Object Hierarchy and Generalization

Lesson 3.1

Learning Outcomes

LO 3.1.1 **Facilitate** object organization through inheritance

LO 3.1.2 **Generalize** more than one class into a parent class to simplify attribute and method declaration and implementation

LO 3.1.3 **Read** and **design** UML diagrams applied with inheritance

Object Hierarchy

Object hierarchy is a way of organizing objects into *families* based on their **shared** characteristics or **attributes**. This helps in creating a better understanding of the relationships between different objects and their behavior.

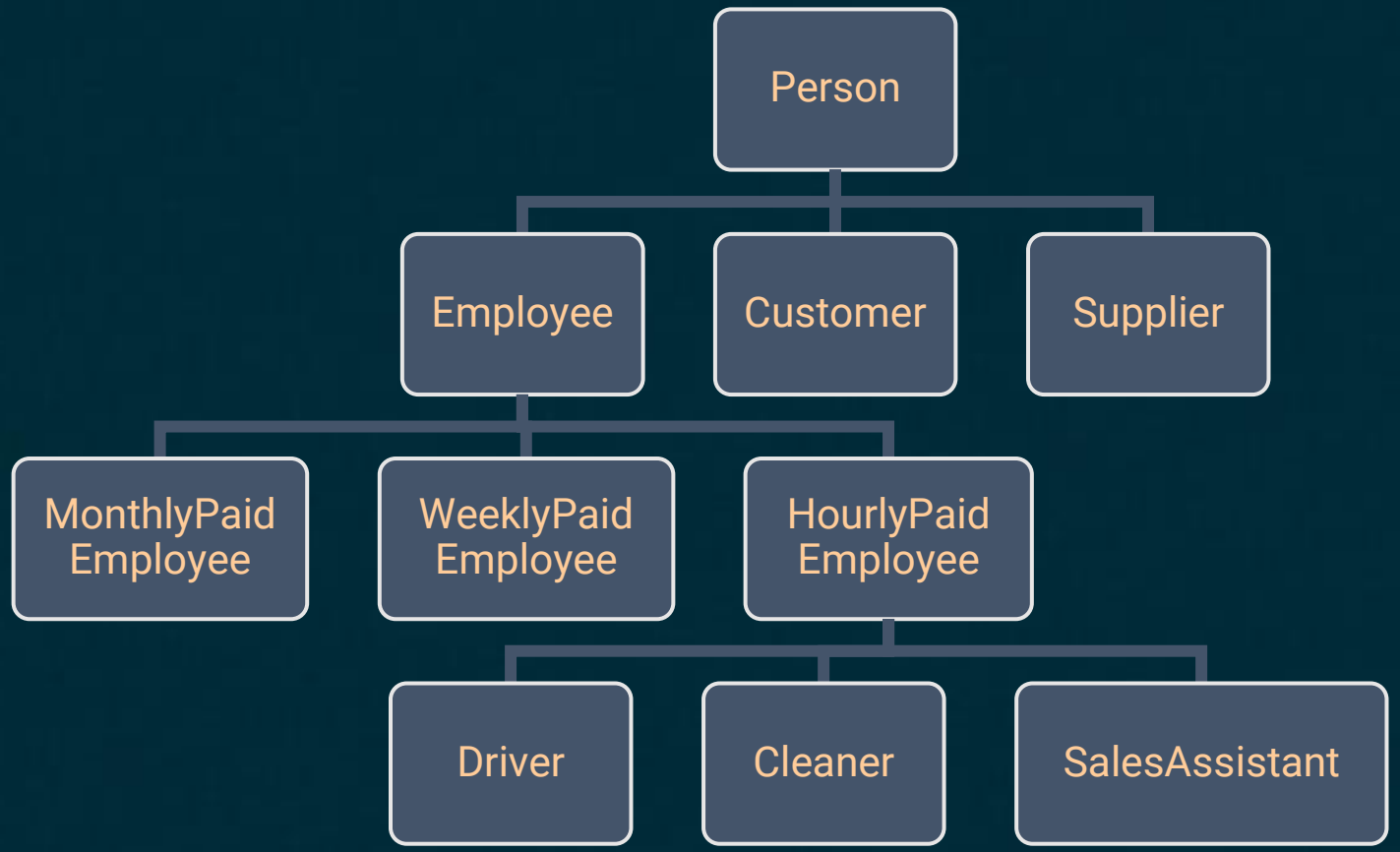
Since objects are organized in hierarchy, objects clustered in a category serves as the *specifications* of that category while the category itself is the *generalization* of those objects.

