Object Oriented Analysis and Design with Java

Prof.J.Ruby Dinakar
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

PES UNIVERSITY ONLINE

Object Oriented Analysis and Design with Java

Class Modelling: UML Class Diagrams and CRC

Prof. J Ruby Dinakar

Department of Computer Science and Engineering

Class Modeling

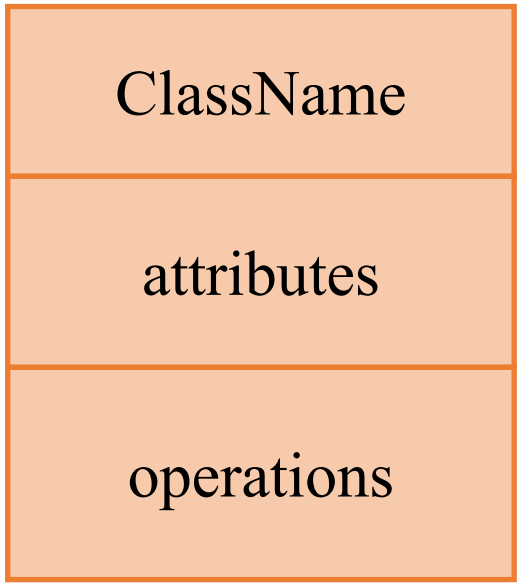
- A class model captures the static structure of a system by characterizing the objects in the system, the relationships between the objects and the attributes and operations for each class of objects.
- **Class diagram** is a graphical notation used to construct and visualize object oriented systems.
- Class diagram can be mapped directly with object oriented languages.
- Class diagrams capture the static structure of Object-Oriented systems as how they are structured rather than how they behave.
- It supports architectural design.
- They identify what classes are there, how they interrelate and how they interact.
- A UML class diagram is made up of:

  - A set of classes and
  - A set of relationships between classes

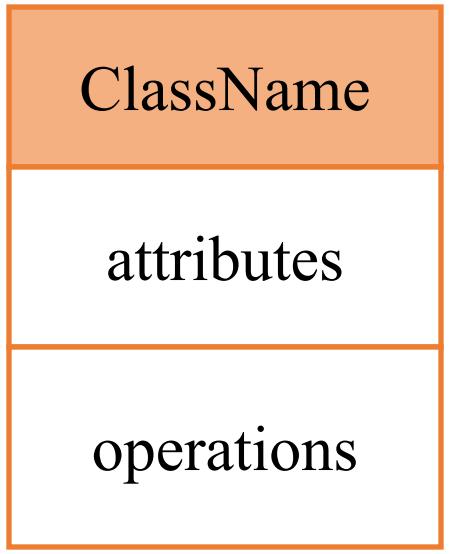
Class

A class is a description of a set of objects that share the same attributes, operations, relationships and semantics.

Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations in separate, designated compartments.

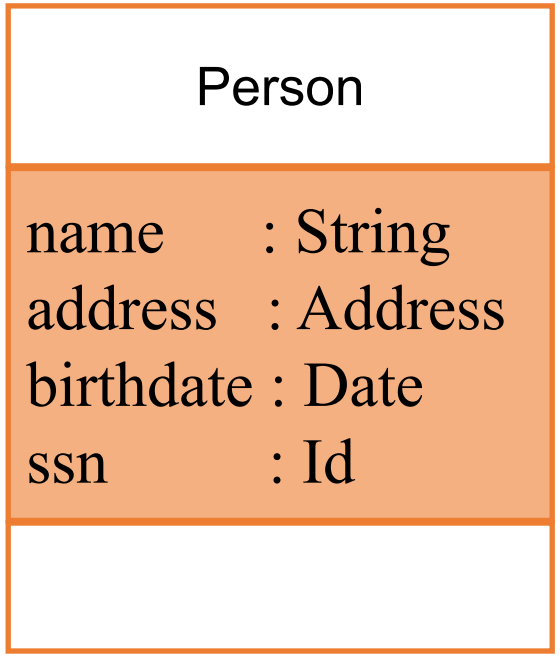


Class



The name of the class is the only required tag in the graphical representation of a class. It always appears in the top-most compartment.

Attributes



Each class can have attributes.

An *attribute* is a named property of a class that describes the object being modeled.

Each attribute has a type.

In the class diagram, attributes appear in the second compartment just below the name-compartment.

