104KM Enterprise Information System

Topic: Objects, Classes and Instances

Aim:

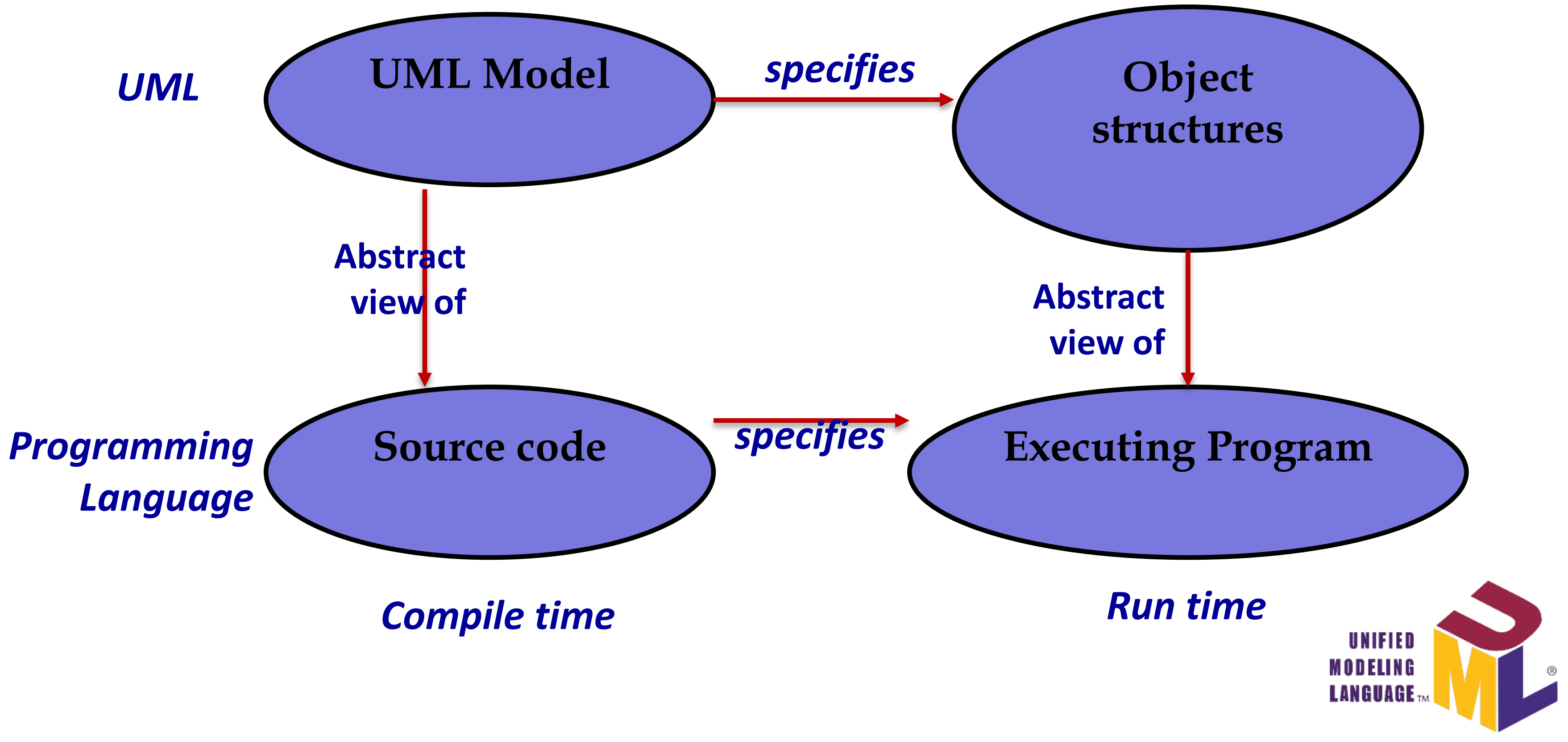
- **What is an object?**
- **What is a class?**
- **What are instances?**

Software Development

*A **successful** software organization is one that consistently deploys **quality software** that meets the **needs of its users**. An organization that can develop such software in a **timely** and predictable fashion, with an efficient and effective use of **resources**, both human and material, is one that has a **sustainable** business.*

Design Models and Code

A **system design** and the **source code** that implements it should at least in principle, be closely related and consistent with each other.



UML Diagrams

No.	Diagram	View
1.	Use case diagram	Use case view
2.	Object diagram	Use case and design view
3.	Sequence diagram	Use case and design view
4.	Collaboration diagram	Use case and design view
5.	Class diagram	Design view
6.	Statechart diagram	Design view
7.	Activity diagram	Design view
8.	Component diagram	Implementation view
9.	Deployment diagram	Deployment view

A Class

- A class is a collection of objects with common:
  - Structure
  - Behaviour
  - Relationships
  - Semantics
- Sometimes referred to as entity classes or model class.
- Represents persistent database objects.

A Class

- Models the static structure of a system rather than how it behaves.
- System structure, as modelled in a class diagram, is based on data.
- Information requirements to support functionality identified is **Use Case Diagram**.
- Combines visual representations for modelling elements.

