Lecture 10 - OO Software Architectures
- Class Design

Reference: H. Gomaa, Chapters 14 - *Software Modeling and Design*, Cambridge University Press, February 2011

1

Design Class Operations

- Design Class Operations from dynamic interaction model
  - Shows direction of message from sender object to receiver object
- Design Class Operations from Static Model
  - May be used for entity classes
  - Standard operations
    - Create, Read, Update, Delete
  - Specific operations
    - Based on functionality provided by class

2

Information Hiding Class Structuring

- Design of Information Hiding Classes
  - Entity classes are categorized further
    - Data abstraction classes
    - Database wrapper classes

Data Abstraction Class

- Data Abstraction Class is an entity class
- Encapsulates data structure
  - Hides internal structure and content of data structure
  - Attributes provided by static model (class diagram)
- Design Class interface
  - Data accessed indirectly via operations
  - Consider functionality required by client objects that interact with data abstraction object
  - Consider interaction model

Design Class Operations

Figure 14.1 Example of data abstraction class (Fig. 14.1b – after design of operations)

Figure 14.1a Analysis model – communication diagram

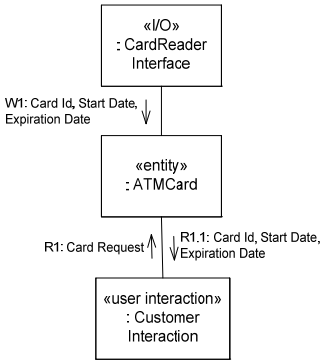
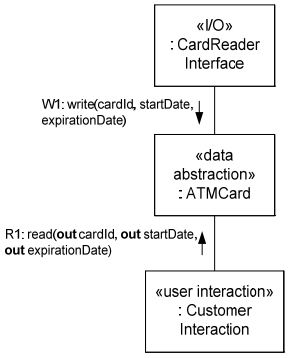


Figure 14.1b Design model – communication diagram



- Design model
  - Use synchronous message notation to show design of operations

5

Figure 14.1c Design model – class diagram

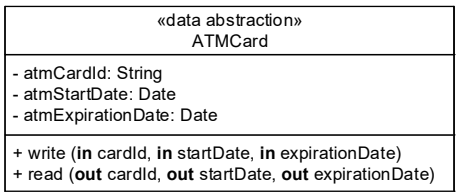


Figure 14.2 Example of data abstraction class

Figure 14.2a Analysis model – communication diagram

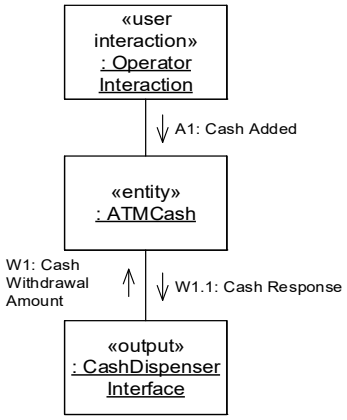


Figure 14.2b Design model – communication diagram

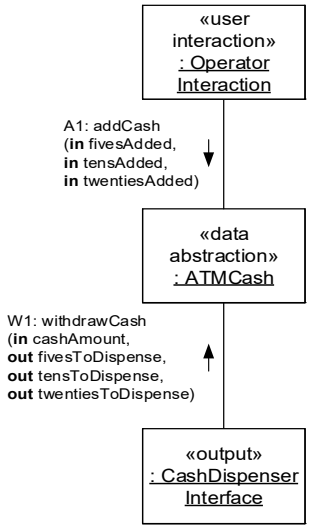
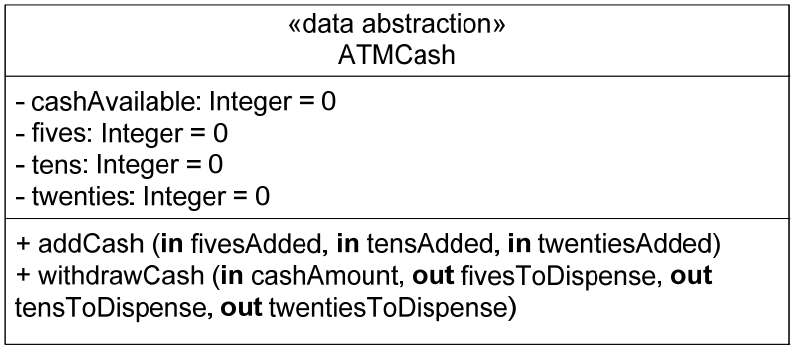


