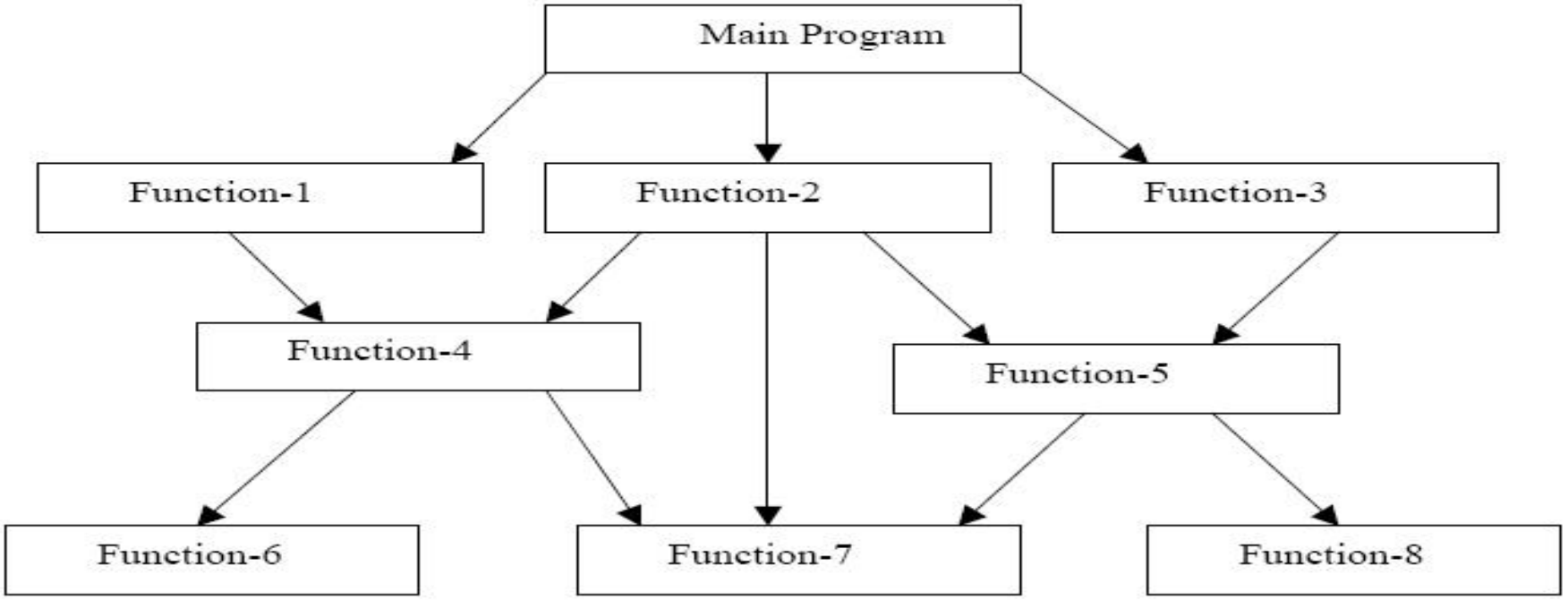
Object Oriented Paradigm

Object Oriented Concepts

By : Ashish K Layek

Procedural Programming

- The problem is viewed as the sequence of things to be done such as reading, calculating and printing
- The primary focus is on functions - the technique of hierarchical decomposition has been used to specify the tasks to be completed for solving a problem



Procedural Programming Contd...

- In a multi-function program, many important data items are placed as global so that they may be accessed by all the functions.
- Global data are more vulnerable to an unintentional change by a function.
- In a large program it is very difficult to identify what data is used by which function.
- In case we need to revise an external data structure, we also need to revise all functions that access the data. This provides an opportunity for bugs to creep in.
- Can not model real world problems very well - functions are action-oriented and do not really corresponding to the element of the problem

Procedural Programming Characteristics

- Emphasis is on doing things (algorithms).
- Large programs are divided into smaller programs known as functions.
- Most of the functions share global data.
- Data move openly around the system from function to function.
  - Data doesn't have a owner – many function can modify data
- Functions transform data from one form to another.
  - Difficulty in maintaining data integrity
  - Difficult to pinpoint bug sources when data is corrupted
- Employs top-down approach in program design.

Object Orientation

The Big Picture

- Object Orientation is a programming paradigm - it is not a computer language itself, but a set of ideas those supported by several languages
- The problem-solving techniques used in object-oriented paradigm is more closely models the way humans solve day-to-day problems
- An object-oriented program is structured as community of interacting agents called objects.
  - Each object has a role to play.
  - Each object provides a service or performs an action that is used by other members of the community

Object Orientation

An example

Suppose you wanted to send flowers to your **Mother** who lives in another city. To solve this problem you simply walk to your nearest **florist** run by, lets say, **Raman**. You tell **Raman** the kinds of flowers to send and the address to which they should be delivered. You can be assured that the flowers will be delivered.

Object Orientation

An example Contd...

Lets examine the mechanisms used to solve your problem

1. You first found an appropriate agent (Raman, in this case) and you passed to this agent a message containing a request.
2. It is the responsibility of Raman to satisfy the request.
3. There is some method (an algorithm or set of operations) used by Raman to do this.
4. You do not need to know the particular methods used to satisfy the request - such information is hidden from view.

