Chapter 3: Concept of Object-Oriented Programming (OOP), Objects and Classes

The topics:

1. Object Oriented Programming

1.1 Object Oriented Programming

It takes the view that programs should start thinking about the data first. After all the primary purpose of computing is the result, not the computing procedure. So here the programmer identifies data objects and their relationships.

The development system based on the object-oriented paradigm is a collection of entities, called objects, that interact together to accomplish certain objectives. The following steps are to be followed to develop an object-oriented program:-

1. **Object Oriented Analysis (OOA)** - It refers to the methods of specifying requirements of the software in terms of a real-world objects, their behaviors and their interaction with each other.
2. **Object Oriented Design (OOD)** - It refers to specification of objects and derived class hierarchies from which objects can be created.
3. **Object Oriented Programming (OOP)** - It refers to the implementation of the program using objects in OOP language.

1.2 Features of OOPS

- **Class & Object** - A class is a group of entities that share similar behavior and characteristics. A class can be defined as a **template** or **blueprint** that represents characteristics and behavior of similar type of objects. An object is an instance of that class.
- **Encapsulation** - A class combines data and functions into a single programming unit, called as encapsulation. Thus, the data is hidden and is safe from accidental alteration. The object's function provides the only way to access the data.
- **Abstraction** - Process of featuring the essential parts and hiding non-essential parts of the class to the outside world. Abstraction leads to data hiding that provides additional security feature to the program.
- **Inheritance** - Inheritance is the process of creating new classes, called derived classes, from existing classes is the base class. The derived class inherits all the capabilities of the base class but can add properties of its own.
  Inheritance has two advantages.
  1. Code reusability - Adding features and capabilities to a class without modification.
  2. Abstraction - The derived class can add data and function members to the class without affecting the base class behavior. This is called data abstraction.
- **Polymorphism** - Using same operator or function in different ways depending on what they are operating on.
  - **Function Overloading** : The process of using the same name but different signature for more than one function. The number, type or sequence of parameter that the function uses are called function signature.
  - **Operator Overloading** : The process of giving ability to an existing operator to operate on a new data type.

1.3 Advantages of OOP:-

- The concept of a data class makes it possible to define subclasses of data objects that share some or all of the main class characteristics, called inheritance. These properties of OOP force a more thorough data analysis, reduce development time, and ensure more accurate coding.
- Data hiding is possible because an object and its modules will know only about the data they need to know about. Because other objects and methods in the application program cannot be accessed without rewriting the object, the possibilities of accidental or unintended data corruption that are possible in procedural programs are not possible with OOP.

1.4 Disadvantages of OOP:-

Object Oriented Programming has several disadvantages which made it unpopular in the early years.

- ***Size***: Object Oriented programs are much larger than other programs. In the early days of computing, space on hard drives, floppy drives and in memory was at a premium. Today we do not have these restrictions.
- ***Effort***: Object Oriented programs require a lot of work to create. Specifically, a great deal of planning goes into an object-oriented program well before a single piece of code is ever written. Initially, this early effort was felt by many to be a waste of time. In addition, because the programs were larger (see above) coders spent more time actually writing the program.
- ***Speed***: Object Oriented programs are slower than other programs, partially because of their size. Other aspects of Object-Oriented Programs also demand more system resources, thus slowing the program down.

2. Procedure Oriented Programming vs Object oriented Programming

2.1 Procedure-oriented programming

It is basically consists of writing a list of instructions for the computer to follow, and organizing these instructions into groups known as functions.

2.2 Characteristics exhibited by POP

(i) It emphasis on algorithms
(ii) large programs are divided into smaller programs
(iii) the functions share global data
(iv) data move openly around the system
(v) function transform data from one form to another
(vi) it employs top-down approach in the program design

2.3 Object-oriented programming

It treats data as a critical element in the program and does not allow it to flow freely around the system. It ties data more closely to the functions that operate on the data and protect it from accidental alteration.
Object-oriented programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.

2.4 Characteristics exhibited by OOP

(i) It emphasis on data rather than procedure
(ii) programs are divided into objects
(iii) data is hidden and cannot be accessed by external functions
(iv) new data and function can be easily added whenever necessary
(v) it follows bottom-up approach in the program design.

