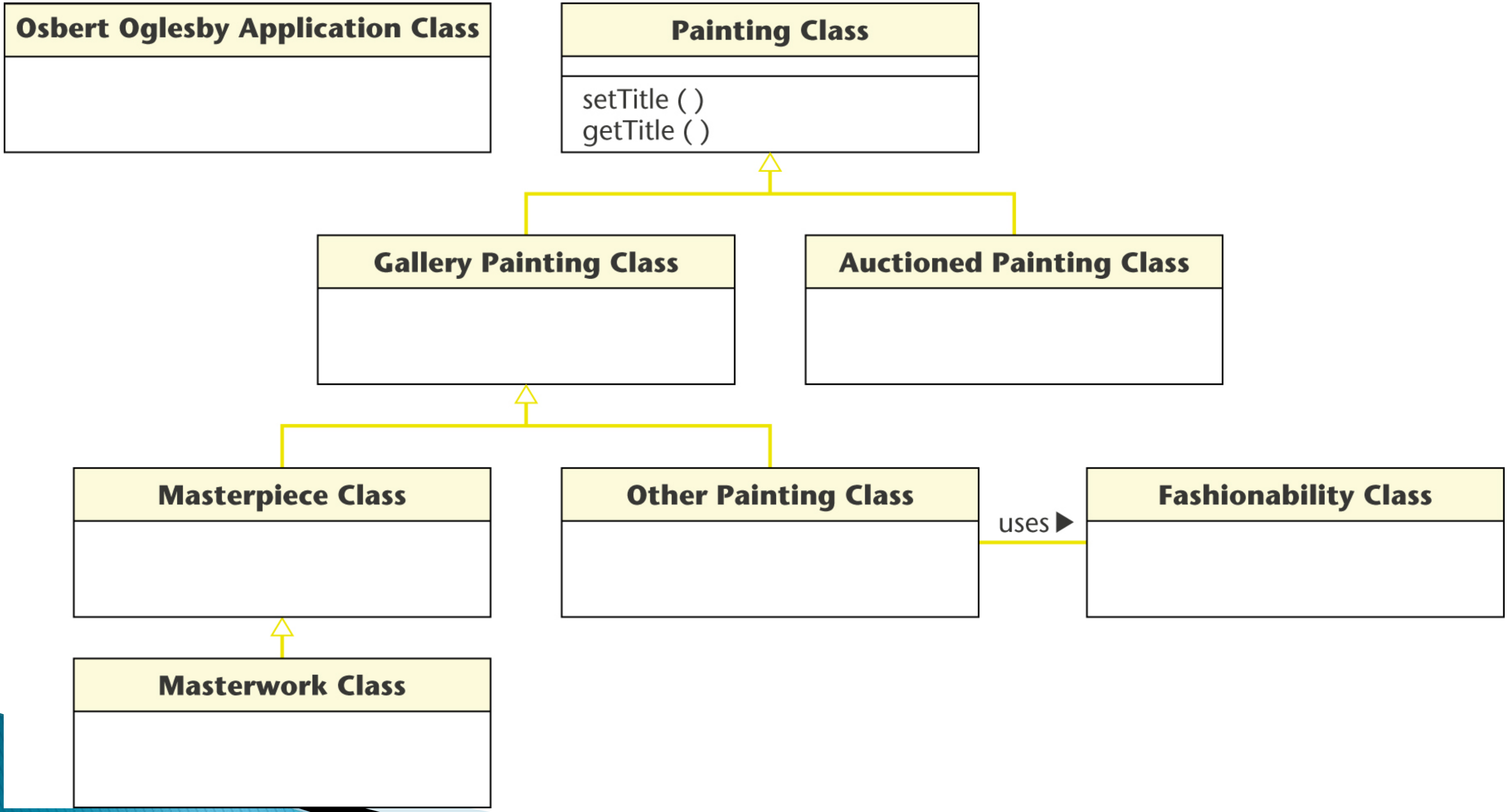
Inheritance: Osbert Oglesby Case Study (contd)