Generalization/Specification

Generalization/specification is hierarchic in nature

- A **person** may be an **employee**, a **customer** or a **supplier**
- An **employee** may be paid **monthly**, **weekly**, or **hourly**
- An **hourly-paid employee** may be a **driver**, a **cleaner**, or a **sales assistant**

Generalization/Specification



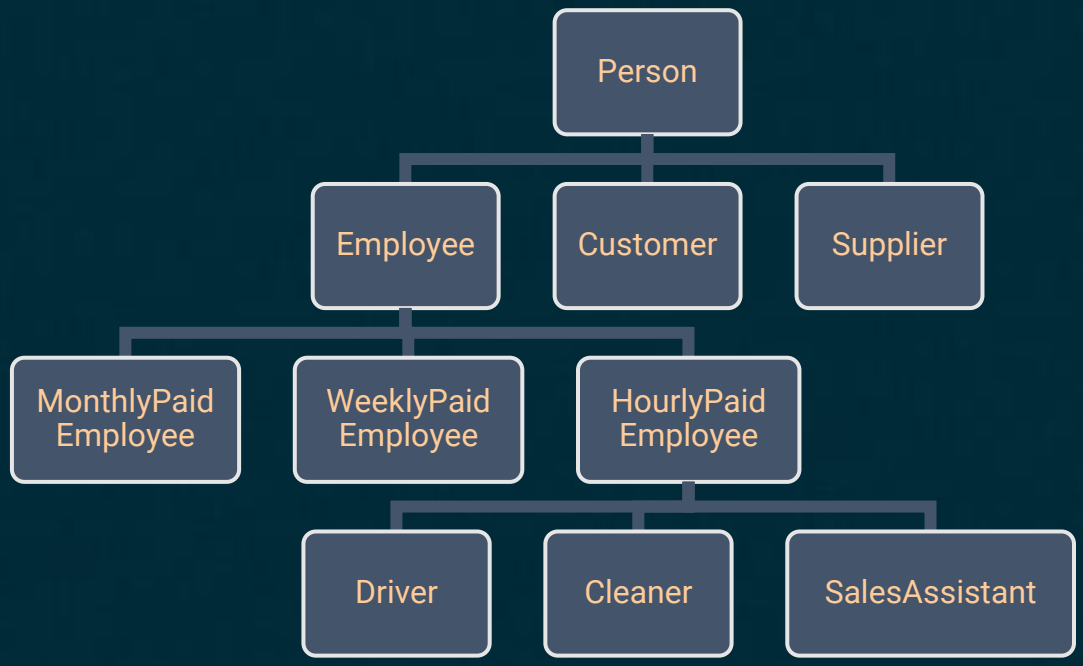
More general
(superclasses)

More specific
(subclasses)

Generalization/Specification

Based on the hierarchy, we can say:

1. A *customer* **is a** *person*.
2. An *hourly-paid employee* **is an** *employee*
3. A *sales assistant* **is an** *hourly-paid employee*, **is an** *employee*, and **is a** *person*