3. Class and Object

We can define a class as –

- It is a blue print or prototype for creating similar objects
- It may be regarded as an object factory that creates similar objects
- It allows us to create user defined data types.
- It is also known as composite data type. (Class is heterogenous data structure)
- It defines the type of data an object stores and operations that can act on that data.
- The fundamental idea behind an object-oriented language is to combine into a single unit both data and the functions that operate on the data. Such a unit is called an object.

3.1 What is class?

A class is a group or community that share common attributes and common properties. In terms of programming language, class is a template that has some member variables (known as data members, also known as instance variables) and behaviors (known as member methods, also known as instance methods).

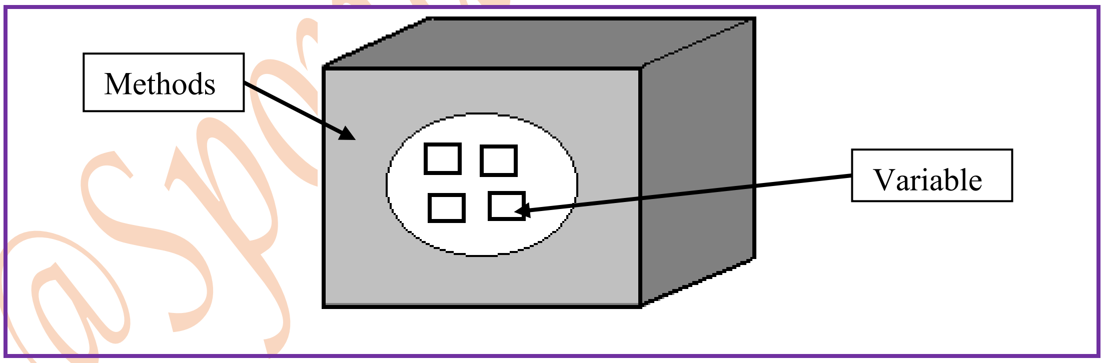
3.2 What is object?

An object is an instance of a particular class. Every object of a particular class has same set of properties. In terms of programming language, an object is a software bundle of variables and related methods defined in a particular class.

3.3 What is the relationship between class and its object?

A class is a collection of similar objects and an object is an instance of a particular class. There is no existence of a class by its own. Object makes the existence of a class in real. Thus, we can say that a class is a concept where as an object of that class is real.

4. Visual Representation of Class and Object

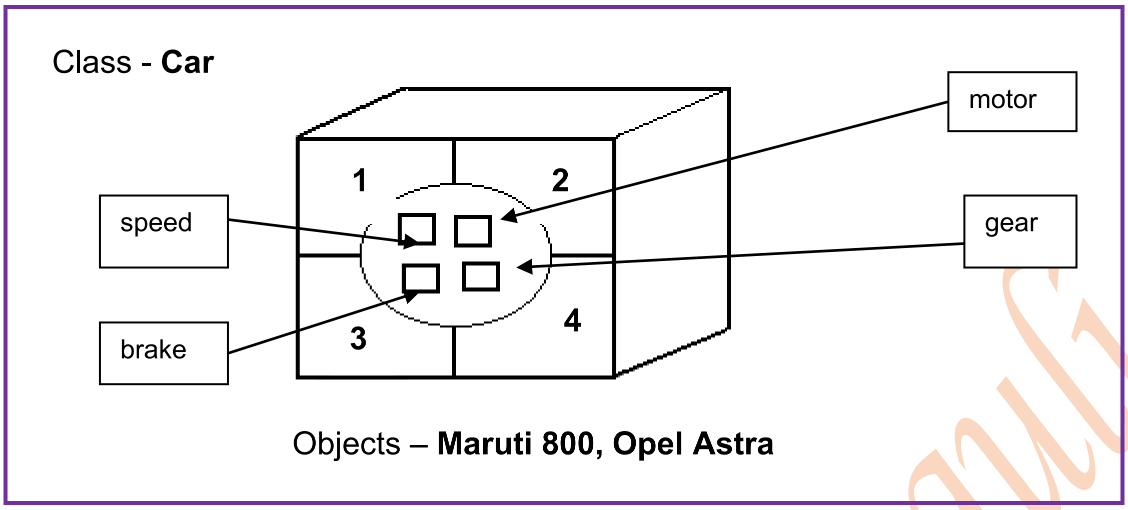


4.1 Representation of Class and its object in real world

For example, if we consider Car as a class, then the structure of a car (like motor, gear, speed, brake will be the state and starting_the_car, changing_gear, increasing_speed, stopping_the_car will be the

behavior). Now any car (like Maruti 800, Opel Astra etc) will be the object of the class Car.
Now we can see that all the types of car (objects) mentioned above share the same states and behaviors and the only difference in their names. Thus, we can say that all the objects derived from a class share similar properties.

