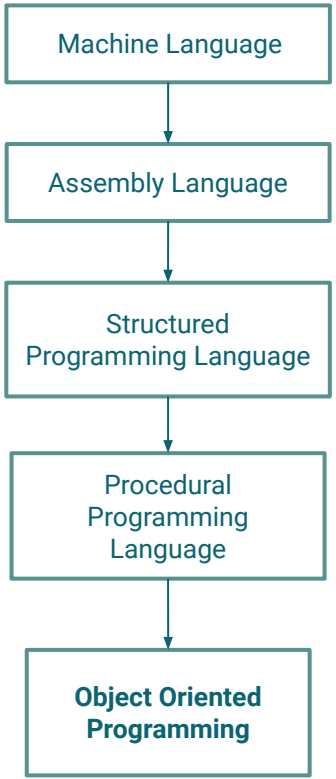
Object Oriented Programming Concepts

By:Apeksha Jadhav
Poornima
Ramya
Sanmati RM
Yashaswini S

TABLE OF CONTENTS

WHY OOPS?

- OOP was introduced to overcome flaws in the procedural approach to programming
- Such as lack of reusability and maintainability



WHAT IS OOPS?

- Object oriented programming (OOP) is a programming technique in which programs are written on the basis of objects
- In OOP, problem is divided into number of entities called objects and then builds data and functions around these objects
- Data of objects can be accessed only by the functions associated with the objects
- Communication between objects is done through functions
- Follows bottom-up approach in program design

OBJECTS

- Object is the basic unit of object oriented programming.Objects are identified by its unique name.
- An object represents a particular instance of a class.They may represent a person,a place or any item that the program must handle
- An object is a collection of data members and associated member functions also known as methods.

• Example:

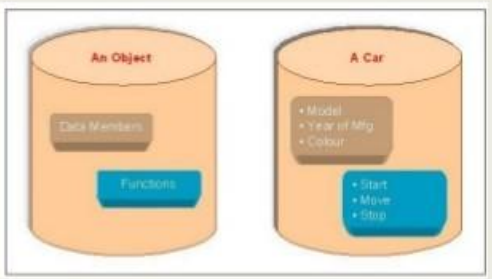


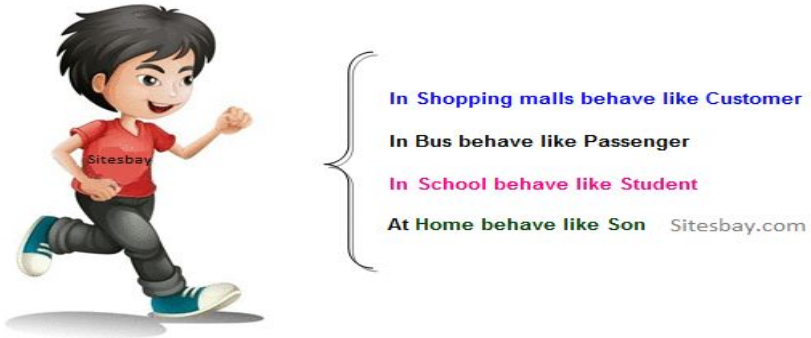
Fig 4: Representation of object.

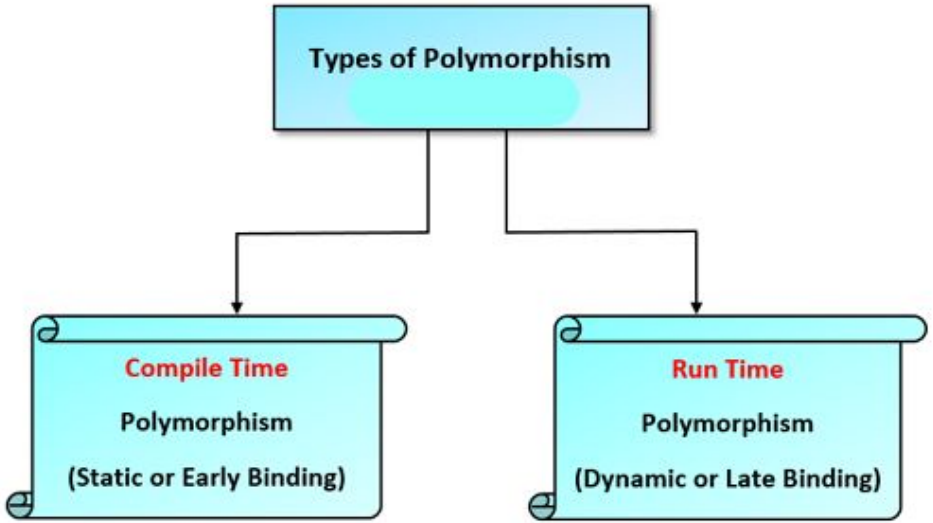
CLASSES

- Classes are data types based on which objects are created.
- Thus a class represents a set of individual objects.
- Objects are variables of class.
- A class is a collection of objects of similar type.
- Example: Student ram, sham;
- In example ram and sham are name of objects fo class Student,We can create any number of objects for class

CHARACTERISTICS:POLYMORPHISM

- Polymorphism is derived from 2 greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.
- Polymorphism is an object-oriented programming concept that refers to the ability of a variable,function or object to take on multiple forms.
- Suppose if you are in class room that time you behave like a student, when you are in market at that time you behave like a customer, when you at your home at that time you behave like a son or daughter, Here one person present in different-different behaviors.