Figure 14.2c Design model – class diagram

Figure 14.2 Example of data abstraction class

Figure 14.2c Design model - class diagram

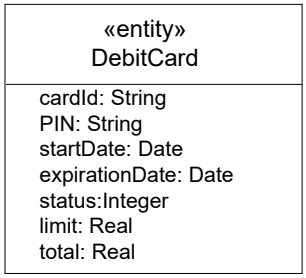


Database Wrapper Class

- Entity class in Analysis Model
  - Encapsulated data stored in database
- Analysis Model class mapped to
  - Database Wrapper Class
    - Hides interface to database (e.g., relational)
  - Attributes of class mapped to
    - Relation (flat file) stored in database
- Database Wrapper Class
  - Provides OO interface to database
  - Hides details of how to access data in database
    - Hides SQL statements

Example of database wrapper class

Analysis model

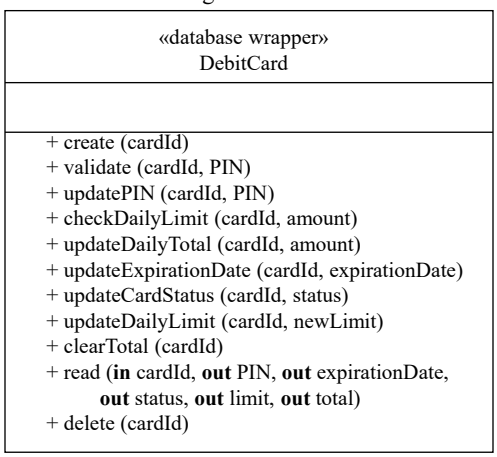


Relation in relational database :
- DebitCard (cardId, PIN, startDate, expirationDate, status, limit, total)

(underline = primary key)

Example of database wrapper class

Design model



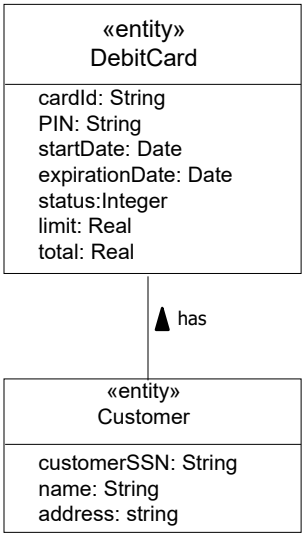
Relation in relational database :

- DebitCard (cardId, PIN, startDate, expirationDate, status, limit, total)

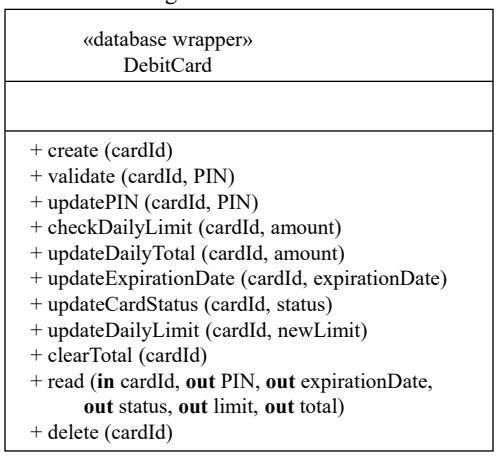
(underline = primary key)

Example of database wrapper class

Analysis model



Design model



Relation in relational database :

- DebitCard (cardId, PIN, startDate, expirationDate, status, limit, total, *customerSSN*)
- Customer (customerSSN, name, address, *cardId)*

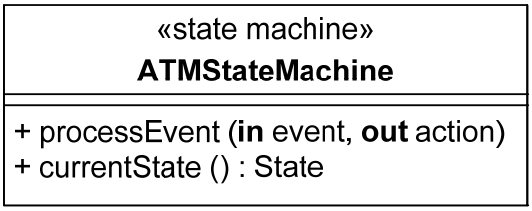
(underline = primary key, italic = *foreign key*)