4.2 Visual representation of the class – CAR and its object is as follows-



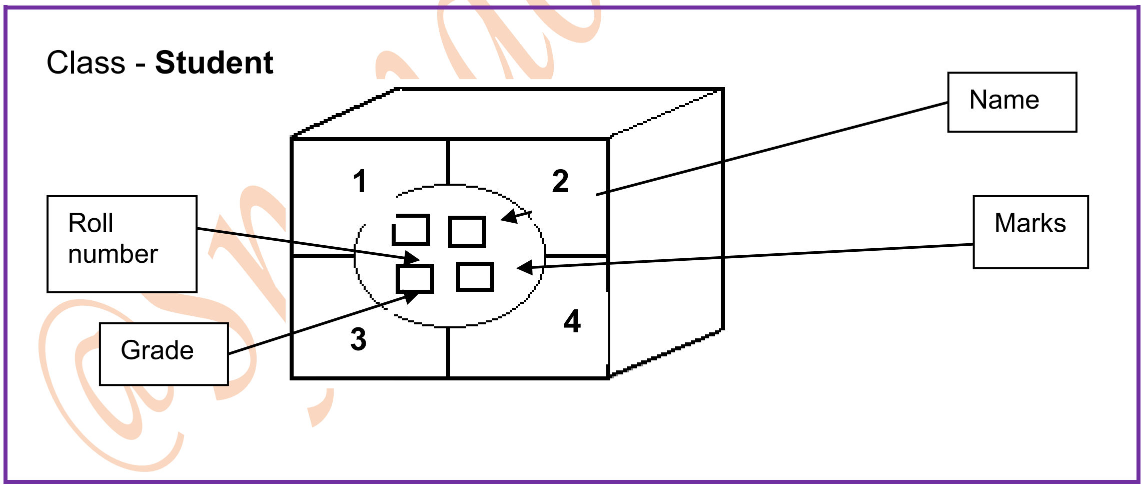
Methods:

1. starting_the_car 2. changing_the_gear
3. increasing_speed 4. stopping_the_car

4.3 The visual representation of the class – STUDENT and its object

Similarly, if we consider Student as a class, then Roll_Number, Name, Marks, Grade are the variables and accept roll number and name, accept marks, calculate grade, display result becomes the methods. Now every student of MCKV will be the object of the class Student.

Visual representation of the above class (data members and member methods) is as follows-



Methods:

1. Accept_Roll_Name 2. Accept_Marks
3. Calculate_Grade 4. Display_Result

5. Features of Object Oriented Programming

5.1 Data Abstraction and Encapsulation

The wrapping up of data and functions into single unit is **encapsulation**. It provides a assurance that the data is not accessible to the outside world and only the function wrapped in the class can access it. This insulation of the data from direct access by the program is data hiding or information hiding.
Abstraction refers to the act of representing essential features without including the background details or explanations. Since, the classes use the concept of data abstraction, they are also known as **Abstract Data Types (ADT)**.

5.2 Inheritance

It is the process by which object of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. Inheritance saves time, effort and unnecessary writing of the features of a class in its derived classes.
Different forms of inheritance

1.	Single Inheritance	-	a derived class with only one base class
2.	Multiple Inheritance	-	one derived class with several base classes
3.	Hierarchical Inheritance	-	the traits of one class may be inherited by more than one child class
4.	Multi-level Inheritance	-	deriving a class from another derived class.
5.	Hybrid Inheritance	-	a situation where we need to apply two or more types of inheritance in the program

A private member of a class cannot be inherited either in public or in private mode. A protected member inherited in public becomes protected and in private becomes private in derived class. A public member inherited in public mode becomes public whereas inherited private becomes private in derived class.

