CRC Modeling

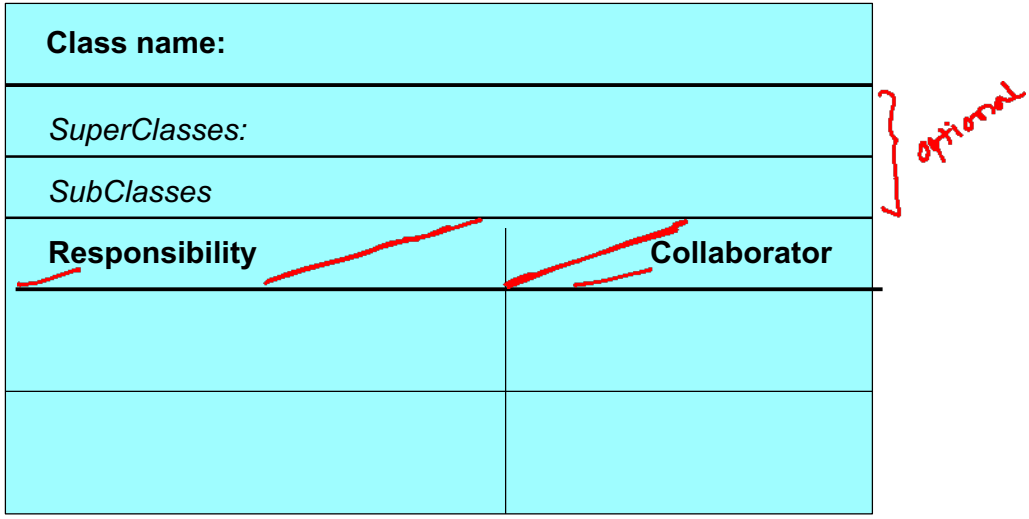
Class Responsibility Collaborator (CRC)

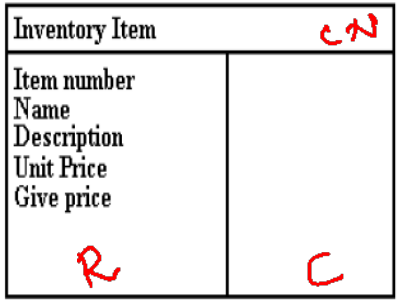
- CRC (class responsibility collaborator) modeling process for identifying user requirements.
- CRC modeling is an effective, low-tech method for developers and users to work closely together to identify and understand business requirements.
- A CRC card is an index card that is used to represent the responsibilities of classes and the interaction between the classes.
- CRC cards are an informal approach to object oriented modeling.
- The cards are created from use-case scenarios, based on the system requirements.

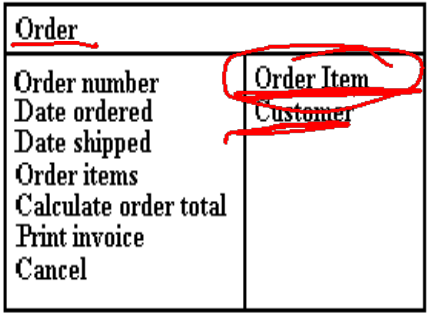
The CRC card Session

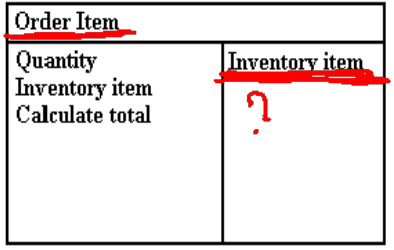
- Groups consisting of five or six people.
- Each group typically consists of developers, domain experts and an OO technology facilitator.

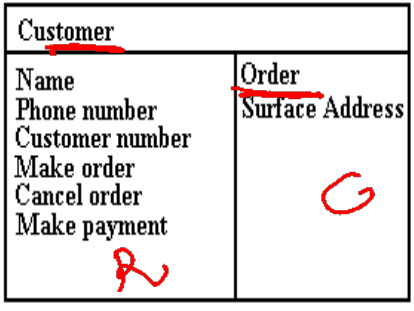
A CRC card

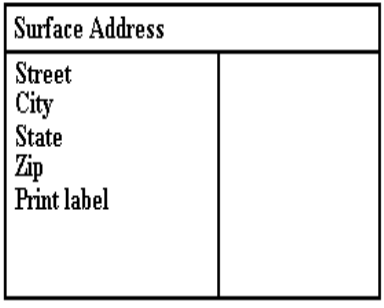

Class name:
SuperClasses:
SubClasses
Responsibility
Collaborator
optional


Inventory Item
Item number
Name
Description
Unit Price
Give price


Order
Order number
Date ordered
Date shipped
Order items
Calculate order total
Print invoice
Cancel
Order Item
Customer


Order Item
Quantity
Inventory item
Calculate total
Inventory item


Customer
Name
Phone number
Customer number
Make order
Cancel order
Make payment
Order
Surface Address


Surface Address
Street
City
State
Zip
Print label

Class

- A class represents a collection of similar objects. An object is a person, place, thing, event, concept, screen, or report that is relevant to the system at hand.
- For example, a shipping/inventory control system with the classes such as **Inventory Item**, **Order**, **Order Item**, **Customer**, and **Surface Address**.
- The name of the class appears across the top of the card.

Responsibility

- A responsibility is anything that a class knows or does. For example, customers have names, customer numbers, and phone numbers.
- These are the things that a customer knows. Customers also order products, cancel orders, and make payments.
- These are the things that a customer does.
- The things that a class knows and does constitute its responsibilities.
- Responsibilities are shown on the left hand column of a CRC card.