- First consider operation setTitle

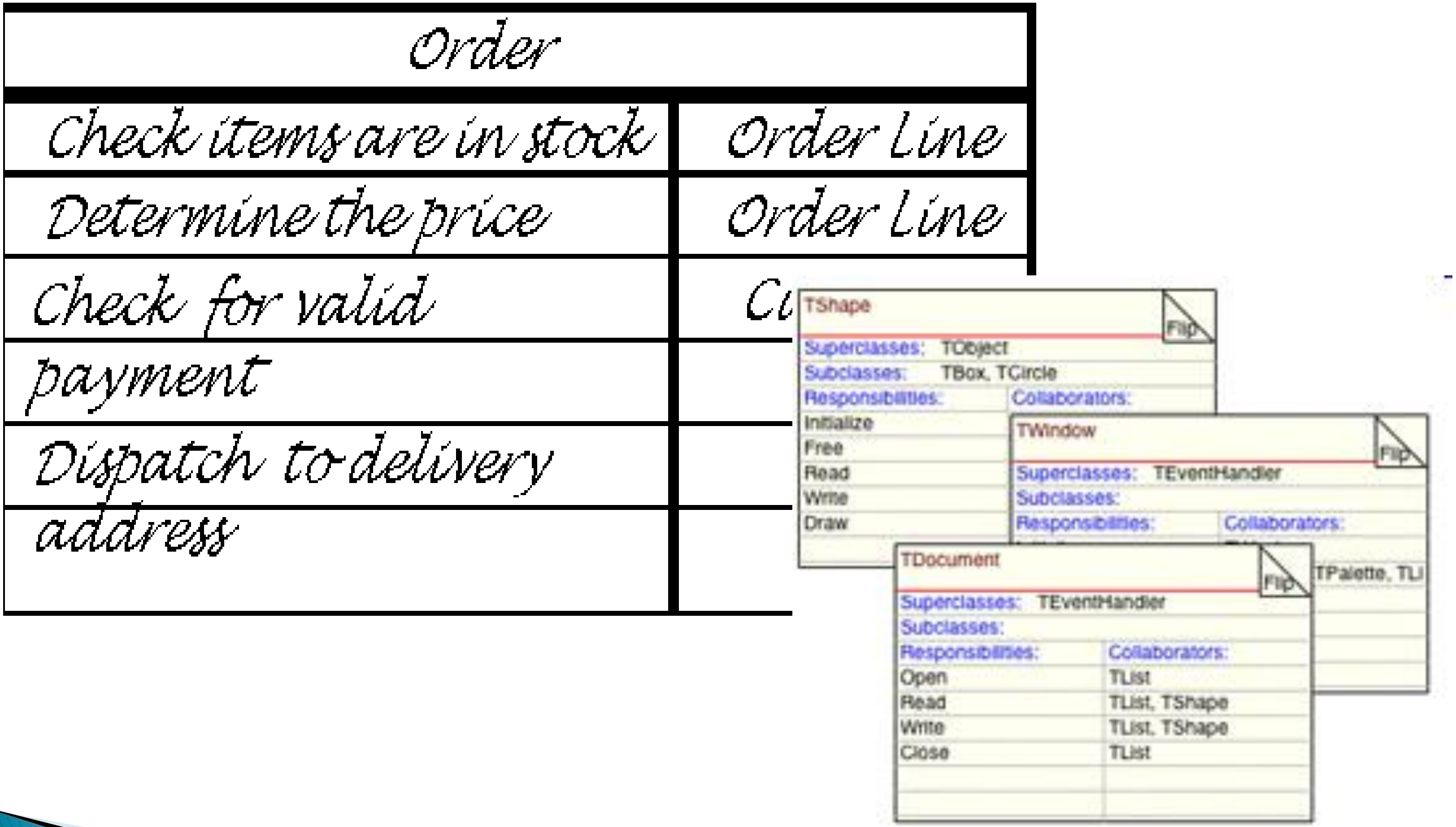
- In the traditional paradigm, there would have to be three different versions of setTitle, one for each type of painting
  - set_masterpiece_title
  - set_masterwork_title
  - set_other_painting_title

- That is because the traditional paradigm does not support inheritance

Allocation of operations setTitle and getTitle



CRC Cards



Order	
Check items are in stock	Order Line
Determine the price	Order Line
Check for valid payment	Cu
Dispatch to delivery address	

TShape

Superclasses: TObject

Subclasses: TBox, TCircle

Responsibilities: Initialize, Free, Read, Write, Draw

Collaborators:

TWindow

Superclasses: TEventHandler

Subclasses:

Responsibilities:

Collaborators: TPalette, TLi

TDocument

Superclasses: TEventHandler

Subclasses:

Responsibilities:	Collaborators:
Open	TList
Read	TList, TShape
Write	TList, TShape
Close	TList