Access Specifier: Keyword that allows the access to the members of a class from within and outside the class.

- Private: These members will be accessed only by the members of the class itself and not by anyone.
- Protected: These members will be accessed by the members of the class and it's child classes. For others (outsiders) it acts as private access specifier
- Public: These members will be accessed by everyone.
- default: (When no access specifier is mentioned, then complier provide default access specifier that acts as public access specifier)

5.3 Polymorphism

Polymorphism means ability to take more than one form. An operation may exhibits different behavior in different instances and the behavior depends upon the types of data used in the operation.
Polymorphism plays an important role in allowing objects having different internal structure to share the same external interface.

5.4 Dynamic Binding

It means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism. Binding refers to the linking of a procedure call to the code to be executed in response to the call.

5.5 Modularity

It is the act of partitioning a program into individual components. It reduces the complexity of the program to some degrees; it also creates a number of well-defined, documented boundaries with the program. A module is a separate unit in itself and it can be compiled separately though it has connections with other modules.

Advantages of Modular programming:-

1. **Complexity:** By sub-dividing the problem into as number of modules, we can hide the information not needed outside the module, and then only concentrate on the specifics of the module inside. It this is still complex, then sub-module can be created further.
2. **Duplication :** Avoiding duplicate code is a reason for modules. It is easier to maintain, save space and reduce complexity of a program.
3. **Debugging :** By isolating areas, debugging becomes easier and reliable because we are **working** with smaller section of the program.
4. **Reusability :** Modules can be used in other program without rewriting or retesting.
5. **Portability :** By isolating non-portable sections of code in modules we can make it easier to move the program over to other machines. Such sections include hardware dependencies and operating system dependencies.

6. Creating a Class in Java

6.1 How to create class in Java?

Every program we need to create in Java must be written inside a class. For this first of all we need to determine the classes, their data members, member methods and objects involved in the problem domain.

After determining all the elements required in the class, we need to declare the class using the keyword class. To declare a class in Java, following structure is needed:

```
<access specifier> class <class name>
{
        //data members of the class
        ….

        //member methods of the class
        ….
}
```

Java provides four types of access specifiers, they are –

- public
- private
- protected
- default (no specifier mentioned)

public classes, methods and fields can be accessed from everywhere within the same package or from other packages in Java. The only constraint is that a file with Java source code can only contain one public class whose name must also match with the filename.
protected methods and fields can only be accessed within the same class to which they belong, within the subclasses of that class and within the classes of the same package. [generally, classes are not declared as protected]
private methods and fields are accessed with in the same class to which they belong and nowhere else not even in the subclasses. [classes are not declared as private since this will make the class inaccessible].
If no specifier is mentioned with the class or with its methods and fields, then it will be considered as default access specifier. In this case, the class and its members are accessible from everywhere except form different package.

Mostly it is recommended to declare the class as public, fields (data members) as private and methods as public (some of the methods can be declared as private if they are to required to be called from outside the class, similarly if the class is to be inherited by another class, then the data members must be declared as protected). This will provide the security of the members of the class from outside access and will be safe from accidental alteration or from unauthorized access. Only the required methods will be accessed from outside the class.

6.2 How to declare the data members (attributes) of a class?

The data members of a class are the attributes of an object of the same class. A field (data member) with a class must be declared in the following way:

```
<access specifier> <data type> name_of_the_field;
```

6.3 How to declare the methods of a class?

The member methods declared in a class are the behavior of an object of the same class. A method should be declared in the class following the given structure:

```
<access specifier> <modifier(if reqd)> <return_type> method_name(parameter list)
{
        //body of the method
        ……
        ……
}
```

- **access specifier** – can be either **public** or **protected** or **private** or default (no access specifier mentioned). These keywords are used to determine the access type of the methods.
- **modifier** – can be one of: **final**, **native**, **synchronized**, **transient**, **volatile**. A **final** method means that the functionality defined inside this method can never be changed.
- **return type –** this specifies the return type of the method, i.e. the type of the value this method will return to the caller one. It may be any valid data type depending upon the type of the value the method will return. If the method is not returning any value, then it should be declared as **void**.
- **method name** – It should be a valid identifier. While naming a method, we should follow the identifier forming rules and method naming convention.
  Link: Identifier forming rules and method naming conventions
- **parameter list** – the parameter list is a comma-separated list of variables that will receive values from the caller method at the time of method call. These lists of variables are referred as arguments or parameters of a method. A method can be declared without any parameters and in that case the parameter list must be kept empty.