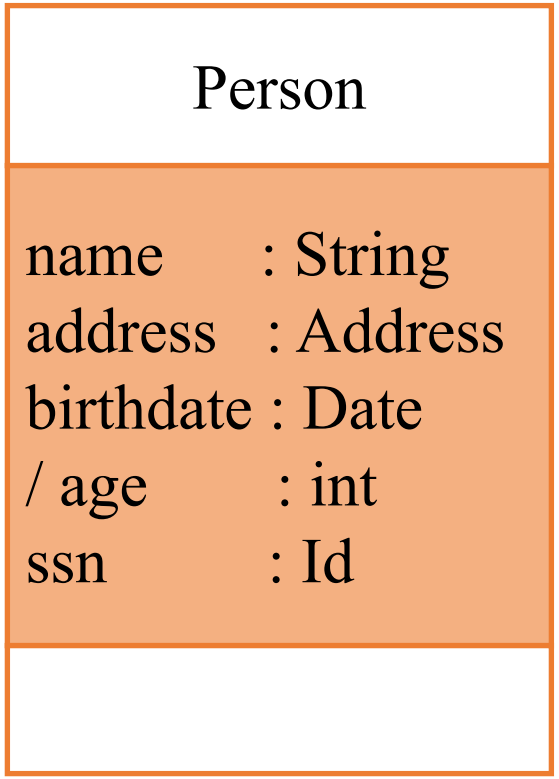
Derived Attribute

Attributes are usually listed in the form:

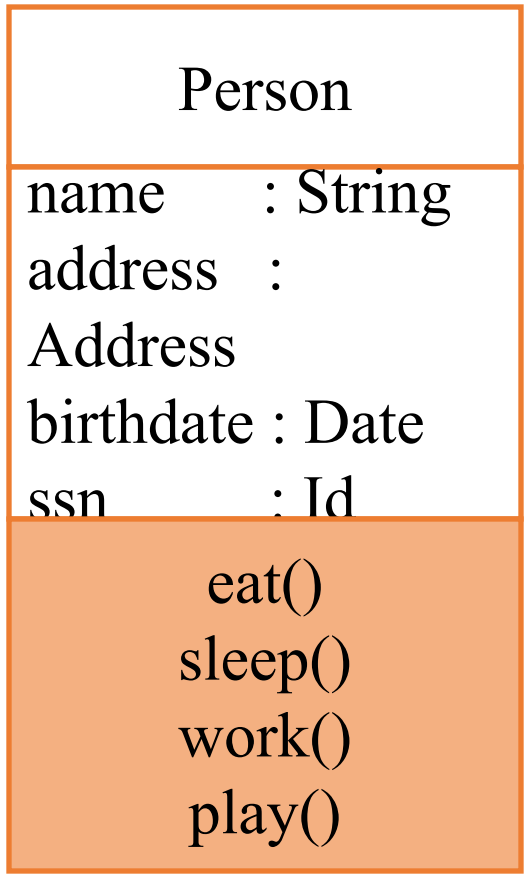
attributeName : Type

A *derived* attribute is one that can be computed from other attributes, but doesn't actually exist.
For example,
a Person's age can be computed from his birth date.
A derived attribute is
designated by a preceding '/' as in:

/ age : int

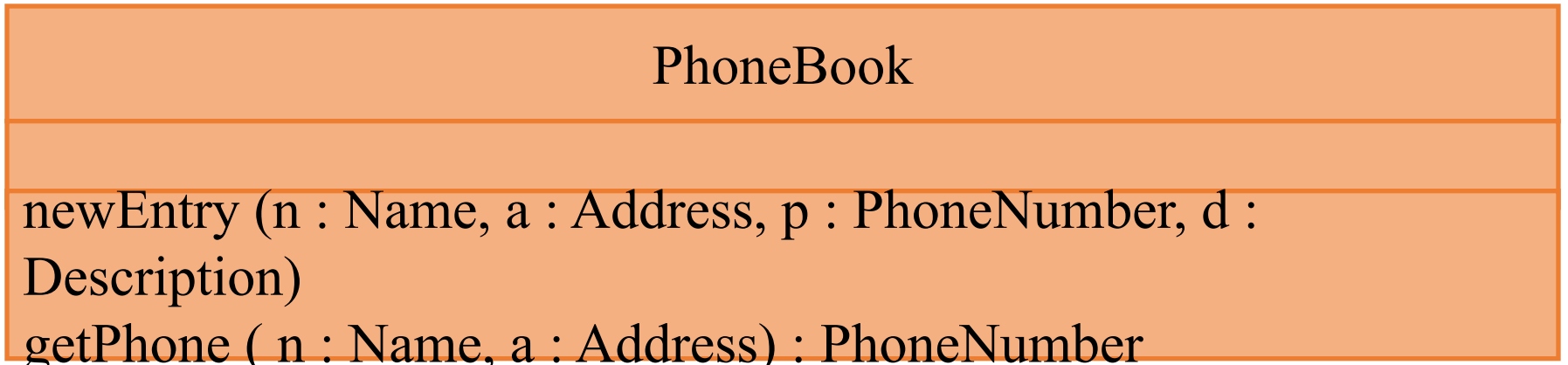


Class Operations



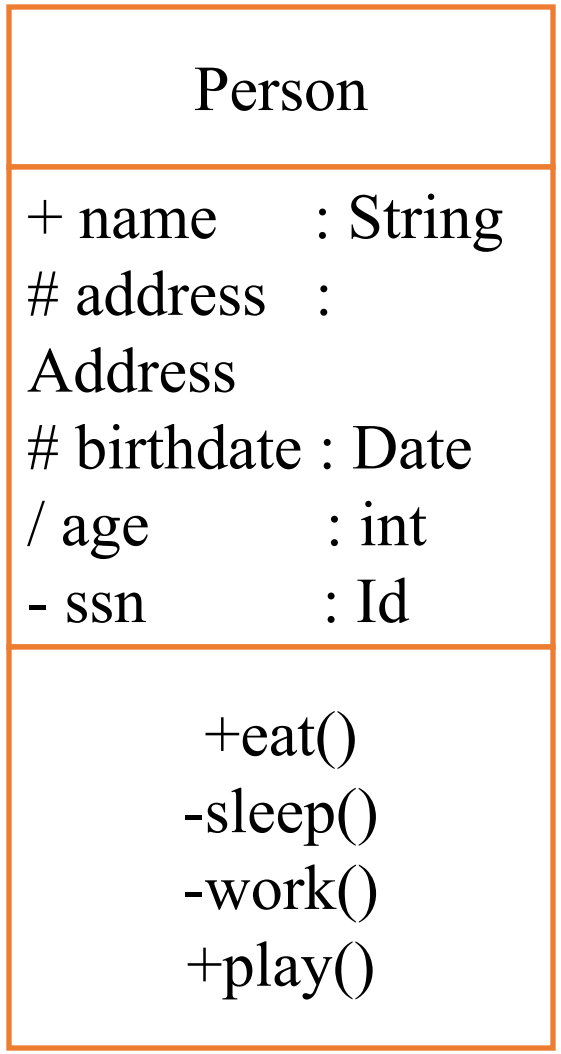
Operations describe the class behavior and appear in the third compartment.

Class operations



You can specify an operation by stating its signature: listing the name, type, and default value of all parameters, and, in the case of functions, a return type.

Visibility



attributes and operations can be declared with different visibility modes:

+ public: any class can use the feature (attribute or operation);

protected: any descendant of the class can use the feature;

- private: only the class itself can use the feature.

Finding Classes

Finding classes in use case, or in text descriptions:
Look for nouns and noun phrases in the description of a use case or a problem statement;
These are only included in the model if they explain the nature or structure of information in the application.

Don't create classes for concepts which:
Are beyond the scope of the system;
Refer to the system as a whole;
Duplicate other classes;
Are too vague or too specific (few instances)

Finding classes in other sources:
Reviewing background information;
Users and other stakeholders;
Analysis patterns;
CRC (Class Responsibility Collaboration) cards.

Object Oriented Analysis and Design with Java

CRC (Class-Responsibility-Collaborators)

CRC cards are tool used for brainstorming in OO design and agile methodologies.

