Core Java

Lesson 01: OOPs Concepts

Capgemini

Lesson Objectives

- In this lesson, you will learn:
  - What is Object-Oriented Programming?
  - Characteristics of OOPS
  - Class and Object
  - Object-Oriented Principles
    - Abstraction
    - Encapsulation
    - Inheritance
    - Polymorphism

What is Object-Oriented Programming

- OOP is a paradigm of application development where programs are built around objects and their interactions with each other.
  - An Object Oriented program can be viewed as a collection of co-operating objects.

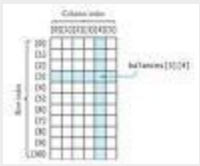
Can you think of a collection of co-operating objects in the scenario from Banking System?

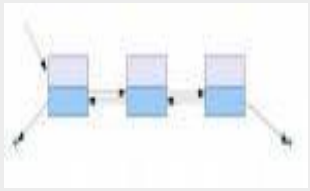
Features of Object-Oriented Programming

- Some of the major advantages of OOP are listed below:
  - Simplicity
  - Modularity
  - Modifiability
  - Extensibility
  - Maintainability
  - Re-usability

What is an Object?

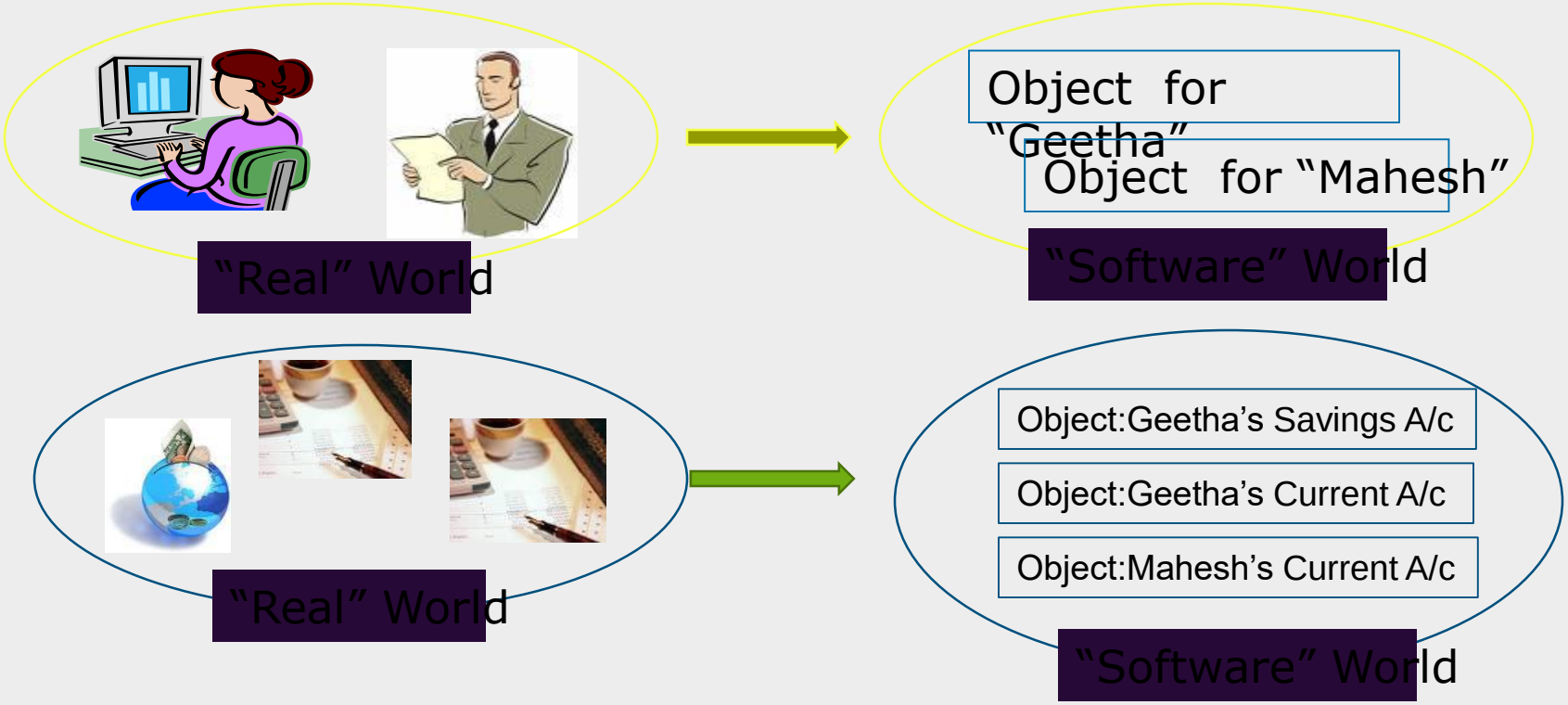
- An object is an entity which could be
  - Tangible
  - Intangible, or
  - Software entity





What is an Object?