State Machine Class

- Hides contents of statechart / state transition table
  - Maintains current state of object
- Process Event Operation
  - Called to process input event
  - Depending on current state and conditions
    - Might change state of object
    - Might return action(s) to be performed
- Current State Operation
  - Returns the state stored in state transition table
- If state transition table changes
  - Only this class is impacted

13

Figure 14.3 Example of State Machine class



Business Logic Class

- Hides business application logic
  - Encapsulate business rules
- Business rules could change
  - Independently of other business logic classes
  - Independently of entity classes
- E.g., Bank Withdrawal Transaction Manager business rules
  - Account must have positive (or zero) balance after withdrawal
  - Maximum daily withdrawal limit is $300

Figure 14.5: Example of business logic class

Figure 14.5a: Analysis model - communication diagram

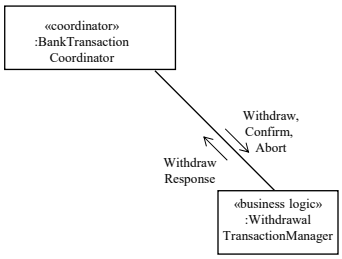


Figure 14.5b: Design model - communication diagram

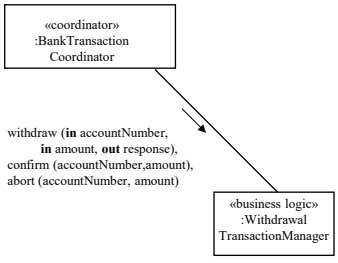
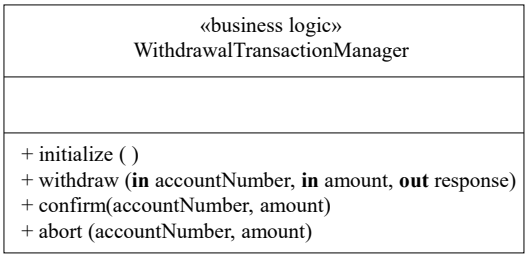


Figure 14.5: Example of business logic class

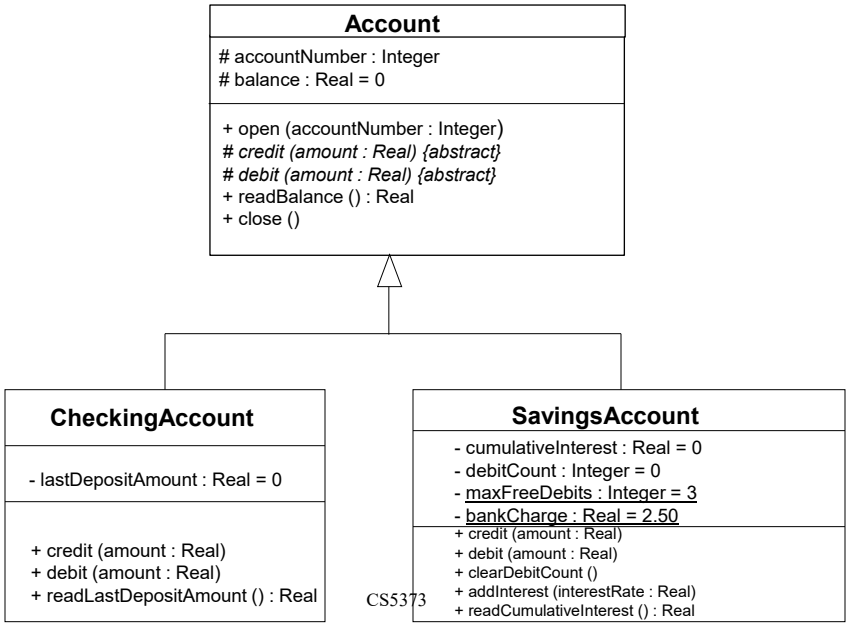
Figure 14.5c: Design model - class diagram



Inheritance in Design

- Subclass inherits generalized properties from superclass
  - Property is Attribute or Operation
- Inheritance
  - Allows sharing of properties between classes
  - Allows adaptation of parent class (superclass) to form child class (subclass)
- Subclass inherits attributes & operations from superclass
  - May add attributes
  - May add operations
  - May redefine operations