CRC Cards

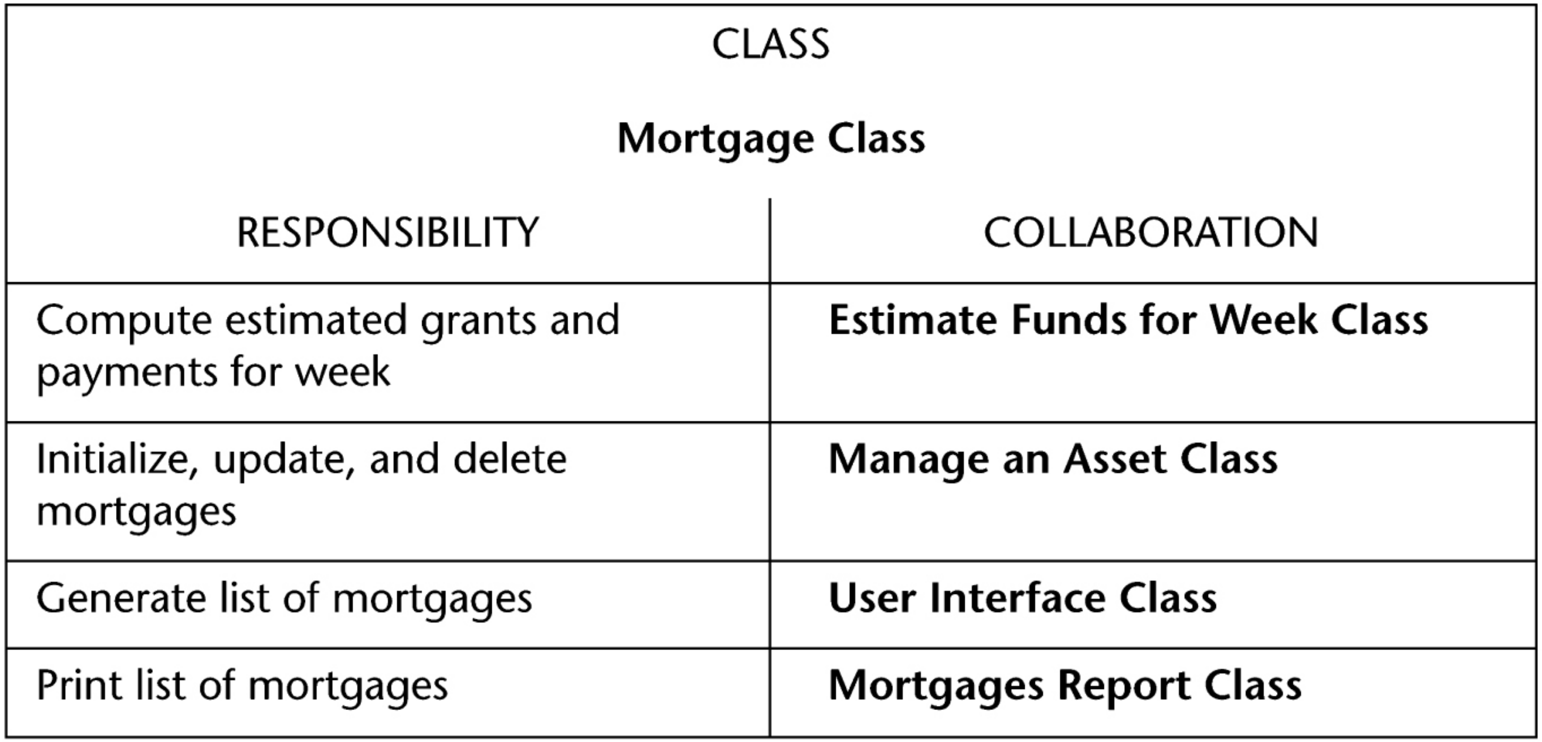
- Since 1990, class-responsibility-collaboration (CRC) cards have been utilized for object-oriented analysis and design
- Kent Beck and Ward Cunningham want to teach Smalltalk
- For each class, fill in a card showing
  - The name of the class;
  - The functionality of that class (its *responsibility*); and
  - The other classes it invokes to achieve that functionality (its *collaboration*)

CRC Cards (contd)

- Crowd around a table, pick a card representing a class, check use case it participates
- CRC cards are an aggressively informal technique, they also encourage discussion
- Get high level behavior

CRC Cards (contd)

- Example:



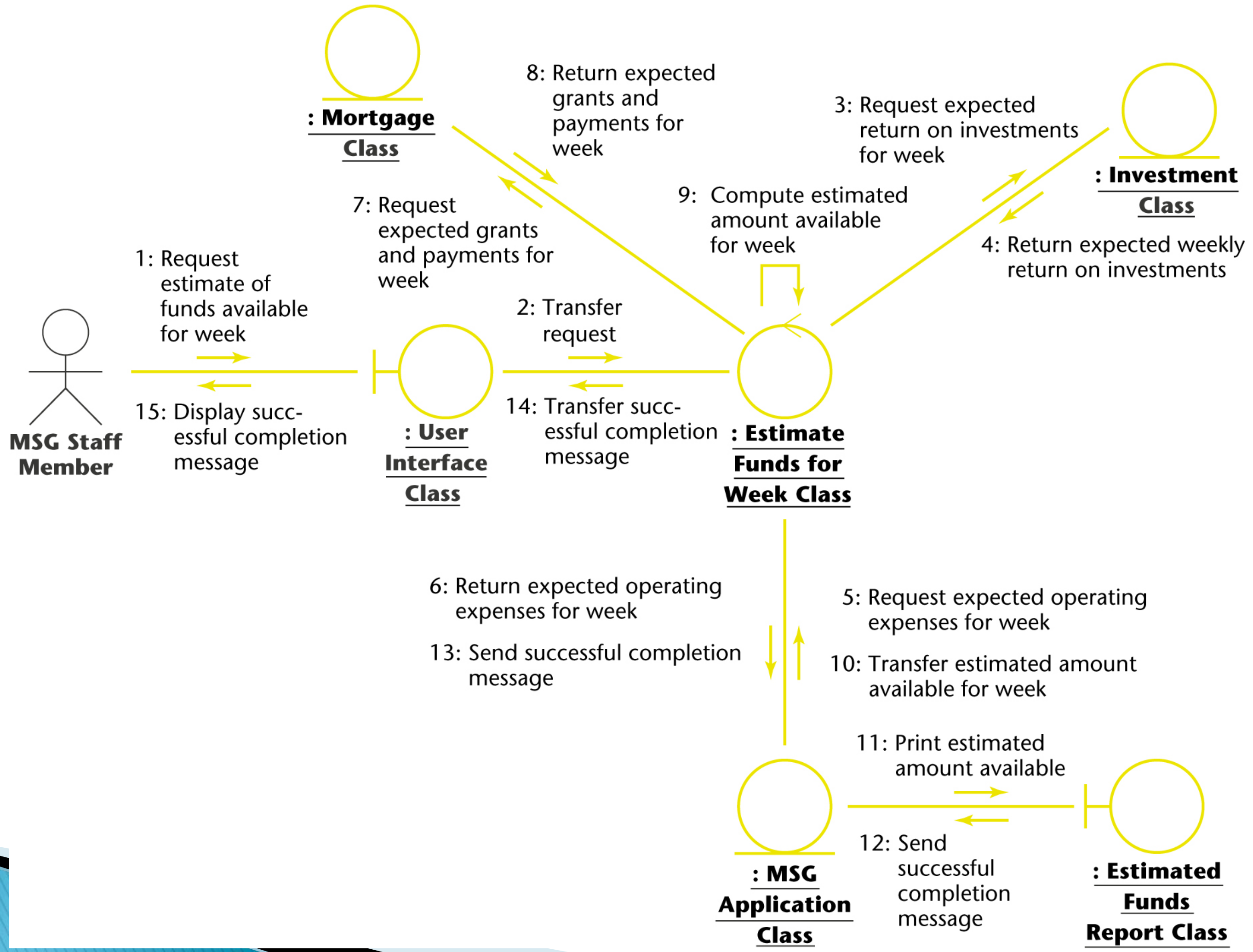
CLASS **Mortgage Class**	
RESPONSIBILITY	COLLABORATION
Compute estimated grants and payments for week	**Estimate Funds for Week Class**
Initialize, update, and delete mortgages	**Manage an Asset Class**
Generate list of mortgages	**User Interface Class**
Print list of mortgages	**Mortgages Report Class**

CRC Cards (contd)

- The data for the CRC card are obtained from the realizations of all the use cases

CRC Cards (contd)

- Consider the MSG collaboration diagram


: Mortgage Class
8: Return expected grants and payments for week
3: Request expected return on investments for week
: Investment Class
7: Request expected grants and payments for week
9: Compute estimated amount available for week
4: Return expected weekly return on investments
1: Request estimate of funds available for week
2: Transfer request
MSG Staff Member
15: Display successful completion message
: User Interface Class
14: Transfer successful completion message
: Estimate Funds for Week Class
6: Return expected operating expenses for week
13: Send successful completion message
5: Request expected operating expenses for week
10: Transfer estimated amount available for week
11: Print estimated amount available
12: Send successful completion message
: MSG Application Class
: Estimated Funds Report Class

CRC Cards (contd)