- Entities from "Real" World would get mapped to Objects in "Software" World



What is a Class?

- A Class characterizes common structure and behavior of a set of objects.
- It constitutes of Attributes and Operations.
- It serves as a template from which objects are created in an application.

Class name	Customer
Class attributes	Name, Address, Email-ID, TelNumber
Class operations	displayCustomerDetails() changeContactDetails()

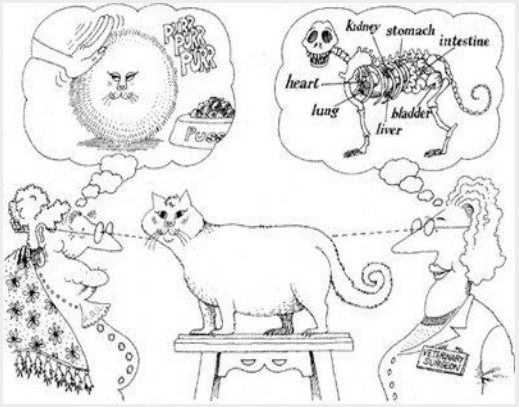
What is a Class?

- Watch out for the “Nouns” & “Verbs” in the problem statement
  - Nouns that have well defined structure and behaviour are potential classes
  - Nouns describing the characteristics or properties are potential attributes
  - Verbs describing functions that can be performed are potential operations

Example: Customers can hold different accounts like Savings Accounts and Current Accounts. Each Account has an Account Number and provides information on the balance in the Account. Customers can deposit or withdraw money from their accounts.

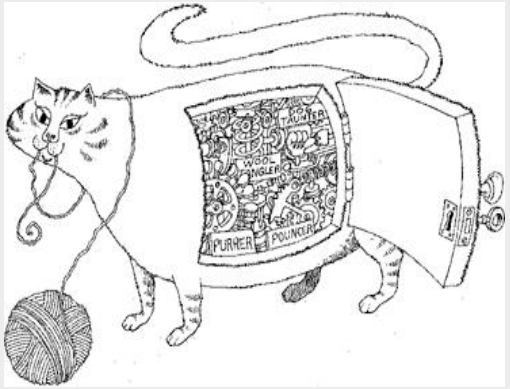
Concept of Abstraction

- Focus only on the essentials, and only on those aspects needed in the given context.
  - **For example:** Customer needs to know what is the interest he is earning; and may not need to know how the bank is calculating this interest
  - **For example:** Customer Height / Weight not needed for Banking System!



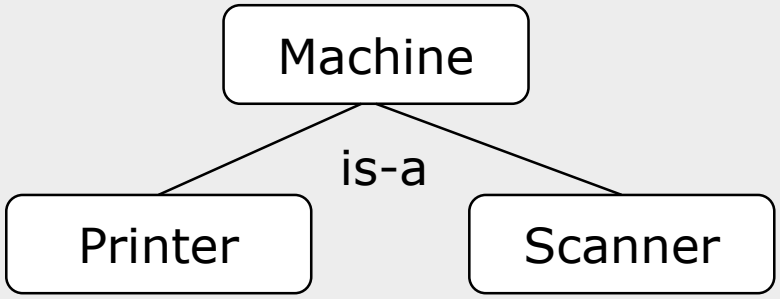
Concept of Encapsulation

- "To Hide" details of structure and implementation
  - **For example:** It does not matter what algorithm is implemented internally so that the customer gets to view Account status in Sorted Order of Account Number.

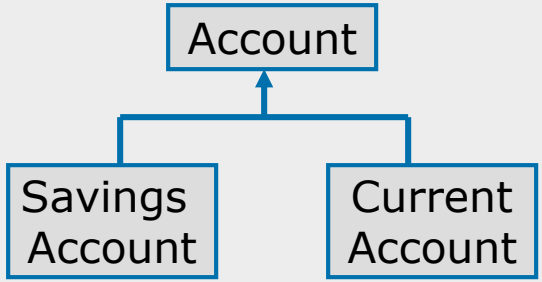


What is Inheritance?

