TOP 30

OOPs

INTERVIEW QUESTION

Q 1. What is Object Oriented Programming (OOPs)?

Ans : Object Oriented Programming (also known as OOPs) is a programming paradigm where the complete software operates as a bunch of objects talking to each other.

An object is a collection of data and the methods which operate on that data.

Q 2. Why OOPs?

Ans :

The main advantage of OOP is better manageable code that covers the following:

1) The overall understanding of the software is increased as the distance between the language spoken by developers and that spoken by users.
2) Object orientation eases maintenance by the use of encapsulation. One can easily change the underlying representation by keeping the methods the same.
3) The OOPs paradigm is mainly useful for relatively big software.

Q 3. What is a Class?

Ans : A **class** is a building block of Object Oriented Programs. It is a user-defined data type that contains the data members and member functions that operate on the data members. It is like a blueprint or template of objects having common properties and methods.

Q 4. What is an Object?

Ans : An **object** is an instance of a class.

Data members and methods of a class cannot be used directly.

We need to create an object (or instance) of the class to use them.

In simple terms, they are the actual world entities that have a state and behavior.

Eg. The code below shows is an example in C++ of how an instance of a class (i.e an object) of a class is created

```
#include <iostream>
using namespace std;

class Student{
private:
        string name;
        string surname;
        int rollNo;
public:
        Student(string studentName, string studentSurname, int
studentRollNo){
                name = studentName;
                surname = studentSurname;
                rollNo = studentRollNo;
        }
        void getStudentDetails(){
                cout <<"The name of the student is "<< name
```

```
<<""<< surname << endl;
                cout <<"The roll no of the student is "<< rollNo <<
endl;
        }
};
int main() {
        Student student1("Vivek", "Yadav", 20);
        student1.getStudentDetails();
        return 0;
}
```

Output :

```
The name of the student is Vivek Yadav
The roll no of the student is 20
```

Q 5. What are the main features of OOPs?

Ans : The main feature of the OOPs, also known as 4 pillars or basic principles of OOPs are as follows:

- Encapsulation
- Data Abstraction
- Polymorphism
- Inheritance

Q 6. What is Encapsulation?

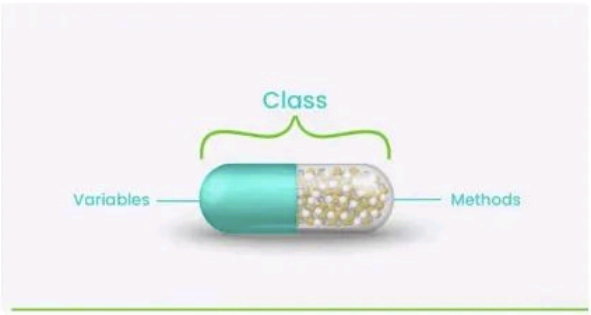
Ans : Encapsulation is the binding of data and methods that manipulate them into a single unit such that the sensitive data is hidden from the users

It is implemented as the processes mentioned below:

1) **Data hiding:** A language feature to restrict access to members of an object. For example, private and protected members in C++.

2) **Bundling of data and methods together:** Data and methods that operate on that data are bundled together.

For example, the data members and member methods that operate on them are wrapped into a single unit known as a class.