- The message
  - 7: Request estimated grants and payments for week

  is passed from **: Estimate Funds for Week Class** to **: Mortgage Class**, followed by the message
  - 8: Return estimated grants and payments for week

  in the reverse direction

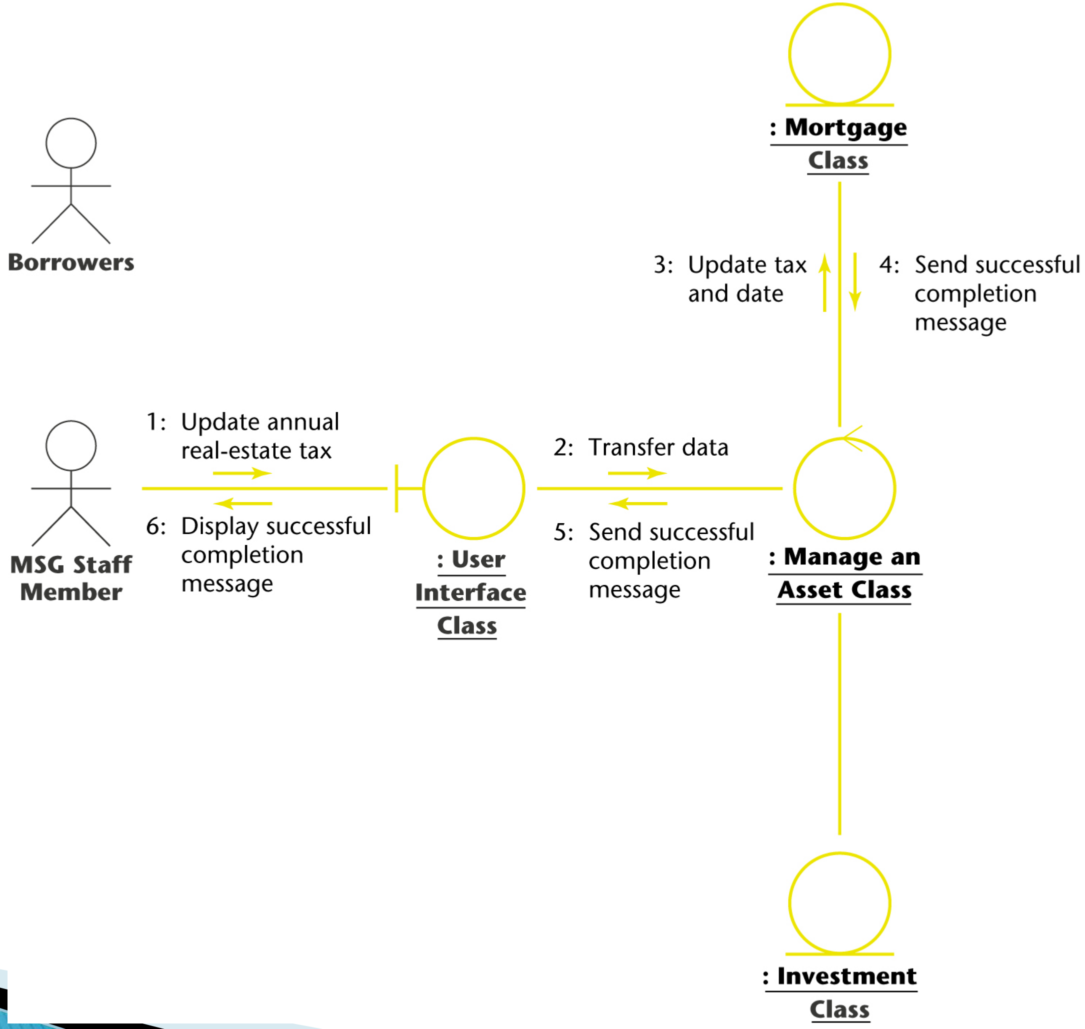
CRC Cards (contd)

- **Mortgage Class** therefore has a responsibility to
  - Compute estimated grants and payments for week

- In order to do this, it must collaborate with **Estimate Funds for Week Class**
  - This is shown in the first entry in the CRC card

CRC Cards (contd)

- Consider the MSG collaboration diagram



Borrowers
: Mortgage Class
3: Update tax and date
4: Send successful completion message
1: Update annual real-estate tax
6: Display successful completion message
MSG Staff Member
: User Interface Class
2: Transfer data
5: Send successful completion message
: Manage an Asset Class
: Investment Class

CRC Cards (contd)