This leads to our first conceptual picture of object-oriented programming:

"*An object-oriented program is structured as **community** of interacting agents called **objects**. Each object has a role to play. Each object provides a **service** or performs an action that is used by other members of the community.*"

Object Orientation

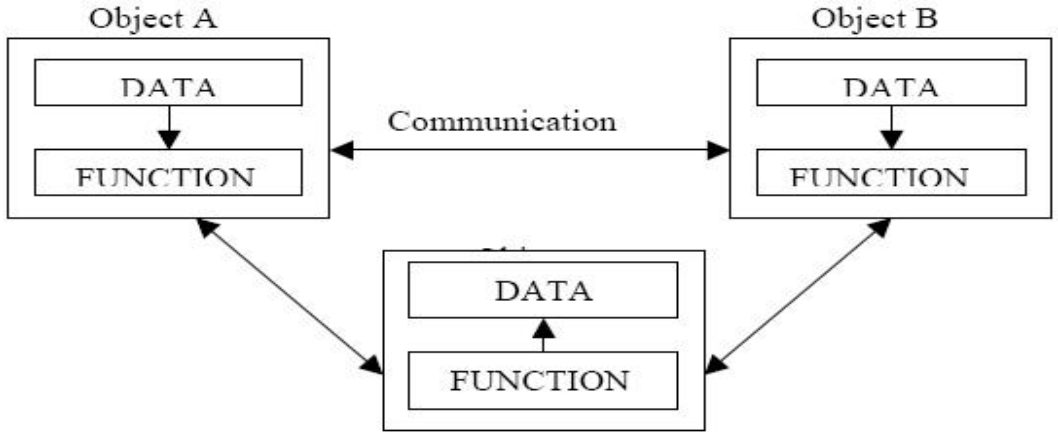
Messages and Responsibilities

Members of an object-oriented community make requests of each other.

- Action is initiated in object-oriented programming by the transmission of a message to an agent (an object) responsible for the actions.
- The message encodes the request for an action and is accompanied by any additional information (arguments/parameters) needed to carry out the request.
- The receiver is the object to whom the message is sent. If the receiver accepts the message, it accepts responsibility to carry out the indicated action.
- In response to a message, the receiver will perform some method to satisfy the request.

Object Oriented Paradigms

- OOP treats data as a critical element in the program development and does not allow it to flow freely around the system.
- It ties data more closely to the function that operate on it, and protects it from accidental modification from outside function.
- OOP allows decomposition of a problem into a number of entities called objects and then builds data and function around these objects.
- The data of an object can be accessed only by the function associated with that object
- One object can communicate to another object using functions associated with that object (message passing)



Object Oriented Programming Characteristics

- Emphasis is on data rather than procedure.
- Programs are divided into what are known as objects. Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are ties together in the data structure.
- Data is hidden and cannot be accessed by external function.
- Objects may communicate with each other through function.
- New data and functions can be easily added whenever necessary.
- Follows bottom up approach in program design.

Object and Class concept

Object

An **object** represents a tangible or intangible entity

- Physical
- Conceptual

Object are always not **physical** items and not **visible** items

Object orientation in computing is intended to thinking about programming closer to think about real world.

"What is an object in computer program => What is the object in real world ?"

Object : A more formal definition

An object is an entity that has a well-defined boundary. That is, the purpose of the object should be clear.

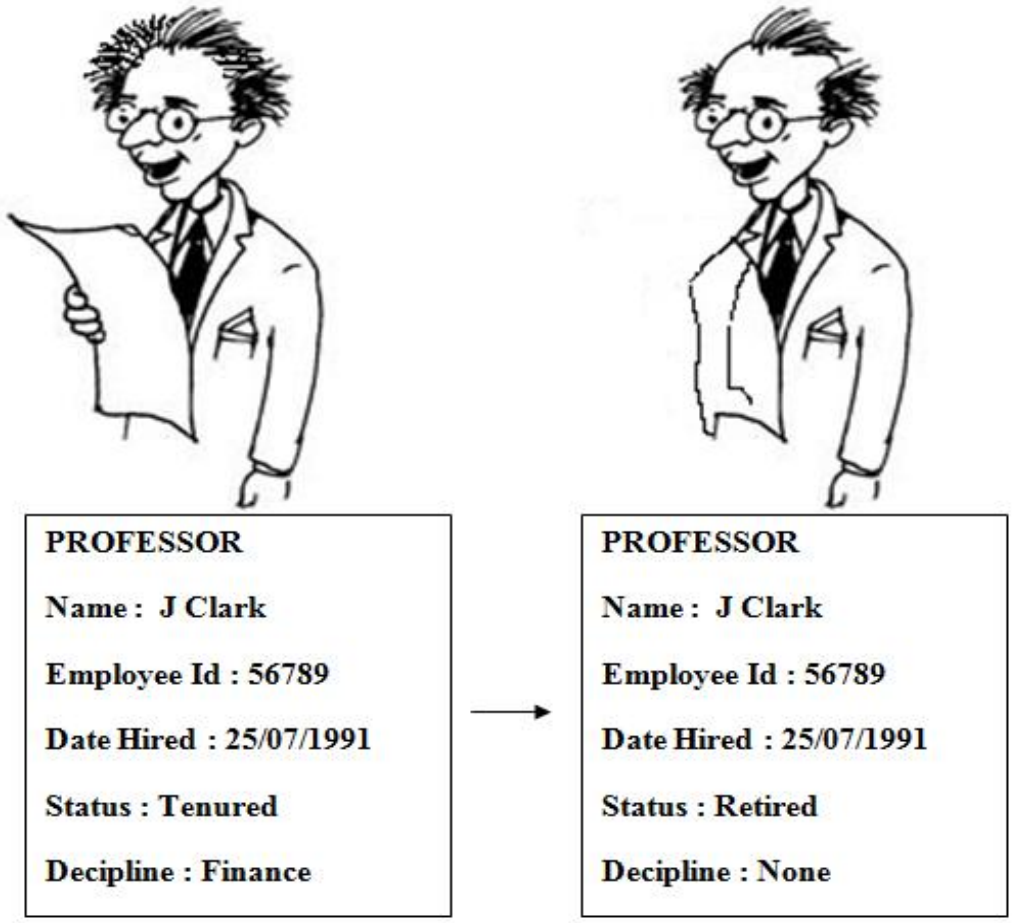
An object has three key components :

- Identity – Distinguishes one object than other
- Attributes – Information describes their current state
- Behaviour – Specific to an object

Object in a computer program are self contained. They are independent to each other.