Now let us have a look in to different class declarations along with their fields and methods.

```
public class Student {
        //data members or fields of the class
                public String name;
                public int roll;
                public double marks;
                public char grade;
        //member methods of the class
                public void acceptData(String nm, int r, double m)
                {
                        name = nm;
                        roll = r;
                        marks = m;
                }
                public void computeGrade( )
                {
                        ……
                }
                public void display( )
                {
                        ……
                }
        }//closing of class Student
```

7. Declaration & Instantiation of Objects

After the class declaration is complete with all its details (data members and member methods), now it is the time to declare an object of the class and to call the methods for execution. Objects of a class must be declared inside a static method of that class (generally, main method is used for this purpose).

In Java, the new keyword is used to declare and to instantiate an object of a class. The new operator creates the object in the memory and returns a reference to the newly created object. This object will remain in the memory as long as our program retains a reference to the object. The following statement should be used for declaration of an object of the above mentioned class Student :

Student s = new Student( );

here, Student is the class, s is a reference to the class Student and "= new Student();" is the statement that will creates the object of type Student by allocating memory for it and returns its reference to 's'.

So the structure of the main method to run the program will be as follows:

```
public static void main(String args[])
{
        Student s = new Student( );
        s.acceptData("Rahul", 11, 567.0);
        s.computeGrade( );
        s.display( );
}
```

This main method can be declared inside the same class or in a separate class under a separate filename.

The object declaration can be done along with initialization of the data members. As we know that a class provides template for objects. From this template, objects are created using the **new** operator along with the class constructor. The following statement creates an object of Student using class constructor to assign the values for the fields:

Student **stud** = new **Student("Rahul",11,567.00)**;

here **stud** is the object of Student class and **Student("Rahul",11,567.00)** is the constructor with arguments for the data members of the object stud.

Each object declaration statement has three parts:

1. **Declaration** – The code on the left-hand side of the = operator is the variable declaration that associate a name with a type.
2. **Instantiation** – The new operator in Java creates a new object and allocate memory space for that.
3. **Initialization** – After the new operator, the call to the constructor of the class assigns values to the object's fields. If no parameters are passed in that case fields are assigned with the default values of the respective data types.

8. Constructor

8.1 A constructor

A constructor is a special member method of a class that has the same name of the class itself. It is called at the time of object creation to initialize the data members of the class. As the constructor has the same name of the class, it is called implicitly at the time of the object instantiation. The primary job of the constructor is to initialize the object to a legal initial value for the class.

8.2 Properties of a constructor

- A constructor has the same name of the class.
- A constructor has no return type, not even void.
- A constructor must be declared with public access specifier so that it can be accessed outside the class also.
- A class can have more that one constructor but with different function signature (this feature is also known as constructor overloading).
- Constructors can not be called explicitly (only called at the time of object creation).

Let us consider an example – a class having a constructor

```
public class Student {
        //data members or fields of the class
                public String name;
                public int roll;
                public double marks;

                public Student( ) //constructor without any parameters
                {
                        name = "Sagnik";
                        roll = 12;
                        marks = 560.00;
                }

                public Student(String nm, int r, double m) //constructor with parameters
                {
                        name = nm;
                        roll = r;
                        marks = m;
                }
        }
```

Now if we create an object using class constructor with empty parameters then it will look like the following:

Student **s1** = new **Student( )**; here the object s1 will acquire following values for its data members :

s1.name = Sagnik
s1.roll = 12
s1. marks = 560.00

Now if we create an object using class constructor with parameters then it will look like the following:

Student **s2** = new **Student("Sourav",14,575.00)**; here the object s2 will acquire following values for its data members :

s2.name = Sourav
s2.roll = 14
s2. marks = 575.00

8.3 Types of Constructor

In Java, constructors are mainly of two types – one which receives parameters (**parameterized** constructor) and another which can not receives parameters (**non-parameterized** constructor).

1. **Parameterized** constructors are those which receive values through the parameters mentioned in the constructor to initialize the data members at the time of object creation.
2. **Non-parameterized** constructors are those which can not receive any values through the parameters but have a set of legal values within the constructor itself to initialize the data members at the time of object creation.