- The message
  - 3: Update tax and date

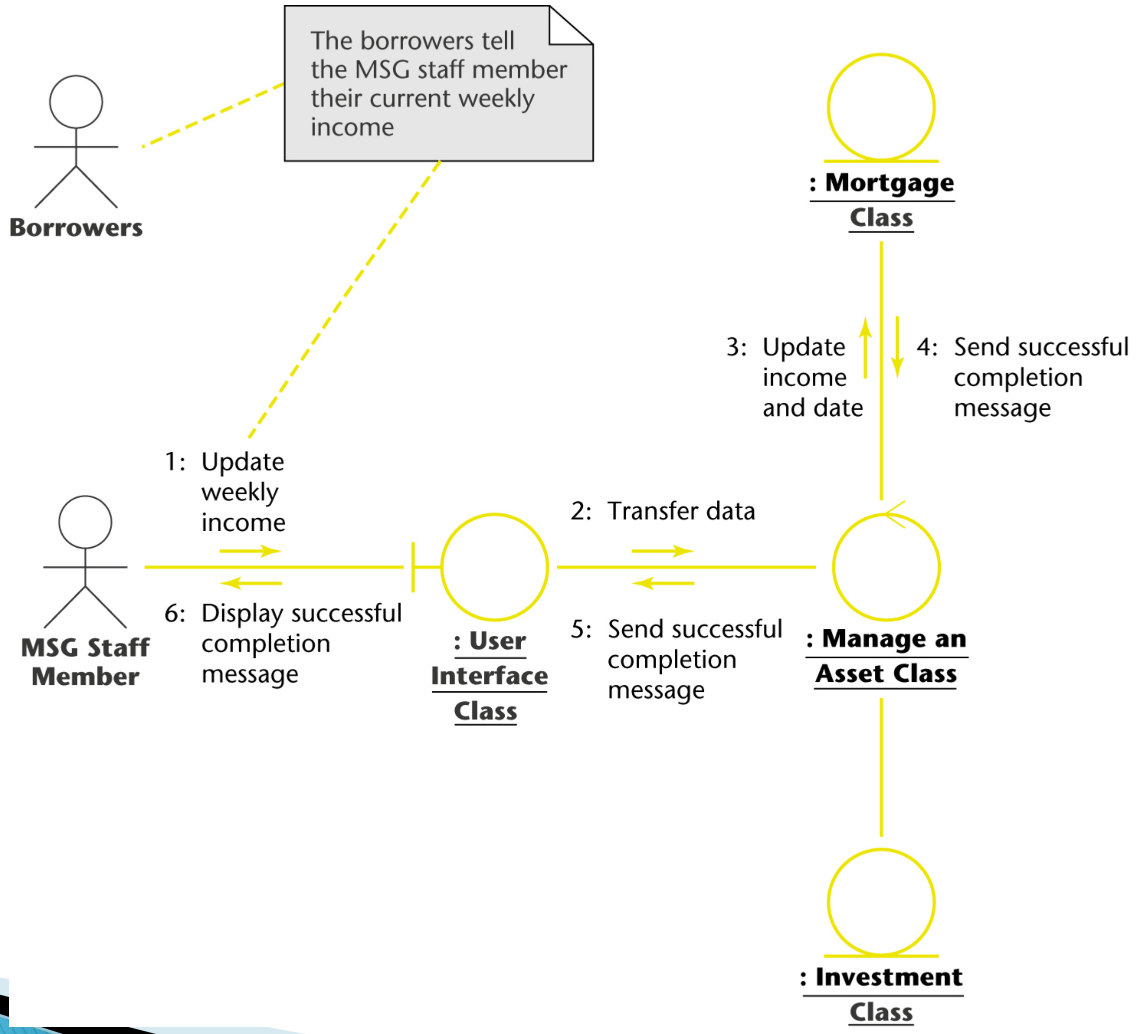
  is passed from **: Manage an Asset Class** to **: Mortgage Class**

- Then, the message
  - 4: Send successful completion message

  is passed back
- This is the second entry in the CRC card

CRC Cards (contd)

- Consider the MSG collaboration diagram



The borrowers tell the MSG staff member their current weekly income
Borrowers
: Mortgage Class
3: Update income and date
4: Send successful completion message
1: Update weekly income
2: Transfer data
MSG Staff Member
6: Display successful completion message
: User Interface Class
5: Send successful completion message
: Manage an Asset Class
: Investment Class

CRC Cards (contd)