An Object has State

- The state of an object is one of the possible conditions that an object may exists in and it normally changes over time.
- The state of an object is usually implemented by a set of properties call attributes. Most object can have multiple attributes
- State of one object is independent than another



An Object has Behaviour

- Objects are intended to mirror the concepts that they are modelled for
- Behaviour is specific to the type of the object
- Behaviour determines how an object act and react to request from other objects
- Object behaviour is represented by the operations that the objects can perform.

PROFESSOR J Clark's behavior

Take Class Tests

Submit Final Grade

Accept Course Offering

Take Sabatical

An Object has an Identity

- The term Identity means that objects are distinguished by their inherent existence and not by descriptive properties at they may have
- The same concept holds true for objects. Although two objects may share the same state (attributes), the are separate, independent objects with their own unique identity
- One object may contain another objects that, still they are separate.

Professor : Clark

Teacher Biology

Professor : Clark

Teacher Biology

Computing Object

Objects in real world generally we consider objects those we can to see or touch.

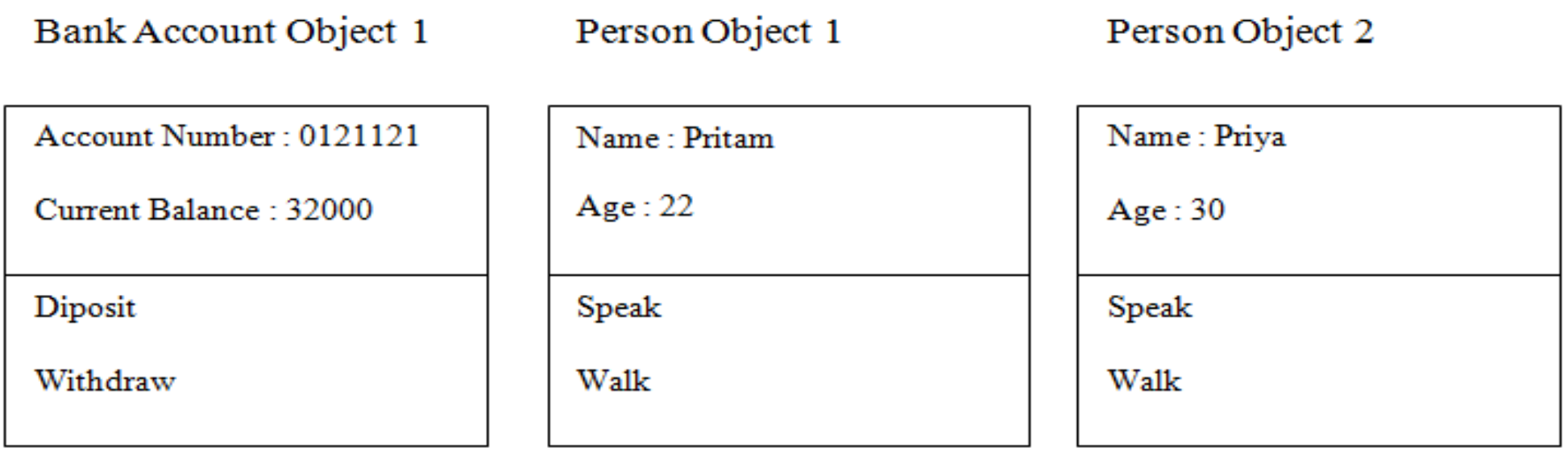
But in computing we take it further – Date, Time, link-list, Bank Account, Event can be objects. These are well defined idea.

Object are always not physical items and not visible items

Computing Object Contd...

Object in a computer program are self contained. They are independent to each other.

Like real world Object, they have identity, state and behaviour



How to find Object in computing

For those are new in Object Oriented Design, it could be potentially challenge to decide on objects in a compute program from the problem definition

It's easy to decide when a computer program need to objects like Car, Employee, document etc.

But while developing application like Event Management application, what about an Event? Is an “Event” be an object?

How to find Object in computing Contd ...

Basic clue :

In the problem statement check for the words those are common noun (physical entity or idea or concept) and add "the" in front those - like the television, the car, the bank account, the time, the event

And verbs in the problem statement are behaviour associated to those Object – like flying, printing, exploding etc.

Where do the Objects come from ?

