Why do OOP?

- **Pre-OOP Days – Write Code Fast as One Can.**
  - **Code was not designed – code smells bad.**
  - **Poor reuse of code – lots of copy and paste.**
  - **Code became unreadable – spaghetti code.**
  - **Difficult to trace – hard to fix bugs.**
  - **Fix one bug - breaks something downstream.**

- **Object Oriented Programming (Design)**
  - **Design First, then Code.**
  - **Promotes reuse of code.**
  - **Code becomes modular with clearly defined interfaces.**
  - **Easy to maintain and modify.**
  - **Add new objects (classes) with small differences to existing objects.**

OOP Example

***Banking Application* -** we might have checking accounts, savings accounts, money market accounts, and lines of credit.

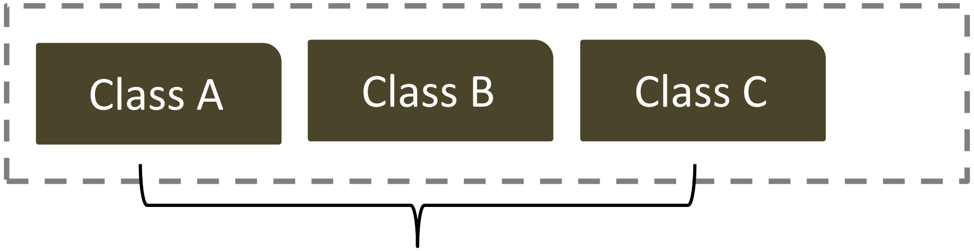
- **Some of these accounts would have similar data fields (e.g., account number, balance).**
- **Some of these accounts would have the same actions (e.g., withdraw money, get balance).**
- **Some of these accounts would have some data and actions specific to the account type not shared with the other accounts.**

Pre-OOP Days *:*

- **Write a component (set of functions) for each account.**
- **Common data fields and actions would be duplicated.**
- **When maintaining/debugging, maybe confusing which data (e.g., account number goes with which account (e.g., checking, savings, money market, line of credit).**

OOP Principles – Class & Objects

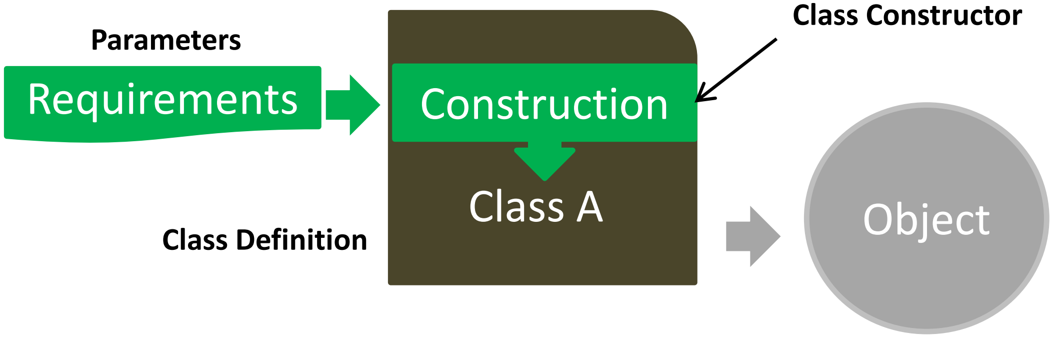
- **Class – A means to construct objects from predefined specifications (e.g, forms/templates), which may contain:**
  - **Initialized Data**
  - **Placeholders for Data**
  - **Methods for Accessing and Manipulating Data**



A collection of specifications for building Something.