Parameterized constructors can be of two types – one with parameters of primitive data types and another with object as parameters. Parameterized constructor with primitive data types as parameters initialize the fields of an object with primitive values and parameterized constructor with object as parameter initialize the fields with the values of the fields of the parameter object (such type of constructors is also known as copy constructor).

Default constructor – normally non-parameterized constructors are some times termed as default constructor but actually a constructor that initializes the data members of an object with their default values are known as default constructor. Also, if no constructor is declared within the class then at the time of object creation Java compiler provides the default constructor to initialize the data members with their default values.

Let us consider the example given below:

```
public class Time {
                public int hour;
                public int minute;
                public int second;
                public Time( ) //constructor initializing the data members with default value
                {
                        hour=minute=second=0;
                }
                public Time (int hr) //constructor with one parameter
                {
                        hour = hr;
                        minute = second = 0;
                }
                public Time (int hr, int min) //constructor with two parameters
                {
                        hour = hr;
                        minute = min;
                        second = 0;
                }
                public Time (int hr, int min, int sec) //constructor with three parameters
                {
                        hour = hr;
                        minute = min;
                        second = sec;
                }
                public Time (Time obj) //constructor with object as parameters
                {
                        hour = obj.hr;
                        minute = obj.min;
                        second = obj.sec;
                }
        }
```

Now if we create objects using above class constructors then it will look like the following:-
Time **T1** = new **Time( )**; here the object T1 will have the time **0 hour 0 minute 0 second**
Time **T2** = new **Time(10,22,45)**; here the object T2 will have the time **10 hour 22 minute 45 second**
Time **T3** = new **Time(T2)**; here the object T3 will also have the time **10 hour 22 minute 45 second**

9. Class Members vs Instance Members

9.1 Class Members

We have seen till now that all the fields and the methods declared inside the class becomes the fields and the methods of all the instances (objects) created from that class. However, there are situations where it would be we need some fields that would not be associated with the instances of a.
Sometimes, a single global-type fields or methods is needed that can be shared by all the instances of the class. These types of members are called as **class members**.

9.2 How to declare a class member?

To declare a class member, we use the keyword '**static**' with the declaration. The keyword 'static' makes the declaration as single global type member of the class. These members are directly associated with the class and not with any instances of the class. These members are also called as static members of the class. This is because, there exists only on copy of these members in the memory and they will be shared by all the instances created from the class.

9.3 Differences between class members and instance members

Class members	*Instance members*
1. Class members are declared with the keyword 'static' and thus there exists only one copy of them in the memory.	1. Instance members are declared without 'static' keyword and thus they exist for every instance separately in the memory.
2. Class members are created at compile time and thus they can be accessed without any existence of an instance of the class.	2. Instance members are created at the time of object creation or instantiation thus their access depends upon the existence of the instance.
3. Class members cannot access any instance members without the existence of an instance.	4. Instance members can access the class members directly.

POINTS TO SUMMARY

Q1. What are the features of OOPS

- **Class & Object**- A class is a group of entities that share similar behavior and characteristics. A class can be defined as a template or blueprint that represents characteristics and behavior of similar type of objects. An object is an instance of that class.
- **Encapsulation** - A class combines data and functions into a single programming unit, called as encapsulation. Thus the data is hidden and is safe from accidental alteration. The object's function provides the only way to access the data.
- **Inheritance** - Inheritance is the process of creating new classes, called derived classes, from existing classes is the base class. The derived class inherits all the capabilities of the base class but can add properties of its own.
  Inheritance has two advantages.
  1. Code reusability - Adding features and capabilities to a class without modification.
  2. Abstraction - The derived class can add data and function members to the class without affecting the base class behavior. This is called data abstraction.
- **Polymorphism** - Using same operator or function in different ways depending on what they are operating on.

Q2. What is class?

A class is a group or community that share common attributes and common properties. In terms of programming language, class is a template that has some member variables (known as data members, also known as instance variables) and behaviors (known as member methods, also known as instance methods).

Q3. What is object?

An object is an instance of a particular class. Every object of a particular class has same set of properties. In terms of programming language, an object is a software bundle of variables and related methods defined in a particular class.