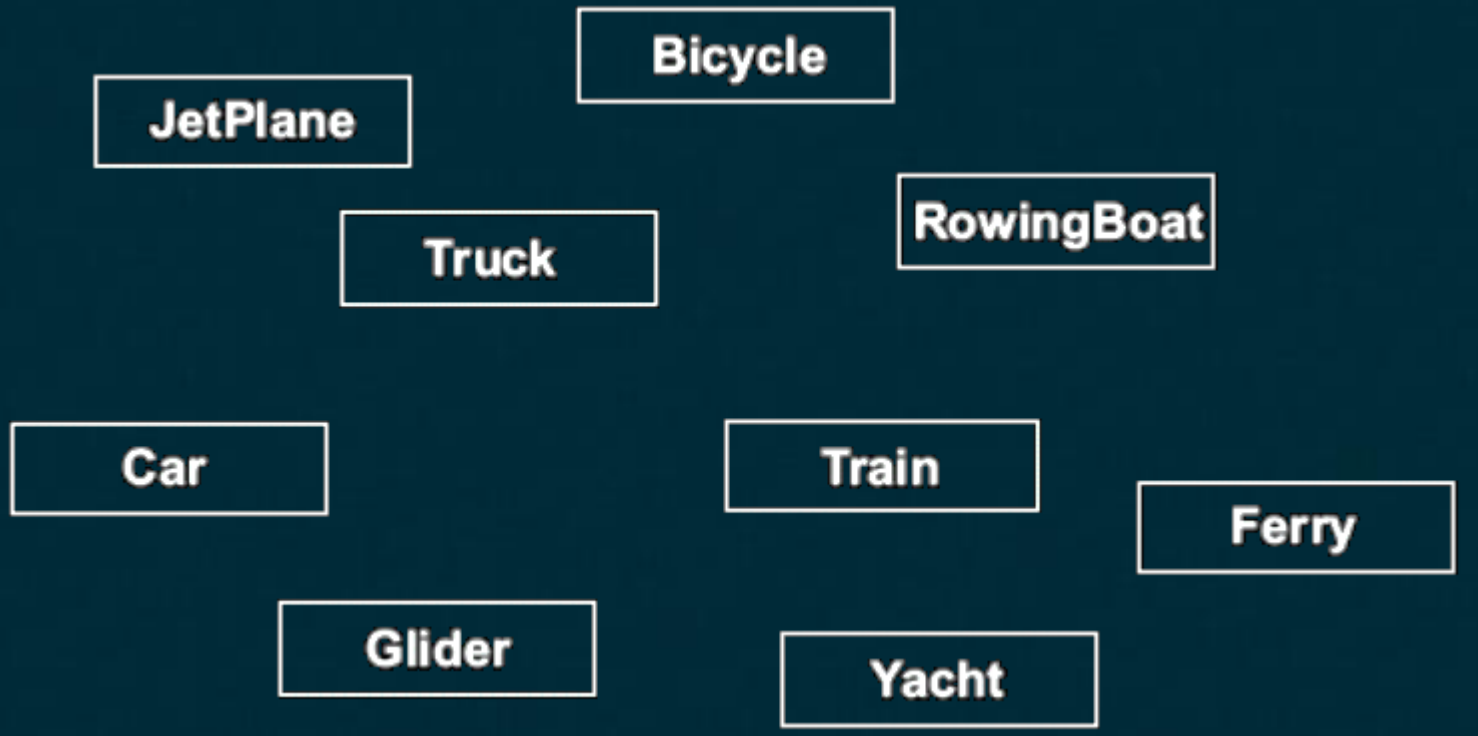


Generalization/Specification

This ‘**is a**’ relationship (*subclass* **is a** *superclass*) instilled on classes/objects is another basic concept of object-oriented paradigm called **inheritance**.

LO 3.1.1 Facilitate object organization through inheritance

How shall we organize these classes into object hierarchy?



Inheritance

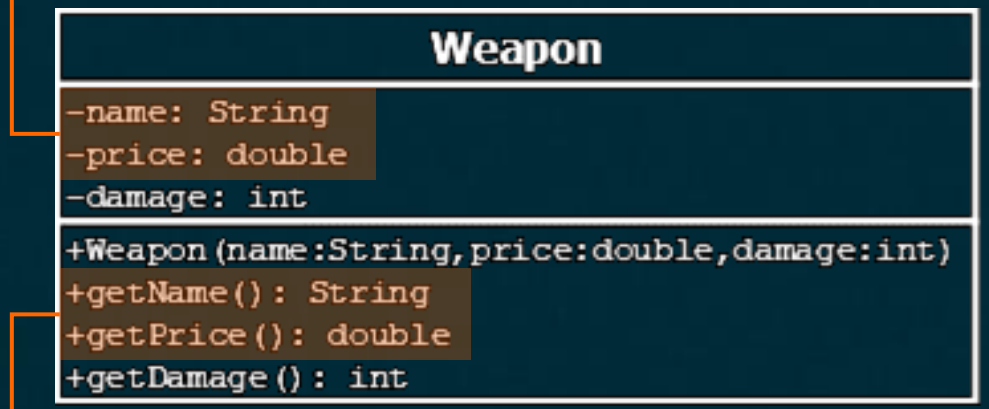
Inheritance is one of the fundamental concepts in object-oriented programming that allows us to define a new class based on an existing class. The existing class is called the **superclass**, while the new class is called the **subclass**.