There is another word, that goes hand-in-hand with object - that word is Class

Entire point in OOD, is not about Objects, rather Class

In Computer program Class comes first, NOT the Object

Class

A Class describes what an Object will be, but it is not the object itself

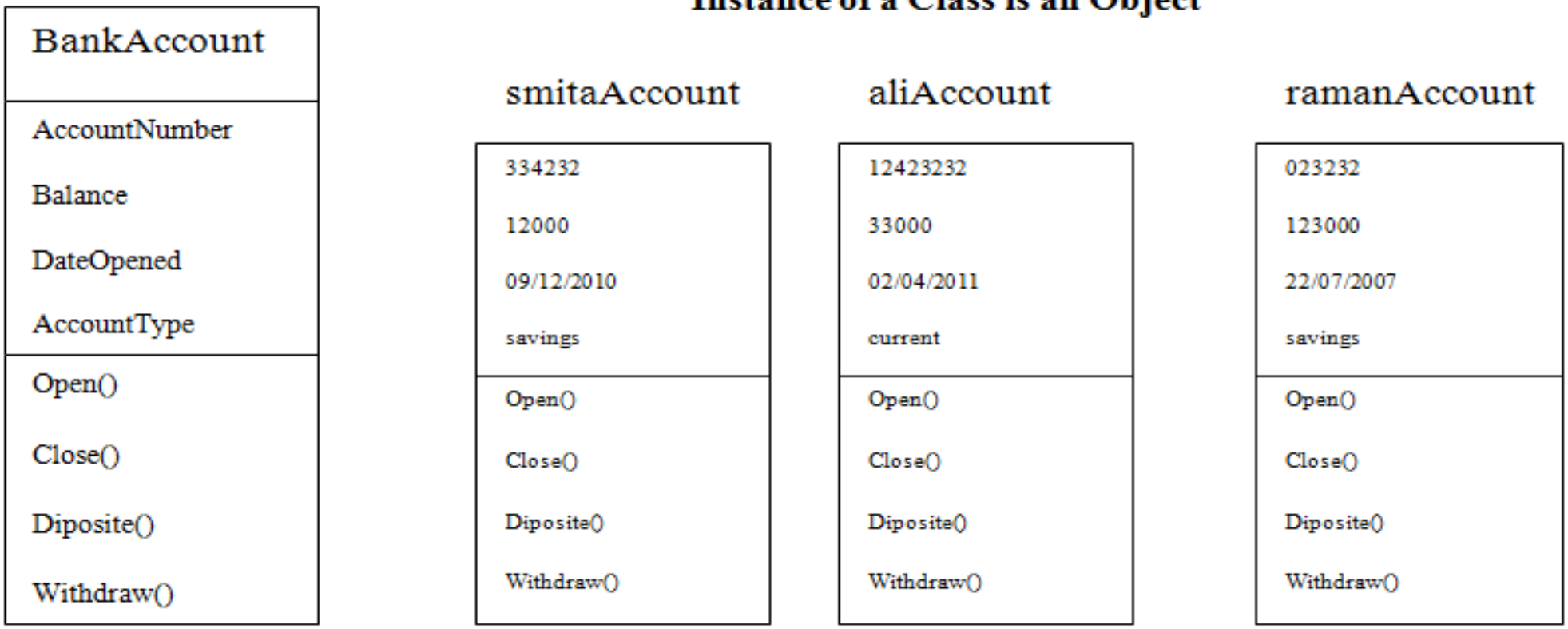
A Class is a blue print – the detail description and definition

A Class is a description of a set of objects that share the same properties (attributes), behaviour (operations), kind of relationships, and semantics.

Class / Object

Creating Objects = Instantiation

Instance of a Class is an Object



Class

Objects

Existing Classes in OOP languages

- Most OOP languages provide many pre-written generic classes at minimum : for example string, dates, file I/O, networking – so sometime you can start creating objects without writing classes
  - C++ standard Library
  - Java Class Library – more than 4000 classes available
  - .Net standard Library – more than 4000 classes available
  - Ruby Standard Library
  - Python Standard Library

- But in non-trivial application, you still need to write Classes following few principle of OOP ...

Fundamental OOP concepts

Fundamental of OOP Concepts

Abstraction

Polymorphism

Inheritance

Encapsulation

Abstraction

Abstraction means focusing the essential quality of something, rather than one specific example.