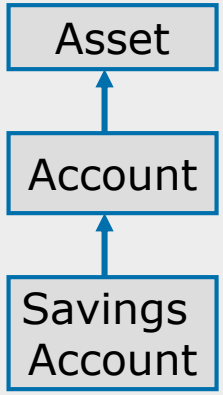
- Inheritance allows programmers to reuse of existing classes and make them extendible either for enhancement or alteration
- Allows creation of hierarchical classification
- Advantage is reusability of the code:
  - A class, once defined and debugged, can be used to create further derived classes
- Extend existing code to adapt to different situations
- Inheritance is ideal for those classes which has “is-a” relationship
- “Object” class is the ultimate superclass in Java

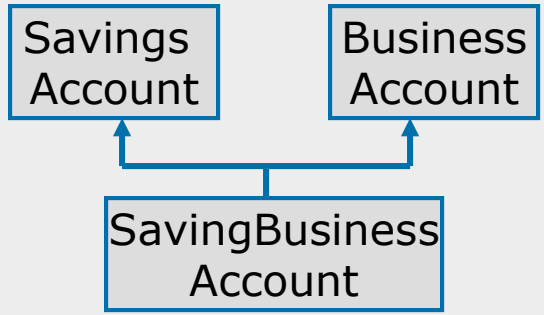


Types of Inheritance Hierarchy



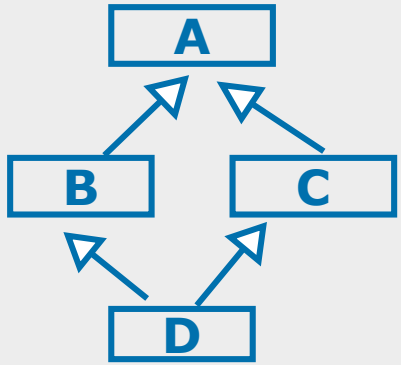
Single-level inheritance





Multiple inheritance

Multiple inheritance challenges: A name conflict introduced by a shared super-class (A) of super-classes (B and C) used with "multiple inheritance".



Polymorphism

- It implies One Name, Many Forms.
- It is the ability to hide multiple implementations behind a single interface.
- There are two types of Polymorphism, namely:
  - Static Polymorphism
  - Dynamic Polymorphism

Static Polymorphism

- Resolution of the "Form" is at compile time, achieved through overloading.

```
int addInteger(int, int);
float addFloat(float, float);
double addDouble(double, double);
```

Traditional Approach

Overloaded Functions

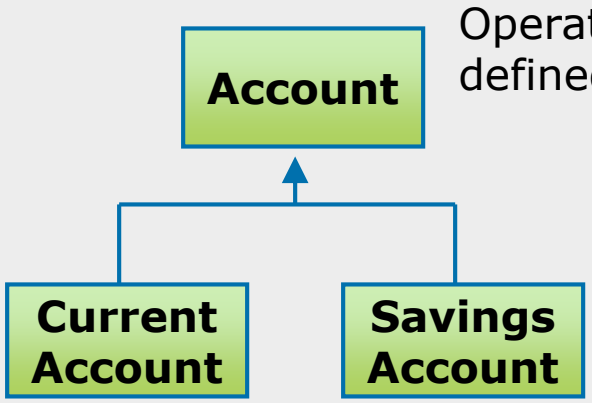
```
int addNumber(int, int);
float addNumber(float, float);
double addNumber(double, double);
```

Resolution At Compile Time

Though the name is the same, the right function is called depending on number and/or types of parameters that are there in the function invocation

Dynamic Polymorphism

- Resolution of the "Form" is at run time, achieved through overriding.



Operation calculateInterest defined in base class

Operation calculateInterest can b redefined in the derived classes

The right operation defined in one of these classes is invoked at Run Time depending on which object is invoking the operation.

Key Feature – Polymorphism

- Why Polymorphism?
  - It provides flexibility in extending the application.
  - It results in more compact designs and code.

Summary

- In this lesson, you have learnt:
  - OOPs
  - Principles of OOPS.

Review Question

- Question 1: Which of the following are features of Structured programming:
  - Option 1: Based on Data structure
  - Option 2: Emphasis on data
  - Option 3: Reusability
  - Option 4: Produces a design unique to the problem
- Question 2: Objects can be grouped together in different ways to form new programs.
  - True / False

Review Question

- Question 3 : ____ methodology specifies the steps to be sequentially followed by a computer to execute a program.