# Webonise Lab
Induction Program

## OOP Concepts

WWW.WEBONISELAB.COM

Object

D

Class

D

Inheritance

D

Polymorphism

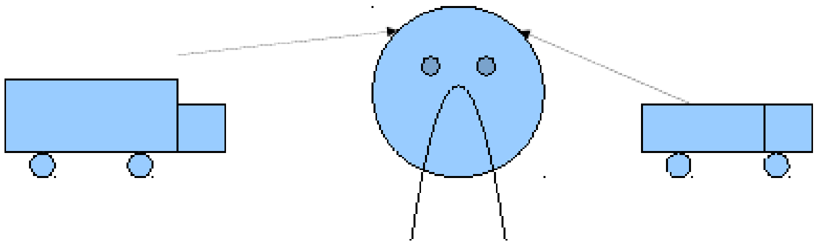
D

Abstraction

---

## Fundamentals

```
Class Vehicle{
...... ....
}
```



## Class Vehicle{ ...... .... }

# Mapping Objects to Database

Map the entire class hierarchy to a single table

Map each concrete class to its own table

Map each class to its own table

Map the classes into a generic table structure

There are two categories of mapping based on multiplicity and direction

Based on Multiplicity: One-to-one relationships. maximum of each of its multiplicities is one, Example of which is holds relationship between Employee and Position. An employee holds one and only one position and a position may be held by one employee (some positions go unfilled).

One-to-many relationships. Also known as a many-to-one relationship, this occurs when the maximum of one multiplicity is one and the other is greater than one. An example is the works in relationship between Employee and Division. An employee works in one division and any given division has one or more employees working in it.

---

Many-to-many relationships. This is a relationship where the maximum of both multiplicities is greater than one, an example of which is the assigned relationship between Employee and Task. An employee is assigned one or more tasks and each task is assigned to zero or more employees

**Based on Direction:**

Uni-directional relationships. A uni-directional relationship when an object knows about the object(s) it is related to but the other object(s) do not know of the original object. An example of which is the holds relationship between Employee and Position, indicated by the line with an open arrowhead on it. Employee objects know about the position that they hold, but Position objects do not know which employee holds it (there was no requirement to do so).

Bi-directional relationships. A bi-directional relationship exists when the objects on both end of the relationship know of each other, an example of which is the works in relationship between Employee and Division. Employee objects know what division they work in and Division objects know what employees work in them.

Relational technology does not support the concept of uni-directional relationships – in relational databases all associations are bi-directional (relationships are implemented via foreign keys, which can be joined/ traversed in either direction)

Relationships in object schemas are implemented by a combination of references to objects and operations.

When the multiplicity is one (e.g. 0..1 or 1) the relationship is implemented with a reference to an object, a getter operation, and a setter operation.

When the multiplicity is many (e.g. N, 0..*, 1..*) the relationship is implemented via a collection attribute, such as an Array or a HashSet in Java, and operations to manipulate that array.

When a relationship is uni-directional the code is implemented only by the object that knows about the other object(s).

---

**Pnsitinn**

Task