Specifications on what to build and how to build it

- **Object – An instance of a class, i.e., built from a specification**



Parameters

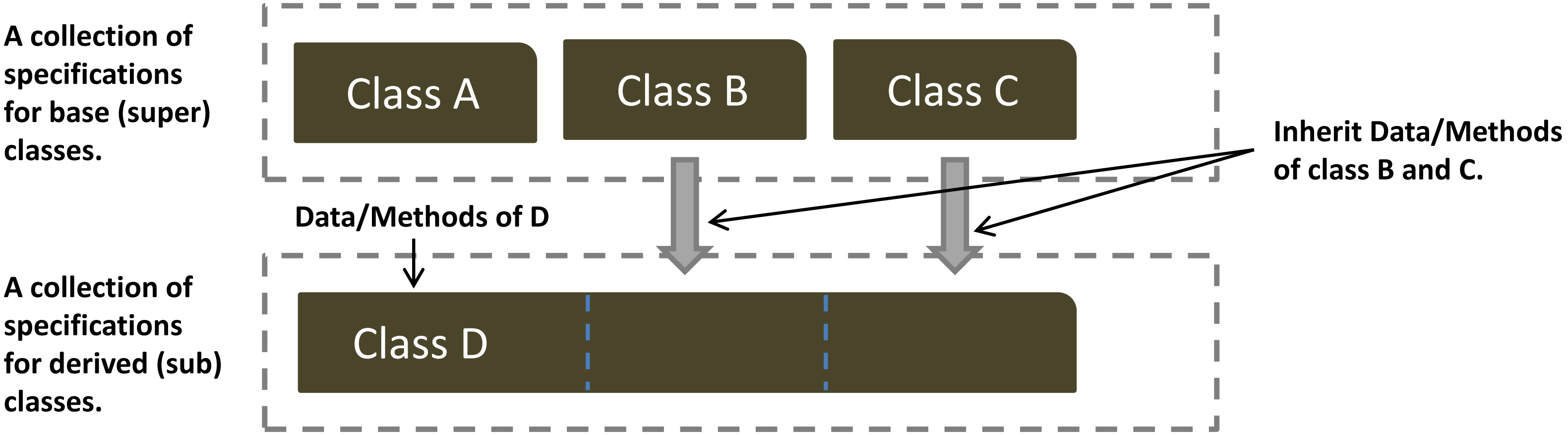
Class Constructor

Class Definition

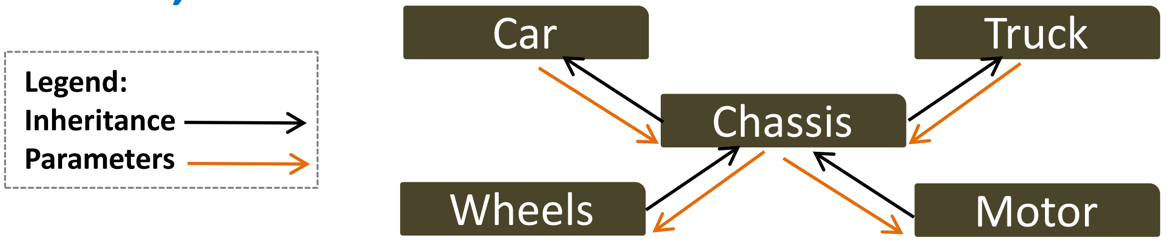
Class Instance (Object)

OOP Principles – Inheritance

- **Class Inheritance – Derived (Sub) Class**
  - **Assembly of itself (derived or sub) with one or more other classes (base or super).**


A collection of specifications for base (super) classes.
Class A
Class B
Class C
Inherit Data/Methods of class B and C.
Data/Methods of D
A collection of specifications for derived (sub) classes.
Class D

- **Hierarchical Classes – One class inherits a class, which inherits another, etc.**

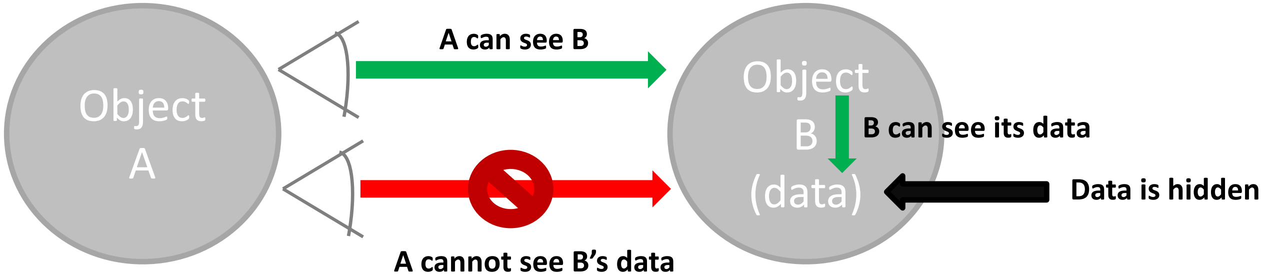


Base (super) classes are inherited Upwards into derived (sub) classes.

Parameters are passed downwards From derived (sub) to the constructors of base (super) classes.