Q4. What is the relationship between class and its object?

A class is a collection of similar objects and an object is an instance of a particular class. There is no existence of a class by its own. Object makes the existence of a class in real. Thus we can say that a class is a concept where as an object of that class is real.

Q5. What is a constructor?

A constructor is a special member method of a class that has the same name of the class itself. It is called at the time of object creation to initialize the data members of the class. As the constructor has the same name of the class, it is called implicitly at the time of the object instantiation. The primary job of the constructor is to initialize the object to a legal initial value for the class.

Q6. Mention some properties of a constructor:

- A constructor has the same name of the class.
- A constructor has no return type, not even void.
- A constructor must be declared with public access specifier so that it canbe accessed outside the class also.
- A class can have more that one constructor but with different function signature (this feature is also known as constructor overloading).
- Constructors can not be called explicitly (only called at the time of object creation).

Q7. What are the different types of constructor?

In Java, constructors are mainly of two types – one which receives parameters (**parameterized** constructor) and another which can not receives parameters (**non-parameterized** constructor).

3. **Parameterized** constructors are those which receive values through the parameters mentioned in the constructor to initialize the data members at the time of object creation.
4. **Non-parameterized** constructors are those which can not receive any values through the parameters but have a set of legal values within the constructor itself to initialize the data members at the time of object creation.

Solved questions

1. What is an object? Give examples of some real world objects.

Ans. An object is some identifiable entity with some characteristics and behavior. Examples of real world objects are human being, Car, House etc.

2. "An object's state and behavior are two different things yet linked to one another'. Comment.

Ans. For instance we consider an Orange is an object. Its characteristics are : it is spherical in shape , its color is orange etc. Its behaviour is : it is citrus in nature, it tastes sweet and sour. Therefore we can find that the behaviour depends upon the characteristic and they are interwoven.

3. Define data abstraction.

Ans. It is the act of representing essential features without including the background details or explanations.

4. Define Encapsulation.

Ans. Encapsulation is the way to implement data hiding and abstraction by wrapping up data and associated functions into single unit called object.

5. What is message passing among objects?

Ans. It is a way of sending or receiving information to/from another object.

6. What are methods? What do they represent?

Ans. The behaviour of the object is depicted through functions called methods. It is the way to access data in an object.

7. What is information hiding?

Ans. It is the process of hiding all secrets of an object that do not contribute to its essential characteristics; the structure of an object is hidden as well as the implementation of its method.

8. How are real world objects implemented / represented in software terms?

Ans. The object is implemented in software terms as follows:

a) Characteristics / attributes are implemented through member variables or data items of the object.
b) Behaviour is implemented through member functions called methods.
c) Data and methods are encapsulated into one unit and given a unique name to give it identity.

9. How is data abstraction associated with encapsulation?

Ans. Encapsulation is the act of combining both data and the functions that operate on the data under a single unit. Encapsulation, in a way implements abstraction.

10. 'There can be many abstraction of an object'. Comment.

Ans. For instance if we talk of a student, we can talk of anything that belongs to him in the real world like his name rollno, school, address etc. But when we talk about student result tracking system, then the abstraction would be – rollno, name, marks obtained, etc. For extracurricular activities, the abstraction would be --- his rollno, talents, awards, etc. So there can be many abstraction of an object.

11. How does object encapsulate its state and behaviour ?

Ans. Since state and behaviour of objects are interwoven, they are said to encapsulate state and behaviour . For instance, a car object has characteristics like number of wheels, seats, make etc. Its state is represented as halted, moving, or stationary. And its behaviour is : it can move, stop, blow horn etc. Now all these things are wrapped up together in the form of car. We cannot segregate them. Thus we can say that objects encapsulate their state and behaviour as their state and behaviour are interlinked, they cannot exist separately.

12. How do objects interact with each other? Give example.

Ans. When the objects need to interact with each other, they pass/ request information to one another. This interaction is known as message passing.
For example in a big company there are so many departments, ASales, Accounts, and Purchase etc. Here one department send message to other department to retrieve some information.