Objects

- An object is a ‘thing’ in the real world.
- Represented as physical things
  - ***Customer, products, members, books***
- Conceptual things
  - ***Orders, loan, reservations, cancellations***
- Organisational things
  - ***Companies or departments***
- Computer implementation features
  - ***GUI, files or linked list***

Stages in developing a Class Diagram

- Identify objects and derive objects from them.
- Identify attributes of classes.
- Identify operations of classes.
- Identify relationships between classes, using associations.
- Iterate and refine the model.

- ***The UML's class diagrams are a superset of Entity-Relationship diagrams, a common modelling tool for logical database design.***

- ***Whereas E-R diagrams focus on data, class diagrams go a step further by permitting the modelling of behaviour.***

Identifying Objects

- Iterative task
- Starting point – **Use case documentation**
- Use case diagram
- Nouns
- Candidate classes

Encapsulation

- Modularity
  - Data and operations are combined into a single construct called the object.
  - Provides a building block and ideal for producing reusable units of code.
- Abstraction
  - All objects which the client needs to know about is provided in the public interface.
- Data Hiding
  - Data can only be accessed through the object’s operation.
  - Operations designed to ensure that data is handled properly.

Representing Objects

Each real world object will have certain properties that as a developer you might be interested in.

e.g.   A Car

Class   **Car**

Attributes

- fuel
- speed

Operations

- start
- accelerate

Instances

- A single object can be also be called an ‘instance’.
- Every object is an instance of some class.
- Therefore a class is an abstract descriptor for a set of instances.

Classes

- A class is a set of objects that share a common structure and behaviour.
- A template for creating objects.
- Has structure – attributes and operations.
- All instances of that class will have the same structure.

Class Notation

<u>Class</u> **Car**

Class name

<u>Attributes</u>

- fuel
- speed

<u>Operations</u>

- start()
- stop()
- moveForward()
- accelerate()

Begin with lower case letter and uppercase after

Relationships

In UML, there are different types of relationships:

Associations

Dependency

Inheritance

Aggregation

Composition

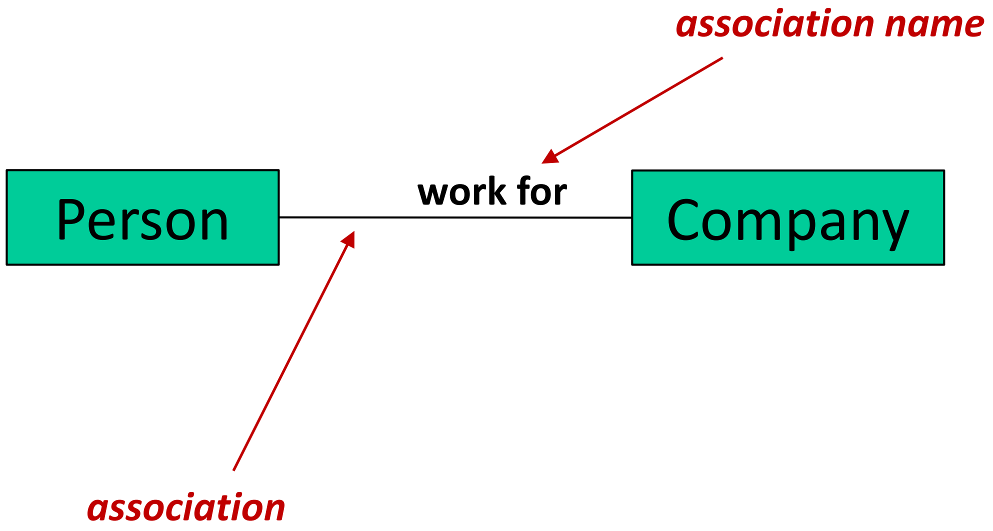
Association

- A bi-directional connection between classes. The communication can be between use cases, actors, classes.

- Uni-directional connection between classes is drawn with single arrow at one end of the association. The end of the arrow indicates who or what is receiving the communication.

- Associations are the most general of all relationships and consequently the most semantically weak. If two objects are usually considered independently, the relationship is an association.

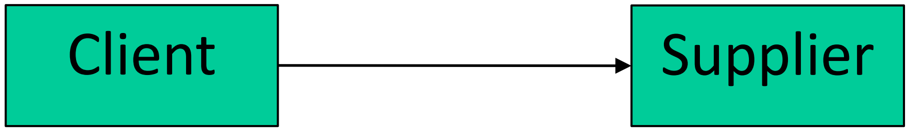
Association Name



Dependency

- A dependency is a relationship between two elements (e.g. classes).
- Dependency relationships are used to connect model elements with the same level of meaning.
- A dependency relationships indicates that the operations of a class (Client) invoke operations of the associated class (Supplier).

Dependency



- *Client class does not have the semantic knowledge of the supplier.*
- *The client class depends on Supplier class to do something.*

Examples

- Bank accounts
  - A sole account has one and only one account holder
  - A joint account has exactly two account holders
  - A business account may have two or more account holders

- Campaign case study
  - A staffMember works on a Campaign
  - A Campaign contains an Advert
  - A StaffMember is the StaffContact for none, one or more than one Client