- The message
  - 3: Update income and date

  is passed from **: Manage an Asset Class** to **: Mortgage Class**

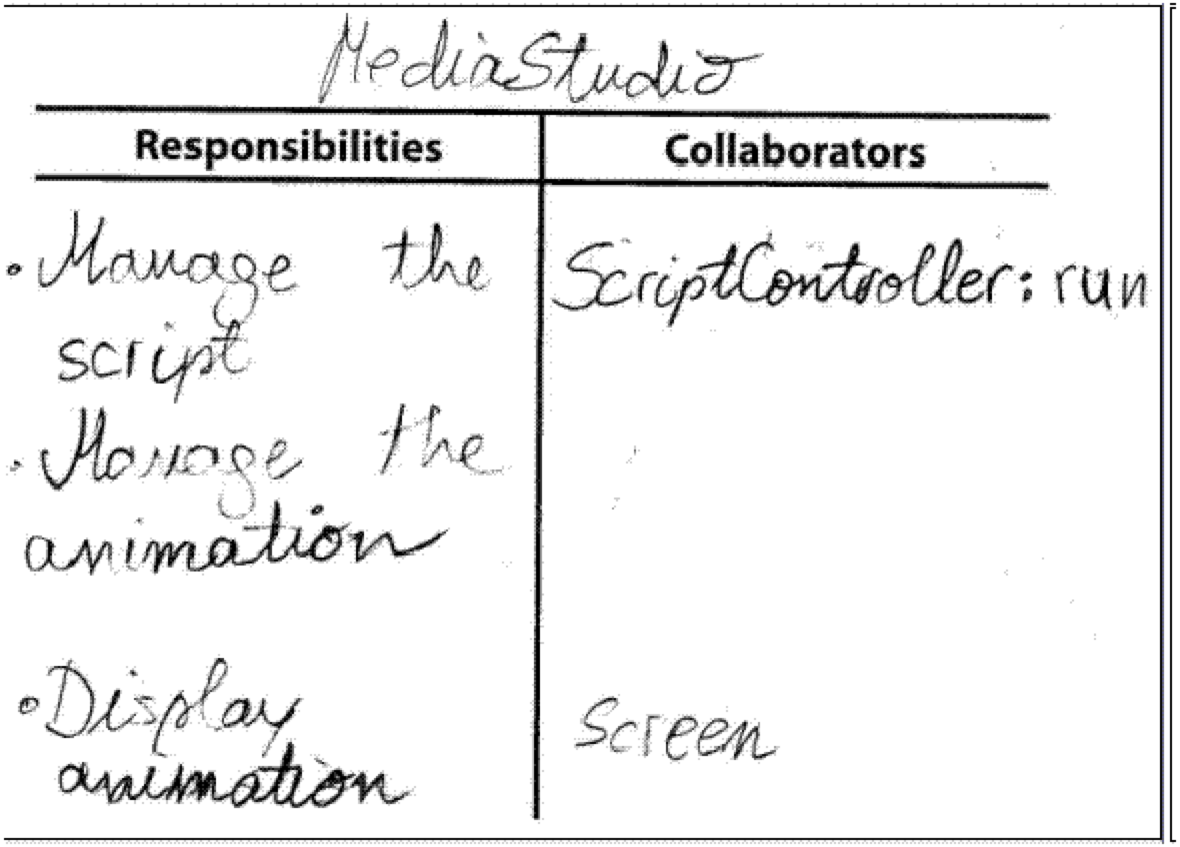
- The message
  - 4: Send successful completion message

  is then passed back

CRC Cards (contd)

- Attribute combinedWeeklyIncome is one of the eight attributes of **Mortgage Class** that might need to be changed by : **Manage an Asset Class**
  - This responsibility is already included in responsibility Initialize, update, and delete mortgages