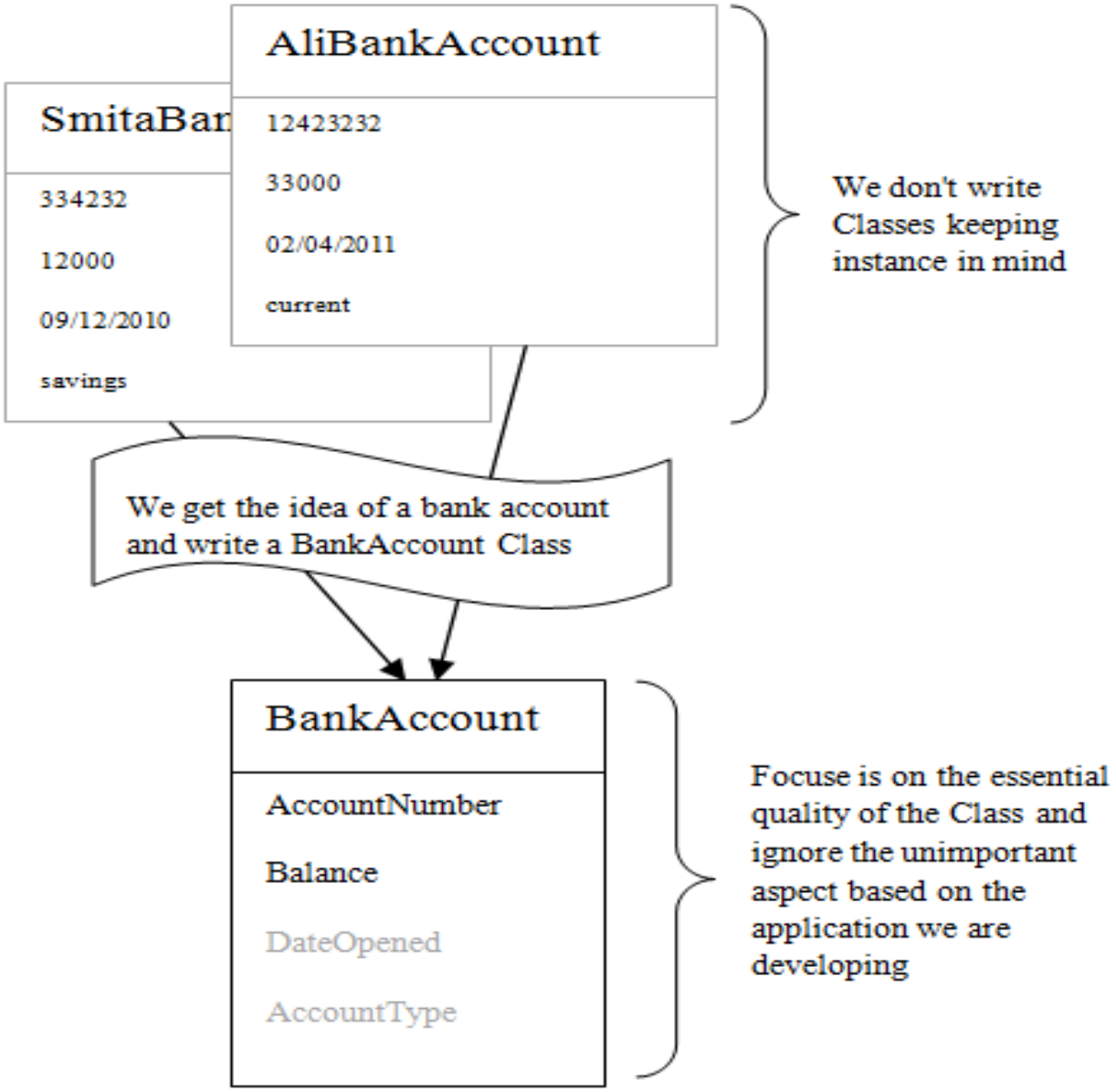
It means automatically discard the unimportant and irrelevant.

Abstraction can be defined as :

"Any model that includes the most important, essential or distinguishing aspects of something while suppressing or ignoring less important, immaterial or diversionary details."

Abstraction

Example



Abstraction

Points to remember

- Abstraction is as an external view of the object.
- Abstraction allows us to manage complexity by concentrating on the essential characteristics of an entity that distinguishes it from all other kind of entities.
- An abstraction is domain and perspective dependent. That is, what is important in one context, may not be in another.
- In the context of OOP, abstraction of an object means its outside view provided by the interface.
  - Note that the interface only tells WHAT the object can do, it does not says HOW it performs its work

Encapsulation

Encapsulation means hiding implementation detail from external world.

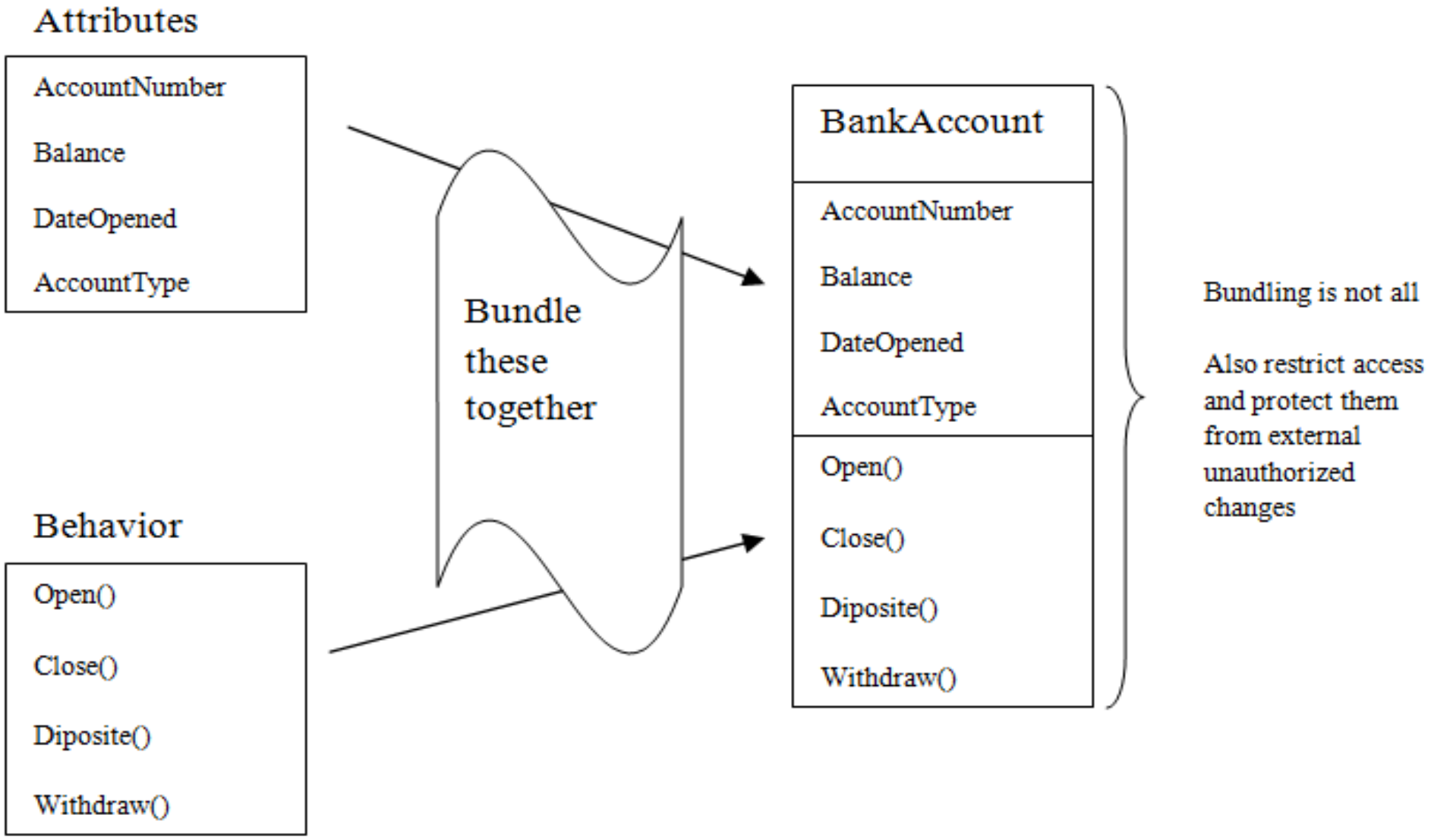
It is the idea of surrounding something, not just the keep the content together, but also to protect the content