Compile time polymorphism: It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.

Runtime polymorphism: It is also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding.

OVERLOADING

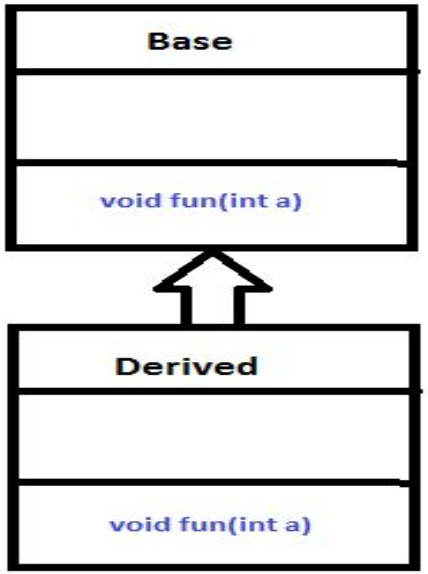
- When there are multiple functions with same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or/and change in type of arguments.
- Example: 1)By using different types of arguments

```
void fun(int a)
void fun(int a, int b)
void fun(char a)
```

- 2) We can make the operator (‘+’) for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands. So a single operator ‘+’ when placed between integer operands, adds them and when placed between string operands, concatenates them.

OVERRIDING

- **Method overriding, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.**
- You can have a method in subclass overrides the method in its super classes with the same name and signature.



Overriding

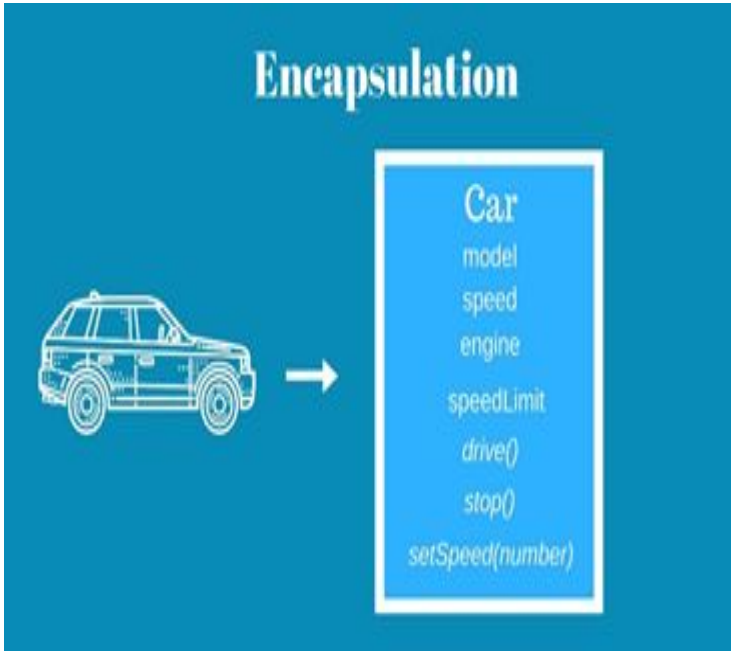
CHARACTERISTICS : ABSTRACTION

- Providing only essential information about the data to the outside world, hiding the background details or implementation.
- **Need to know:**
  - Methods of the object are available
  - Input parameters needed
- **Need not know:**
  - How the method is implemented?
  - Which kinds of actions it has to perform?
- **Example**:The man only knows applying brakes will stop the car but he does not know about the inner mechanism of the car.

CHARACTERISTICS:ENCAPSULATION

- **Binds** together the **data** and **functions** that manipulate the data.
- Hide the internal representation, or state, of an object from the outside. This is called information hiding.

- **Example**: Calculator
  - Press 2+2 then see the result on display.
  - Don't have to know how it works inside.
- **Advantages**
  - Control over the data
  - Data hiding



ENCAPSULATION ~ ABSTRACTION??????

Encapsulation:

For the mathematical equation, shown in the figure assume that complex functions are required - >But I the end we obtain result for it

Calculator shows the result of equation but hides the implementation (calculating the result) involved.

Abstraction:

The calculator shown in the figure has to be powered by a battery source. How the battery module works for the calculator is not necessary to know for the user who uses the calculator.

