OOPS Principles

part-1

Why OOPS?

OOPS is a **methodology** introduced to represents real world objects using a program for automating real-world business by, achieving **security** because business need security.

All living and non living things are considered as object. So the real-world objects such as Person, Animal, Bike, Computer, etc... can be created in OOP languages.

Why do we need real-world objects in program? .

Because real-world object is part a business. As we develop software for automating business we must also create that business related real-world objects in the project. ;

For example:

To automate Bank business we must create real-world objects like - Customer, Manager, Clerk, OfficeAssistant, MarketingExecutive, Computer, Printer;Chair, Table, AC etc... Along with Bank object we must also create all above objects because without all above objects we cannot run Bank business. So technically we call above objects are business objects.

Definition of OOP:

OOP is a **methodology** that provides a way of modularizing a program by creating partitioned memory area for both data and methods that can be used as **template** for creating copies of such modules (objects) on demand.

Unlike procedural programming, here in the OOP programming model; programs are organized around objects and data rather than actions and logic.

Building blocks of OOP:

The building blocks of OOP are

- **class**
- **object**

Every Java program must start with,a class, because using class only we can represent real-world objects like Person, Bike, Animal, etc ...

Definition of class

- A class is a specification or **blue print** or template of an object that defines what goes to make up a particular sort of object.
- Thus a class is a **logical construct**, an object has physical reality.
- A class is a **user defined data type**.
- A class defines the structure, state and behavioμr (data & code) that will be shared by a set of objects. Hence each object of a given class contains the structure, state and behaviour defined by the class.
- Specifically, the data defined by that class are referred to as member variables or instance **variables** or attributes. The code that operates on that data is referred to as member methods or **methods.**

For example .Bike{ bikeNurnber, model, color, start(), move(), stop() }.

Definition of object

- Object is the **physical reality** of a class.
- Technically object can be defined as **"It is an encapsulated form of all non-static variables and non-static methods of a particular class".**
- An **instance of a class** is the other technical term of an object.
- The process of creating objects out of a class is called **instantiation.**
- Two objects can be communicated by passing messages-(arguments).
- For example Bike, Car, Dog, Computer, BankAccount

@codewith.shiv

Definition of instance:

An instance is a single, unique memory allocation of a class that represents that object physically with specific values:
 Bike [8192, "Pulsar 18011, Color.RED]

Technically speaking

- "**class Bike{}**"=create·s Bike·object·loglcally
- "**Bike b = new Bike()**" - creates bike object physically, nothing but instance (memory)

Q) Is object and instance both are same?
 No; memory allocated for creating object physically with specific values is called instance. This is the reason object is also defined as "instance of the class".

@codewith.shiv