Encapsulation can be defined as :

"The physical localization of features (e.g. properties, behaviors) into a single black-box abstraction that hides their implementation (and associated design decisions) behind a public interface"

Encapsulation

Example



Encapsulation

Points to remeber

- It is the approach of putting related things together
- Distinguishes between interface and implementation – exposes interface, hides implementation
- Encapsulation is often referred to as “information hiding”, making it possible for the clients to operate without knowing how the implementation fulfills the interface
- It is possible to change the implementation without updating the clients as long as the interface is unchanged

Inheritance

Inheritance is the process of involving building classes upon an existing classes. So that additional functionality can be added.

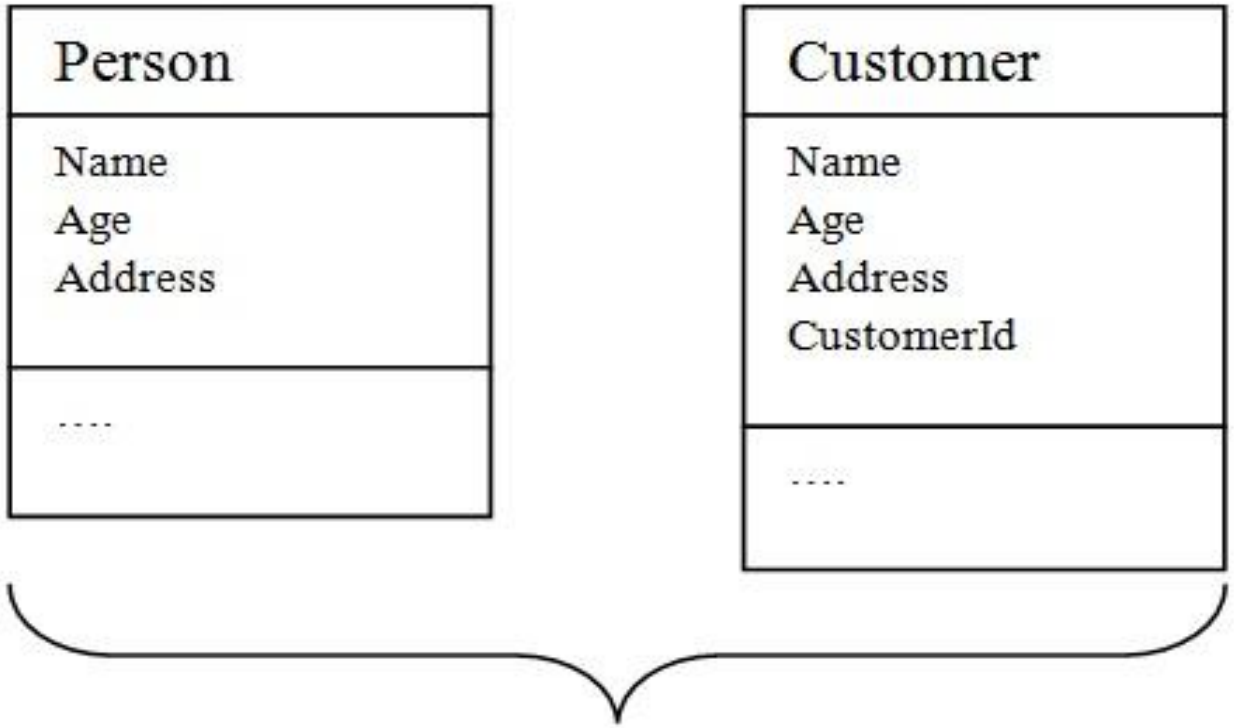
Using inheritance the hierarchical relationships are established. Inheritance allows the reusability of an existing operations and extending the basic unit of a class without creating from the scratch.

Inheritance can be defined as :

"Inheritance is a way by which a newly defined Class inherits attributes and behaviors of an existing Class along with its own properties"