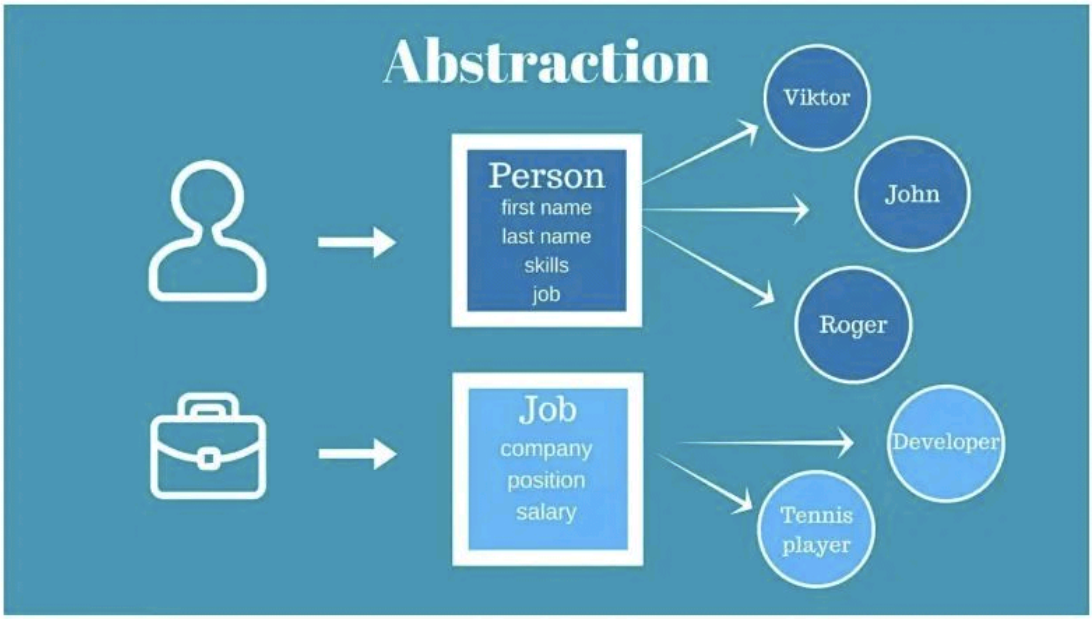


Q 7. What is Abstraction?

Ans : Abstraction is similar to data encapsulation and is very important in OOP.

It means showing only the necessary information and hiding the other irrelevant information from the user.

Abstraction is implemented using classes and interfaces.



Q 8. What is Polymorphism?

Ans : The word "**Polymorphism**" means having many forms.

It is the property of some code to behave differently for different contexts.

For example, in C++ language, we can define multiple functions having the same name but different working depending on the context.

Polymorphism can be classified into two types based on the time when the call to the object or function is resolved. They are as follows:

A. Compile Time Polymorphism
B. Runtime Polymorphism

A) Compile-Time Polymorphism

Compile time polymorphism, also known as static polymorphism or early binding is the type of polymorphism where the binding of the call to its code is done at the compile time.

Method overloading or operator overloading are examples of compile-time polymorphism.

B) Runtime Polymorphism

Also known as dynamic polymorphism or late binding, runtime polymorphism is the type of polymorphism where the actual implementation of the function is determined during the runtime or execution. Method overriding is an example of this method.

Q 9. What is Inheritance? What is its purpose?

Ans : The idea of inheritance is simple, a class is derived from another class and uses data and implementation of that other class.

The class which is derived is called child or derived or subclass and the class from which the child class is derived is called parent or base or superclass.

The main purpose of Inheritance is to increase code reusability. It is also used to achieve Runtime Polymorphism.

```
// an example of inheritance
class Student {
public void read() {
        System.out.println("The student is reading");
}
}
class SchoolStudent extends Student {
public void read(String book) {
        System.out.println("the student is reding "+
book);
}
}
```

Q 10. What are access specifiers? What is their significance in OOPs?

Ans : Access specifiers are special types of keywords that are used to specify or control the accessibility of entities like classes, methods, and so on. **Private**, **Public**, and **Protected** are examples of access specifiers or access modifiers.

The key components of OOPs, encapsulation and data hiding, are largely achieved because of these access specifiers.

```
class User {
public String userName;
protected String userEmail;
private String password;
public void setPassword(String password) {
        this.password = password;
}
}
```

```
public class Student {
    // private data members
    private String name;
    private int rollNo;
    // public getter method to access the name
    public String getName() {
        return name;
    }
    // public getter method to access rollNo
    public int getRollNo() {
        return rollNo;
    }
    // public setter method to set name
    public void setName(String name) {
        this.name = name;
    }
    // public setter method to set rollNo
    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }
}
```

```
public class OOPS {
public static void main(String args[]) {
        User user1 = new User();
        user1.userName = "Vivek_Kumar_Yadav";
        user1.setPassword("abcd@12345");
        user1.userEmail = "abc@gmail.com";
}
}
```

Q 11. What are the advantages and disadvantages of OOPs?

Ans :

Advantages of OOPs	Disadvantages of OOPs
OOPs provides enhanced code reusability.	The programmer should be well-skilled and should have excellent thinking in terms of objects as everything is treated as an object in OOPs.
The code is easier to maintain and update.	Proper planning is required because OOPs is a little bit tricky.
It provides better data security by restricting data access and avoiding unnecessary exposure.	OOPs concept is not suitable for all kinds of problems.
Fast to implement and easy to	The length of the programs is much larger