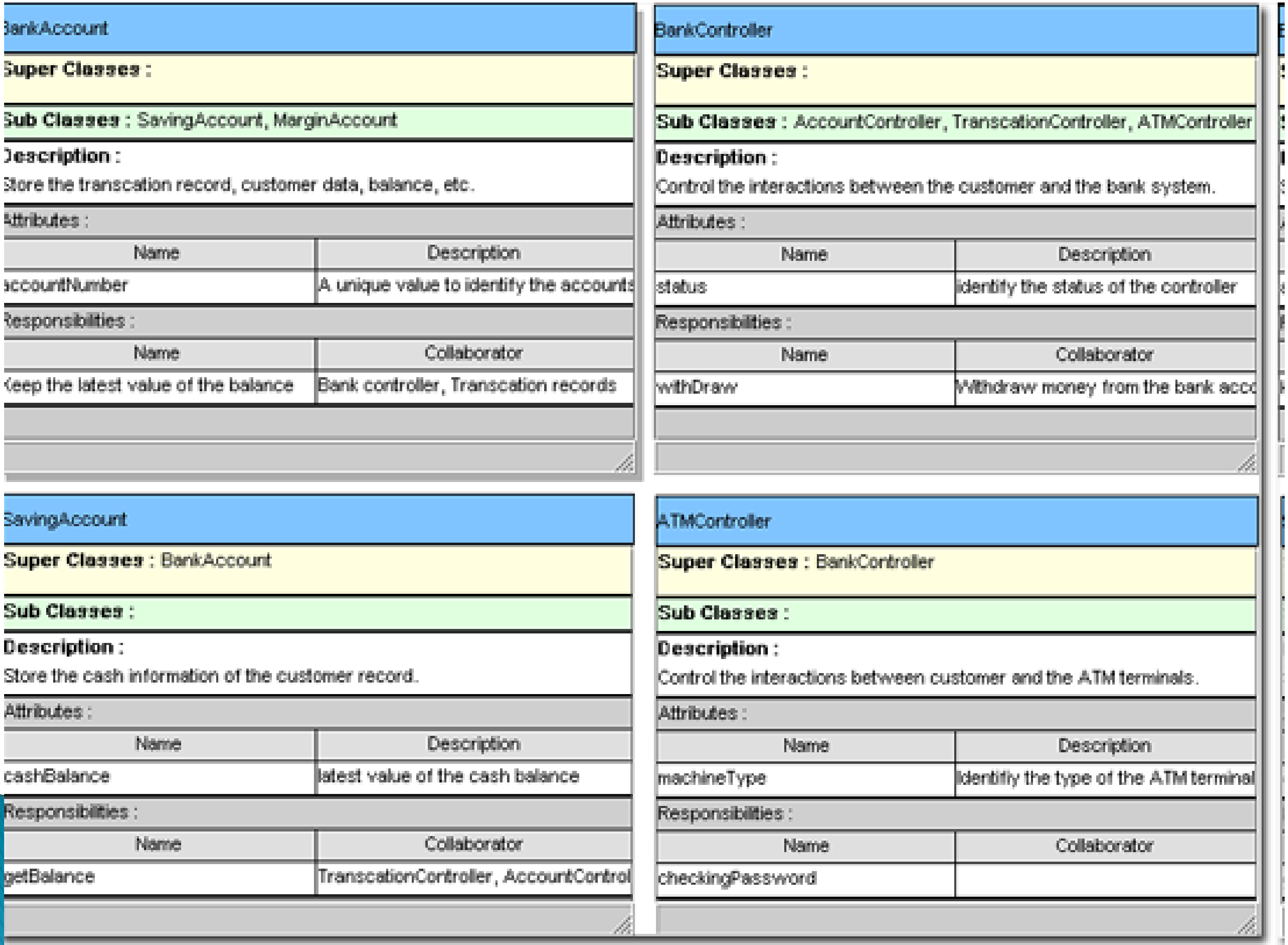
What is wrong ?



MediaStudio

Responsibilities	Collaborators
• Manage the script	ScriptController: run
• Manage the animation	
• Display animation	Screen

CRC cards have been extended



BankAccount

Super Classes :

Sub Classes : SavingAccount, MarginAccount

Description :
Store the transcation record, customer data, balance, etc.

Attributes :

Name	Description
accountNumber	A unique value to identify the accounts

Responsibilities :

Name	Collaborator
Keep the latest value of the balance	Bank controller, Transcation records

BankController

Super Classes :

Sub Classes : AccountController, TranscationController, ATMController

Description :
Control the interactions between the customer and the bank system.

Attributes :

Name	Description
status	Identify the status of the controller

Responsibilities :

Name	Collaborator
withDraw	Withdraw money from the bank acco

SavingAccount

Super Classes : BankAccount

Sub Classes :

Description :
Store the cash information of the customer record.

Attributes :

Name	Description
cashBalance	latest value of the cash balance

Responsibilities :

Name	Collaborator
getBalance	TranscationController, AccountControl

ATMController

Super Classes : BankController

Sub Classes :

Description :
Control the interactions between customer and the ATM terminals.

Attributes :

Name	Description
machineType	Identify the type of the ATM terminal

Responsibilities :

Name	Collaborator
checkingPassword	

CRC Cards (contd)

- When used by a team, CRC cards can highlight missing or incorrect attributes or operations

- Having the members of the team act out the responsibilities of each CRC card is an effective means of verifying that the classes are correct

Exercises

Using CRC technique to identify the responsibilities and collaborators of 5 classes in the Osbert Olesby Case Study