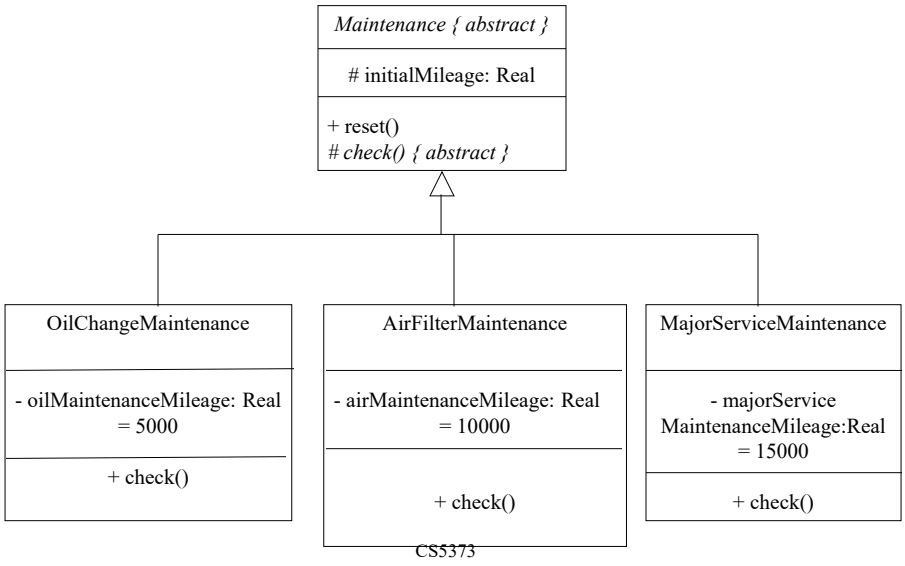
Figure 14.7: Example of superclass and subclass



Abstract Class

- Abstract Class
  - Template for creating subclasses
  - Has no instances
  - Only used as superclass
  - Defines common interface for subclasses
- Abstract operation
  - Operation declared in abstract class but not implemented
- Abstract Class defers implementation of some or all of its operations to subclasses
- Different subclasses can define different implementations of same abstract operation

Example of abstract superclass and subclasses



Class Interface Specification

- Information hidden by class
- Class structuring criterion
- Assumptions made in specifying class
  - E.g., one operation needs to be called before another
- Anticipated changes
  - Encourage “design for change”
- Superclass (if applicable)
- Inherited Operations (if applicable)
- Operations provided by class

Class Interface Specification

- Operations provided by class. For each operation:
  - Function performed
  - Precondition
    - Condition that must be true when operation is invoked
  - Postcondition
    - Condition that must be true at completion of operation
  - Invariant
    - Condition that must be true at all times during execution
  - Input parameters
  - Output parameters
  - Operations used by class (provided by other classes)

Example of Class Interface Specification (pages 245-246)

Information Hiding Class: `Checking Account`

Information Hidden: Encapsulates checking account attributes and their current values, in particular

`balance` and `lastDepositAmount`

Class structuring criterion: Data abstraction class.

Assumptions: Checking accounts do not have interest.

Anticipated changes: Checking accounts may be allowed to earn interest.

Superclass: Account

Inherited operations: open, credit, debit, readBalance, close

Operations provided:

Example of Class Interface Specification -Operations Provided

(pages 245-246)

Information Hiding Class: `Checking Account`

Operations provided:

1) `credit (in amount : Real)`

Function: Adds `amount` **to** `balance`**. Sets** `lastDepositAmount` **equal to** `amount`**.**

Precondition: Account has been created.

Postcondition: Checking account has been credited.

Input parameters: `amount` **– funds to be added to account**

Operations used: None

2) `debit (in amount : Real)`

Function: Deducts `amount` **from** `balance`**.**

Precondition: Account has been created.

Postcondition: Checking account has been debited.

Input parameters: `amount` **– funds to be deducted from account**

Output parameters: None

Operations used: None

3) `readLastDepositAmount () : Real`

Function: Returns the amount last deposited into the account.

Precondition: Account exists.

Invariant: Values of account attributes remains unchanged.

Postcondition: Amount last deposited into the account has been read.

Input parameter: None

Output parameters: Amount last deposited into the account

Operations used: None