13. Write down some differences between class members and instance members.

Class members	Instance members
Declared with static keyword	Declared without static keyword
Exist at class level and can be used even if no instance of class level exist in memory.	Exist at instance level and cannot be used if there are no instance of class in memory.
Created when the class is first referred to.	Created with each instance.
Destroyed when the program in over.	Destroyed when the instance containing them is destroyed.
All instances of class share the same copy.	All instances have individual copy.

MCQ with answers:

1. Which of the following approach help us understand better about Real time examples, say Vehicle or Employee of an Organisation?
a) Procedural approach
b) Object Oriented approach
c) Both a and b
d) None of the mentioned

Answer: b
Explanation: Object Oriented Programming supports the concept of class and object which help us understand the real time examples better. Vehicle is defined to be a class. Car, truck, bike is defined to be objects of the class, Vehicle.

2. Which of the following Paradigm is followed by Object Oriented Language Design?
a) Process-Oriented Model
b) Data Controlling access to code.
c) Both a and b
d) None of the mentioned

Answer: b
Explanation: Object-oriented programming organizes a program around its data(that is, objects) and a set of well-defined interfaces to that data.

3. Which of the following approach is followed by Object Oriented Language during the execution of a program?
a) Bottom-up approach
b) Top-down approach
c) Both a and b
d) None of the mentioned

Answer: a
Explanation: Bottom-up approach begins with details and moves to higher conceptual level there by reducing complexity and making the concept easier to understand.

4. Which of the following is/are advantage of using object-oriented programming?
a) Code Reusability
b) Can create more than one instance of a class without interference
c) Platform independent
d) All of the mentioned

Answer: d

5. Java Compiler translates the source code into?
a) Machine code
b) Assembly code
c) Byte code
d) JVM code

Answer: c
Explanation: Java program is converted into 'byte code' which makes it easier to run on wide variety of environments. Only the run-time package JVM has to be implemented for each platform.

6. What is the output of this program?
```
class PrintTest {
   public static void main(String args[])
   {
      int num = 10;
      if (NUM < 100) {
         System.out.println("The value of num is"+ num);
      }
   }
}
```
a) Compilation error
b) Run time error
c) The value of num is 10
d) None of the mentioned

Answer: a
Explanation: Java is case-sensitive. The variable ‘num’ used in print statement is not same as the variable ‘NUM’ used in the if conditional statement.

7. Which of the following is called as ‘Compilation unit’?
a) Object file
b) Byte code
c) Source file
d) All of the mentioned

Answer: c

8. What is the output of this program?
```
class TypeChecking {
   public static void main(String args[])
   {
      int num = 10.5;
      System.out.println("Output :The value of num is" +num);
   }
}
```
a) Output: The value of num is 10.5
b) Run-time error
c) Compilation error
d) None of the mentioned

Answer: c
Explanation: Java is strongly typed language. The variable num is declared as ‘int’ but the value assigned to it is ‘float’, as a result the code does not get compiled.

9. Which is a named memory location assigned a value by the program?
a) variable
b) literal
c) identifier
d) None of the mentioned

Answer: a

Unsolved questions (Short)

Q1. Define object with respect to real world.

Q2. What is the difference between state and behavior of an object?

Q3. Define Data abstraction

Q4. What is Encapsulation?

Q5. How is data abstraction associated with encapsulation?

Q6. How do objects interact with one another? Give an example.

Q7. How do objects encapsulate state and behavior?

Q8. How are objects implemented in software terms?

Q9. State two significance of data abstraction in program

Q10. “Class has no existence of its own” – Explain.

Unsolved questions (Long)

1. What is a class? Explain with an example
2. What is an object? Explain with an example
3. What is a method? Explain with an example
4. What is the need of a class? Explain with an example
5. What is Encapsulation? Explain with an example
6. Define object with respect to real world.
7. Define Data abstraction. Explain with an example
8. How is data abstraction associated with encapsulation?
9. How do objects interact with one another? Give an example.
10. How do objects encapsulate state and behavior?
11. How are objects implemented in software terms?
12. How can you say that a class is an object factory?
13. How do you map an abstraction into software?
14. What is the difference between state and behavior of an object?
15. Why are methods so important for the description of objects?
16. State two significance of data abstraction in program
17. “Class has no existence of its own” – Explain.
18. “Classes as abstraction of sets of objects” – Explain
19. Can there be multiple abstractions of a real world entity?
20. Explain the concept of multiple classes in Java.