CRC

CSC207 Winter 2015

Computer Science
UNIVERSITY OF TORONTO

CRC Cards

- A tool and method for systems analysis and design.
- Part of the Object-Oriented development paradigm.
- Highly interactive and human-intensive.
- Final result: definition of classes and their relationships.
- *What* rather than *How*.
- Benefits

  Cheap and quick: all you need is index cards

  Simple, easy methodology

  Forces you to be concise and clear

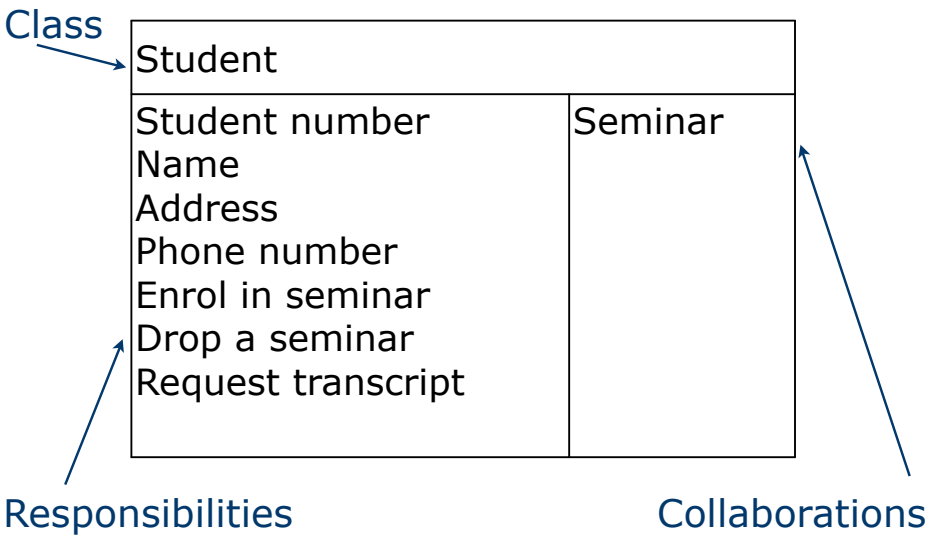
  Input from every team member

What is a CRC Card?

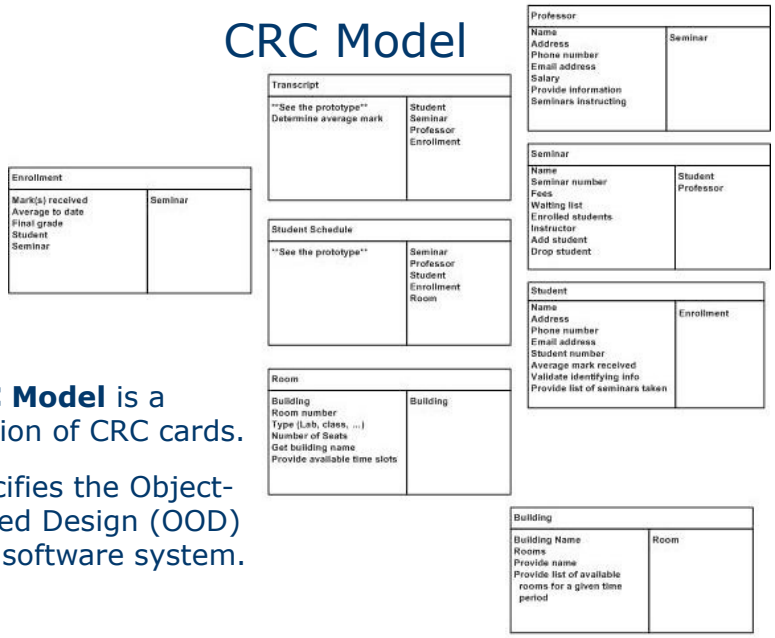
CRC stands for **C**lass, **R**esponsibility and **C**ollaboration.

- **Class**
  - An object-oriented class name.
  - Include information about super- and sub-classes.
- **Responsibility**
  - What information this class stores
  - What this class does
  - The behaviour for which an object is accountable.
- **Collaboration**
  - Relationship to other classes.
  - Which other classes this class uses

What does a CRC Card Look Like?



CRC Model



- A **CRC Model** is a collection of CRC cards.
- It specifies the Object-Oriented Design (OOD) of the software system.

How To Create a CRC Model?

Typically, you are given a description (in English) of the requirements for a software system.

You work in a team.

Ideally, you all gather around a table.

You need a set of index cards and some pens.

Coffee / other beverages are optional.

How to Create a CRC Model?

Read the description. Again. And again.

Identify core **classes** (simplistic advice: look for nouns).

Create a card per class (begin with class names only).

Add **responsibilities** (simplistic advice: look for verbs).

Which other classes does this class need to talk to to fulfil its responsibilities? Add **collaborators**.

Add more classes as you discover them.

Put classes away if they become unnecessary. (But don't tear them up yet!)

Refine by identifying abstract classes, inheritance, etc.

Keep adding/refining until everyone on the team is satisfied.

How Can We Tell It Works?

A neat technique: a **Scenario Walk-through**.

Select a scenario and choose a plausible set of inputs for it.

Manually "execute" each scenario.

- Start with initial input for scenario and find a class that has responsibility for responding to that input.
- Trace through the collaborations of each class that participates in satisfying that responsibility.
- Make adjustments as necessary.
- Repeat until scenario has "stabilized" (that is, no further adjustments are necessary).