Inheritance

Example



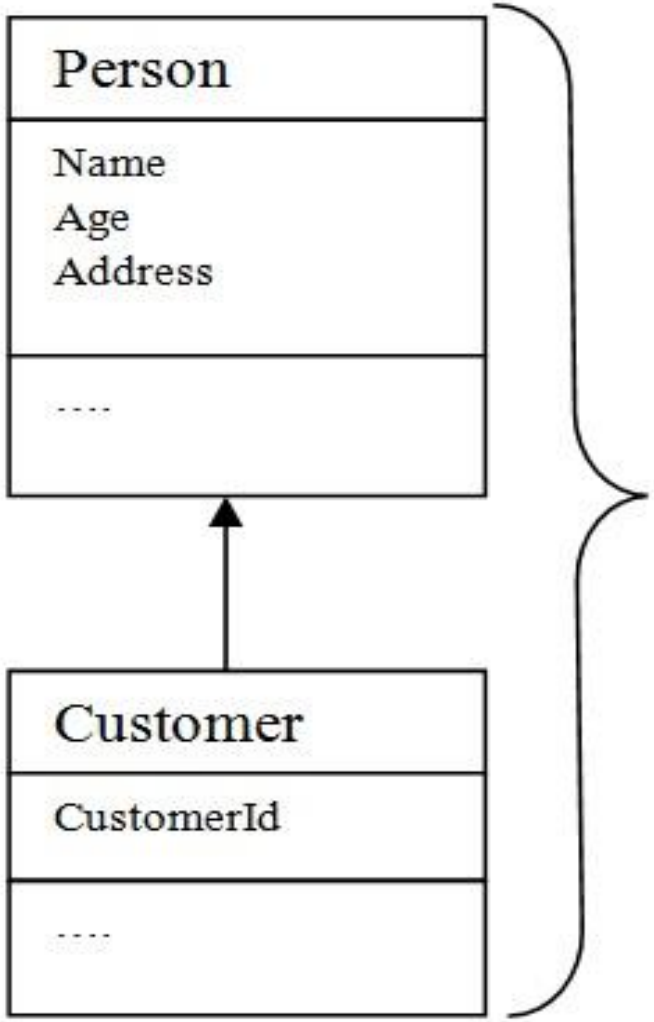
Customer class is exactly the same as Person class with some extra Attributes and Behavior.

Approaches could be -

- Writing the entire new class for Customer – but reusability is not achieved
- Add additional property to the Person Class –the Abstraction property for Person class gets violated

Inheritance

Example Contd...I



Customer class automatically has everything (such as Attributes and Behavior) in Person class without writing any code.

As well it has its own specific property

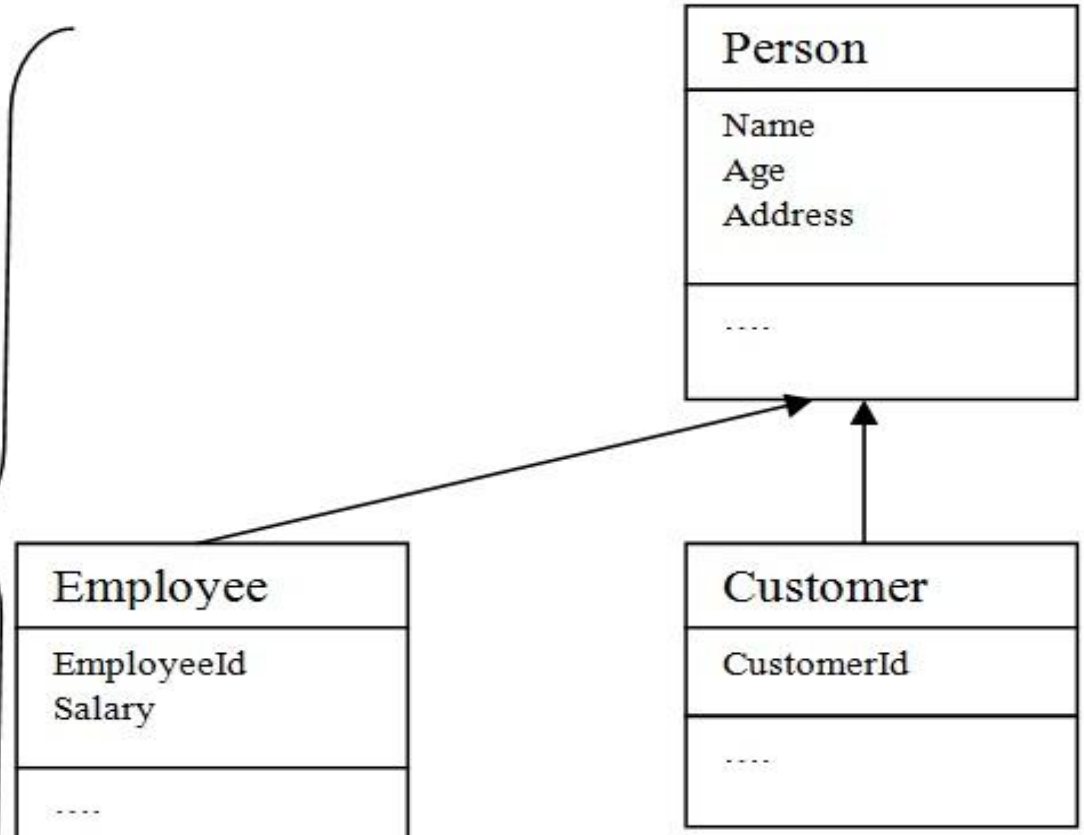
It's a "is a" relationship

Customer inherited from Person class

Inheritance

Example Contd...II

Multiple classes can be inherited from one class. Both the derived class can co-exists without impacting each other.



Employee and Customer inherited from Person class

Base Class	Parent Class	Super Class
Derive Class	Child Class	Sub Class

Inheritance

Points to remember

- Its is a "is a" relation between Classes
- Used for creating hierarchy of types
- Its a code re-use mechanism
- One Base Class can be inherited by multiple Derived Classes
- One Derived Class can inherit properties from multiple Base classes – Multiple inheritance

When do we use Inheritance ?