OOP Principles – Data Encapsulation

- **Data Encapsulation – i.e., data hiding**



Object A

A can see B

Object B (data)

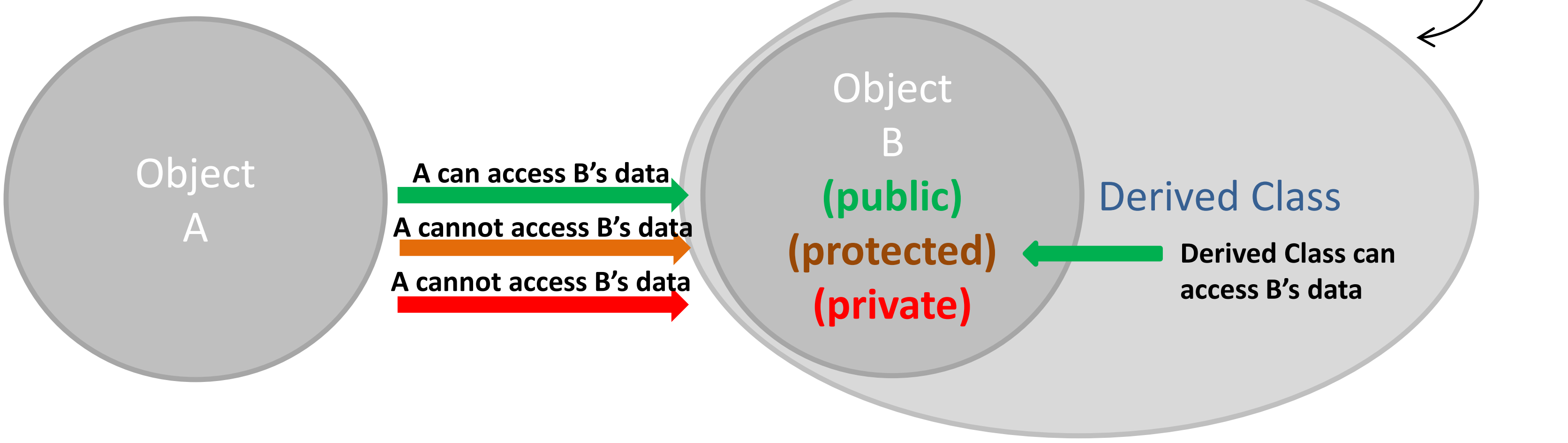
B can see its data

Data is hidden

A cannot see B's data

- **Data Accessibility**

Inheritance



Object A

A can access B's data

A cannot access B's data

A cannot access B's data

Object B (public) (protected) (private)

Derived Class

Derived Class can access B's data

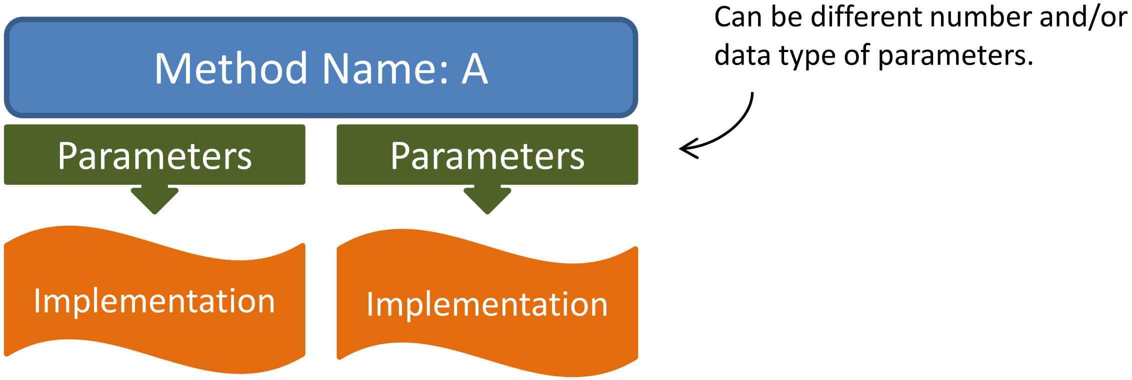
OOP Principles – Polymorphism

- **Method Overloading – The same method (function) can have multiple implementations for different parameters.**

One (same) name for the Method

Switch to implementation based on parameters

Parameters specific implementations



Method Name: A

Parameters

Parameters

Implementation

Implementation

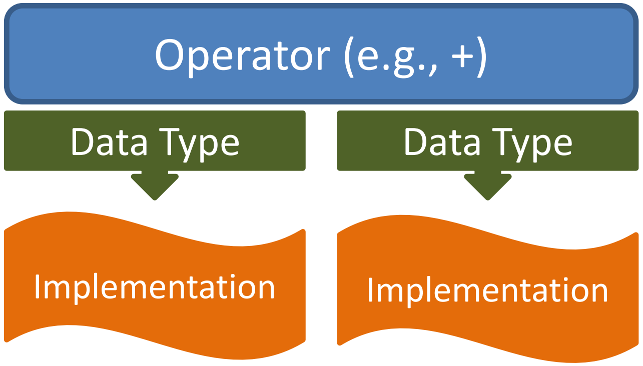
Can be different number and/or data type of parameters.