CRC cards are created from Index cards and each member of the brain storming session with write up one CRC card for each relevant class/object of their design

Objects need to interact with other objects (Collaborators) in order to fulfill their responsibilities

Since the cards are small, prevents to get into details and give too many responsibilities to a class

They can be easily placed on the table and rearranged to describe and evolve the design

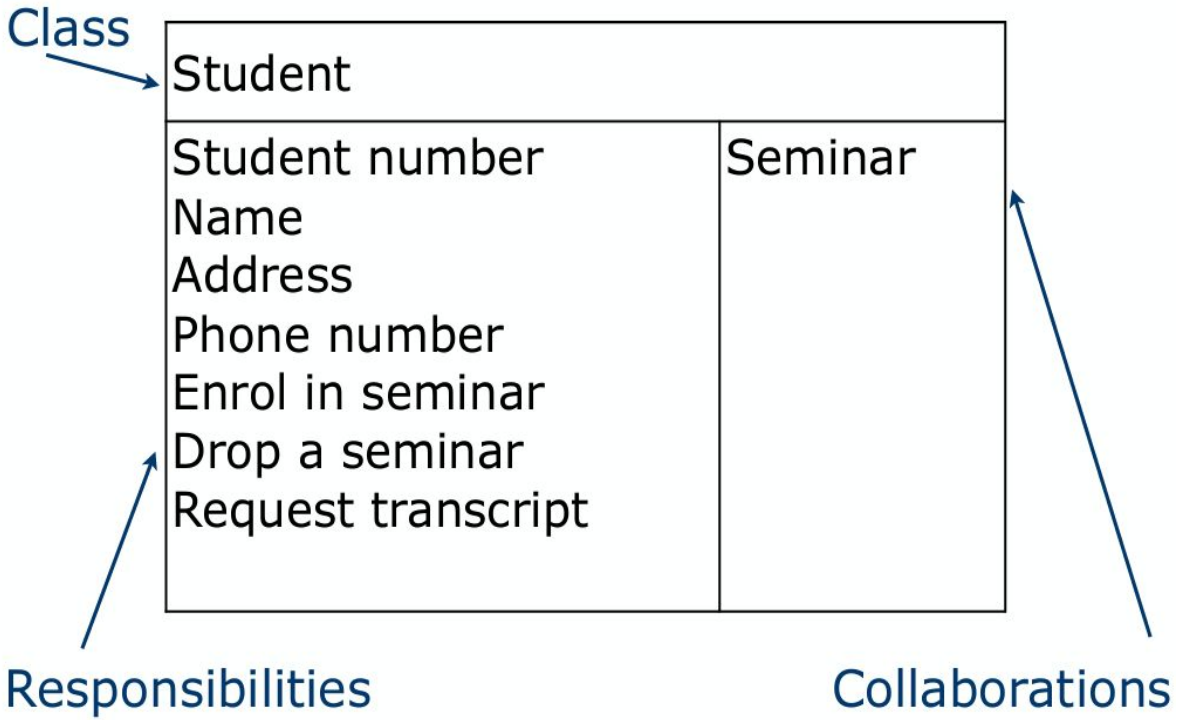
What is a CRC Card?

CRC stands for Class, Responsibility and Collaboration.

- 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

Class Name	
Responsibilities	Collaborators

Class: Account	
Responsibilities	**Collaborators**
Know balance	
Deposit Funds	Transaction
Withdraw Funds	Transaction, Policy
Standing Instructions	Transaction, StandingInstruction Policy, Account



students only have information about themselves (their names and so forth), and not about seminars. What the student needs to do is collaborate/interact with the card labeled *Seminar* to sign up for a seminar. Therefore, *Seminar* is included in the list of collaborators of *Student*.

A responsibility is anything that a class knows or does.
For example, students have names, addresses, and phone numbers. These are the things a student knows. Students also enroll in seminars, drop seminars, and request transcripts. These are the things a student does.
The things a class knows and does constitute its responsibilities.

CRC cards are an aid to a group role-playing activity .
Index cards are used in preference to pieces of paper due to their robustness and to the limitations that their size (approx. 15cm x 8cm) imposes on the number of responsibilities and collaborations that can be effectively allocated to each class.
A class name is entered at the top of each card and responsibilities and collaborations are listed underneath as they become apparent.
For the sake of clarity, each collaboration is normally listed next to the corresponding responsibility.
From a UML perspective, use of CRC cards is in analyzing the object interaction that is triggered by a particular use case scenario. The process of using CRC cards is usually structured as follows.
1.Conduct a session to identify which objects are involved in the use case.

2. Allocate each object to a team member who will play the role of that object.

3. Act out the use case.

4. Identify and record any missing or redundant objects.

THANK YOU

Prof. J Ruby Dinakar

Department of Computer Science and Engineering

rubydinakar@pes.edu