Collaborator

- Sometimes a class will have a responsibility to fulfill, but will not have enough information to do it.
- When this happens it has to collaborate with other classes to get the job done.
- For example, an **Order** object has the responsibility to calculate it's total. Although it knows about the **Order Item** objects that are a part of the order, it doesn't know how many items were ordered (**Order Item** knows this) nor does it know the price of the item (**Inventory Item** knows this). To calculate the order total, the **Order** object collaborates with each **Order Item** object to calculate its own total, and then adds up all the totals to calculate the overall total.
- For each **Order Item** to calculate its individual total, it has to collaborate with **Inventory Item** to determine the cost of the ordered item, multiplying it by the number ordered (which it does know). The collaborators of a class are shown in the right-hand column of a CRC card.

CRC Models

- A CRC model is a collection of CRC cards that represent whole or part of an application or problem domain.
- The most common use for CRC models, the one that this white paper addresses, is to gather and define the user requirements for an object-oriented application.
- Example CRC model for a shipping/inventory control system, showing the CRC cards as they would be placed on a desk or work table.
- Note the placement of the cards: Cards that collaborate with one another are close to each other, cards that don't collaborate are not near each other.
- CRC models are created by groups of business domain experts, led by a CRC facilitator who is assisted by one or two scribes.
- The CRC facilitator is responsible for planning and running the CRC modeling session.

CRC Cards

- **Step 1**: Identify the classes in the problem domain.
  - Use the problem statement or requirements document to find all of the nouns and verbs in the problem statement.
  - The nouns represent the object/classes in the system; the verbs may show what their responsibilities are.

CRC Cards

- **Step 2**: Take at least one card per person . Each person should be responsible for at least one class.
- **Step 3**: Add the class name and add responsibilities that are obvious from the requirements. Attributes typically are not added at this time. Add super or subclasses that are obvious.

CRC Cards

- **Step 4**: Walk-through a scenario that represents an important system function in the requirements document.

  Decide which class (es) is responsible for this function. The owner of the class then picks up her card and announces that she needs to fulfill this responsibility.

  The responsibility may be refined into smaller tasks if possible.

  These smaller tasks can be fulfilled by the same object or they can be fulfilled be interacting with other objects.

  If no appropriate class (to fulfill this responsibility) exists, you may need to make a class.

Class: Book

Responsibilities	Collaborators
knows whether on loan	
knows due date	
knows its title	
knows its author(s)	
knows its registration code	
knows if late	Date
check out	

Class: Librarian

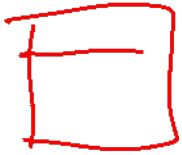
Responsibilities	Collaborators
check in book	Book
check out book	Book, Borrower
search for book	Book
knows all books	
search for borrower	Borrower
knows all borrowers	

Class: Borrower

Responsibilities	Collaborators
knows its name	
keeps track of borrowed items	
keeps track of overdue fines	

Class: Date

Responsibilities	Collaborators
knows current date	
can compare two dates	
can compute new dates	



Example-Vending Machine

Let us take an example application and examine the process of identifying the objects/classes and describe their responsibilities using CRC cards.

The example application is a Vending Machine that allows users to buy snack items. In addition, a user can find out the caloric content of her choice.

Example-Vending Machine

Specification :

A Vending machine holds a number of snack items and displays the list of snack items and their prices through an user interface with a display screen and buttons for making selections. In addition, the vending machine has a receptacle for money and an item dispenser.

A user can make a selection and query for the number of calories of a snack item. The calories are displayed on pressing a button. A user can place the money in the receptacle and select an item.

Example-Vending Machine

- Let us select the nouns and the verbs in the specification.
- Nouns are in blue and the verbs are in green.
- *A **Vending machine** **holds** a number of **snack items** and **displays** the **list of snack items** and their **prices** through an **user interface** with a **display screen** and **buttons** for **making selections**. In addition, the vending machine has a **receptacle** for **money** and an **item dispenser**.*
- *A **user** can **make a selection** and **query** for the number of **calories** of a snack item. The calories are **displayed** on **pressing** **a** **button**. A user can **place** the money in the **receptacle** and select an item.*

Example-Vending Machine

- Most of the nouns are objects/classes.
- Some nouns are attributes of these classes.
- The verbs are actions that can be attached to these objects.
- In order to focus on the problem-domain objects, let us separate the object/classes into presentation-specific (user-interface related) and problem-specific classes.
- We will then select two problem specific classes and write the CRC cards for them.

Example-Vending Machine

Problem-specific classes:

- Vending Machine
- Snack item
- *Price*
- *Calories*
- Selection
- User

Presentation-specific classes:

- Display screen
- Selection Buttons
- Item Dispenser
- Money receptacle

A CRC card for class SnackItem

Class name: SnackItem	
Responsibility	**Collaborator**
Knows its price and calories	

A CRC card for the Vending Machine

Class name: VendingMachine	
Responsibility	**Collaborator**
Maintains a collection of SnackItems. Allows addition and removal of SnackItems	SnackItem

Showing the Collaboration of a VendingMachine and SnackItem

Exercise

Problem Statement

An ***Automated Teller Machine (ATM)*** allows **bank customers** to perform a number of financial transactions: to **withdraw** and **deposit** funds to an account, query the balance of any account. The ATM offers an user interface with a display screen, keypad, cash dispenser, deposit slot and a card reader.

Once a customer's card is verified, the customer can **query to see the balance** in all her account (s), **deposit**, **withdraw** or **transfer** money from one account into another.

Using Object-Oriented analysis and design techniques, build an object model for the ATM machine.

Exercise

- Identify the classes from the problem statement and describe their responsibilities using CRC cards.