A derived class should pass the litmus test of "**is a**"

- 2 wheeler "is an" automobile. OK
- 4 wheeler "is an" automobile. OK
- Car "is a" 4 wheeler. OK
- Steering wheel "is a" Car. Not OK
- Event "is a" Date. Not OK

The last two are example of **Composition** or "**has a**" hierarchy, hence using Inheritance mechanism here is a misuse

- Car "has a" steering wheel. OK
- Event "has a" date. OK

Polymorphism

The Greek term polymorphism means “having many forms” - means being able to invoke different kinds of functionalities using the same interface

Polymorphism can be defined as :

"Polymorphism is sharing a common interface for multiple types, but having different implementation for different types”

Polymorphism

Analogy

A + B = ? What + sign do ?

For some language -

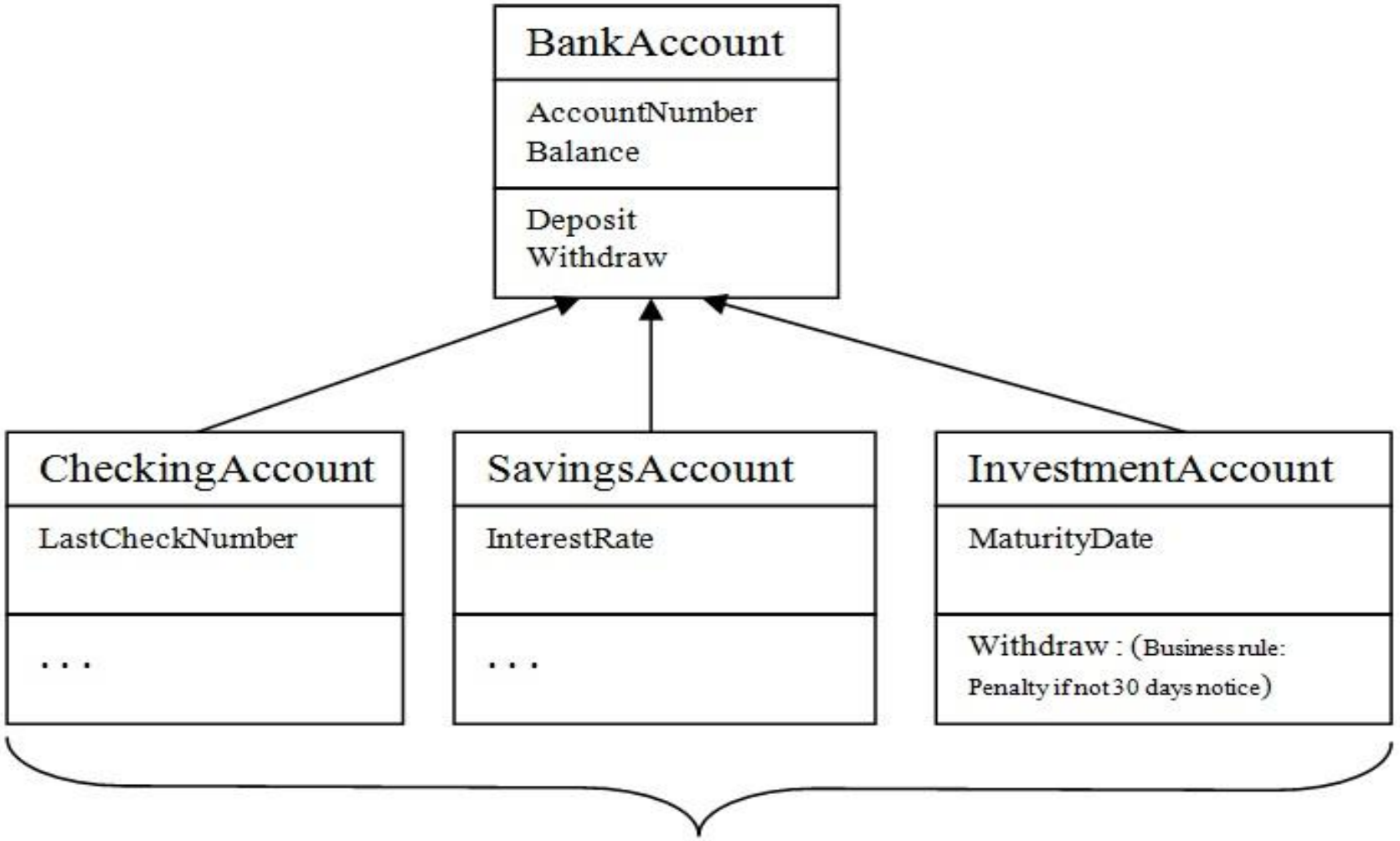
- When A and B are Integer variable having value 6 and 7, it results 13.
  - The + sign behaves as Addition

- When A and B are String variable having value “Hello” and “World”, it results “HelloWorld” .
  - The + sign behaves as Concatenation

It does automatically correct and different behavior based on the context of operation – this is the philosophy of Polymorphism

Polymorphism

Example



Withdraw behavior is written differently for this particular Class – this is **overriding**.

Rule: Inherit when useful, Override when not useful

Lets assume there is array of 10K account Objects of different types loaded in the computing system. Now Withdraw method can be called each one of these Objects without knowing the type of the Class. Withdraw method will do the correct behavior for each one – this is called Polymorphic behavior

Polymorphism

Points to remember

In OOP, polymorphism is a technique where objects of classes belonging to the same hierarchical tree (i.e. inherit from a common base class) and may possess interface bearing the same name but each having different behaviors

- Its the way of inheriting when useful, overriding when not useful
- It allows automatically do the correct behaviour even if we are working with many different forms
- There are two kinds of Polymorphism
  - Static Polymorphism
  - Dynamic Polymorphism