- **Operator Overloading – The same operator (e.g., +) can have multiple implementations for different data types.**

Same operator

Switch to implementation based on data type of parameters.

Data Type specific implementations



Operator (e.g., +)

Data Type

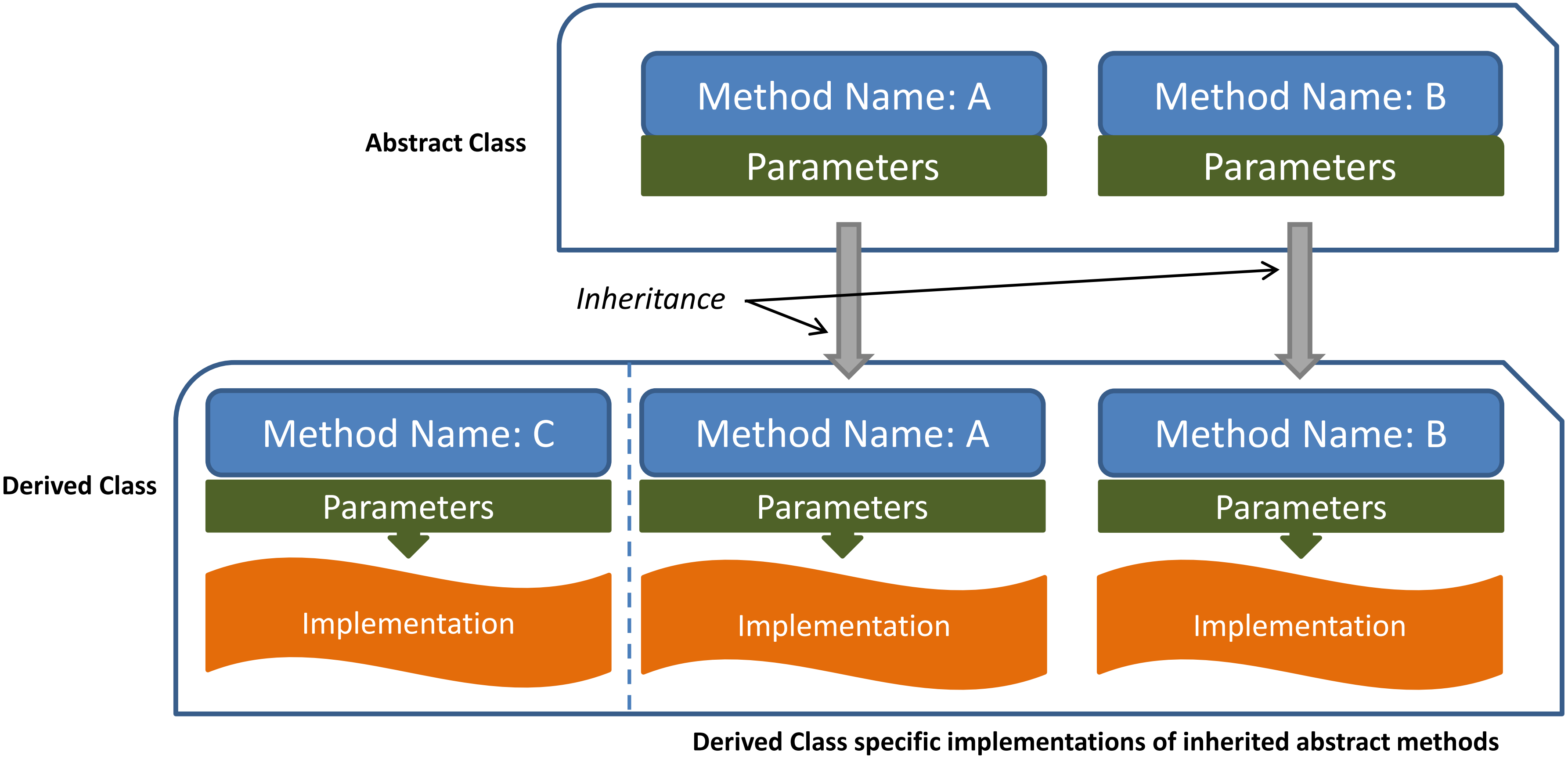
Data Type

Implementation

Implementation

OOP Principles – Abstraction

- **Abstract Classes – Reduce Complexity by Hiding Details**
  - **Has Method Signatures (declarations), but not implementation.**
  - **Abstract Methods must be implemented by derived (sub) class.**


Abstract Class
Method Name: A
Parameters
Method Name: B
Parameters
Inheritance
Derived Class
Method Name: C
Parameters
Implementation
Method Name: A
Parameters
Implementation
Method Name: B
Parameters
Implementation
Derived Class specific implementations of inherited abstract methods