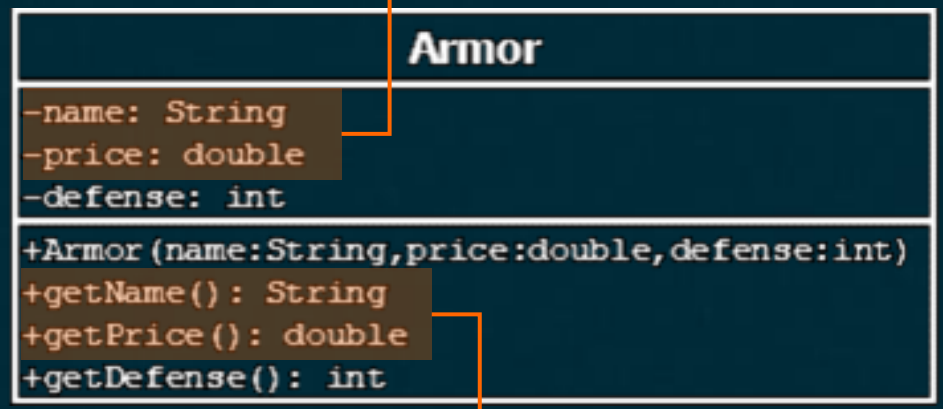
Inheritance

By inheriting from a superclass, the subclass automatically inherits all the attributes and methods of the superclass. This allows us to **reuse** code and **avoid duplicating** code across multiple classes, leading to more efficient and maintainable implementation.

Inheritance

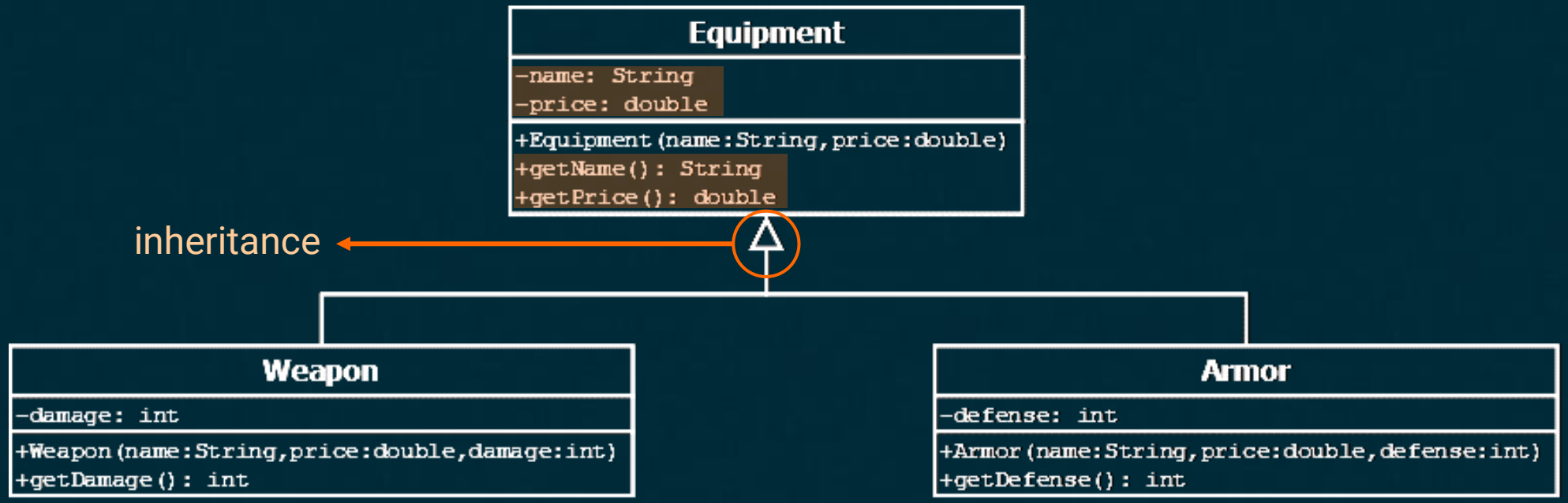
duplicate attributes





duplicate methods

Inheritance



Weapon **is an** Equipment

Armor **is an** Equipment

LO 3.1.2 Generalize more than one class into a parent class to simplify attribute and method declaration and implementation

LO 3.1.3 Read and design UML diagrams applied with inheritance

Design UML for classes **Bicycle**, **Car**, and **Ship** as *vehicles* (3 classes should have common attributes and/or methods).

Then design a class **Vehicle** which will be a *superclass* for these 3 classes then re-implement the 3 classes after applying *inheritance* from class **Vehicle**.