Using Battery module along with other modules we use calculator -> Thus using Abstraction encapsulation is performed

CHARACTERISTICS: INHERITANCE

- Mechanism in which one object acquires all the properties and behaviors of a parent object.
- The "**extends**" keyword indicates that you are making a new class that derives from an existing class.
- Why use it:
  - For Method Overriding
  - For Code Reusability

```
class Animal{
void
eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark()
{System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

CREATION OF OBJECTS

How to Declare, Create and Initialize an Object in Java

A class is a blueprint for Object, you can create an object from a class. Let's take Student class and try to create Java object for it.

Let's create a simple Student class which has name and college fields.

Let's write a program to create declare, create and initialize a Student object in Java.

Simple Program Demonstrating The Creation Of Objects Using ‘New’ Keyword

```
public class Student {

 private String name;

 private String college;

 public Student(String n, String c) {

 name = n;

 college = c;

 }

public static void main(String[] args) {

Student student = new
Student("Ramesh", "BVB");

Student student2 = new Student("Prakash",
"GEC");

 Student student3 = new Student("Pramod",
"IIT");}

}
```

From the above program, the Student objects are:

```
Student student = new Student("Ramesh", "BVB");

Student student2 = new Student("Prakash", "GEC");

Student student3 = new Student("Pramod", "IIT");
```

Each of these statements has three parts:

Declaration: The code Student student; declarations that associate a variable name with an object type.

Instantiation: The new keyword is a Java operator that creates the object.

Initialization: The new operator is followed by a call to a constructor, which initializes the new object.

CALLING OBJECTS USING METHODS

Now, we have the complete program below which creates three Student objects and invokes the printDetails() method on each of these objects.

```
public class StudentTest {

public void printdetails(String name,String college)

{
        System.out.println("Name of student is: "+ name + "\n");

        System.out.println("Name of college student is studying in: "+ college + "\n");

}

 public static void main ( String[] args ) {

        Student s1 = new Student ("Ramesh","BVB") ;

Student s2 = new Student ( "Prakash", "GEC" );

 Student s3 = new Student ( "Pramod" , "IIT");

System.out.println("Student s1:\n ");

  s1.printDetails();

  System.out.println("\nStudent s2: ");

  s2.printDetails();

  System.out.println("\nStudent s3: ");

  s3.printDetails(); }
```

When we run this program, we get the following output:

```
 Student s1:

Name of  student is: Ramesh

Name of college student is studying in is:  BVB


Student s2:

Name of  student is: Prakash

Name of college student is studying in is:  GEC


Student s3:
Name of  student is: Pramod

Name of college student is studying in is:  IIT
```

Now that we have created the objects, we will look into how the variables and methods of these objects can be accessed. In order to access a variable or a method, we use the reference/dot operator (.) in the following way:

```
s1.printDetails();
```

The above statement invokes the printDetails() method on the Student object s1. If the method requires arguments, we state them within the parentheses. Variables too are accessed in a similar way, except that the parentheses are not included. This is what separates the variables from methods and helps us to distinguish between the two. If the name variable was public we could have used the following statement to assign a String to the name variables of s1 in the following way:

```
s1.name="Ram";
```

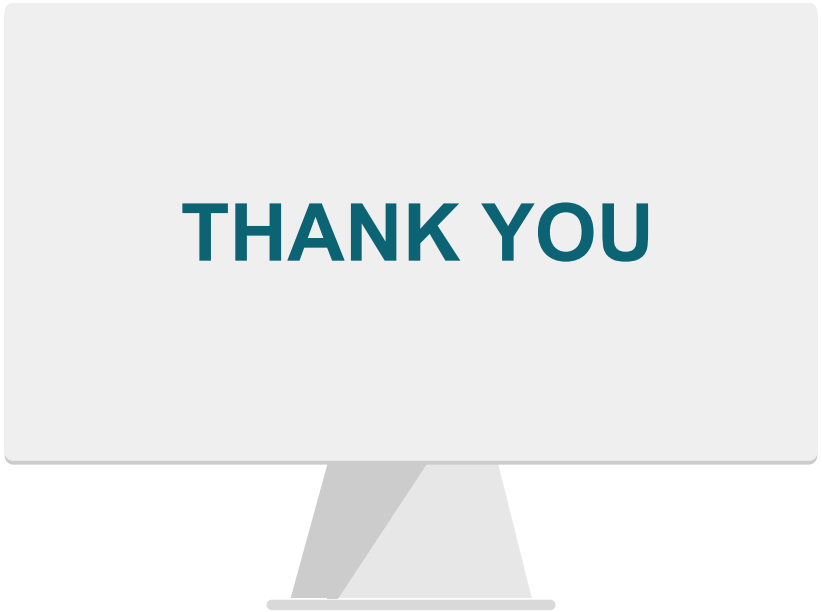
We can also print these values, as we would have printed a String variable if the variable was declared as public.

```
System.out.println("Name is "+ s1.name);
```

But, for the current objects such access is denied since the variables were declared to be private which means that the variables are not accessible outside the class.

DIFFERENCE BETWEEN OBJECT ORIENTED PROGRAMMING AND PROCEDURAL PROGRAMMING

Object Based Programming	Procedure-Based Programming
Focuses on data	Focuses on doing things(functions)
Follows bottom-up approach	Follows top-down approach
Data abstraction and data hiding feature prevents accidental change in data	Use of global variables puts the data at risk
Extending the program is easier than procedure-based programming	Extending the program is a difficult task


THANK YOU