OOPs Interview Questions

Q) What is OOPs & why it is important?

-> OOPs is programming technique that employs Objects rather than just functions & procedures.

-> All objects are grouped into classes in Object Oriented Programming.

-> OOPs integrates real-world concepts into programming such as inheritance, Polymorphism, & abstraction.

Why OOPs is Important

-> OOPs allows more clarity in programming there by allowing simplicity in solving Complex problems.

-> OOPs reduces Redundancy

-> OOPs provides ability to bind both data & code together.

-> Allows in keeping Sensitive data Confidential.

-> OOPs improves code-readability

-> Polymorphism gives flexibility to the programs by allowing the entities to have multiple forms.

Q] What Are classess & Objects.

* class

1] A class is an Object's blueprint or template it is a data type that a User specifies.

2] Inside class we can define Variables, Constants, member functions, etc.

3] A class does not consume memory at the Run-Time.

- Objects are basically instances of class. They can access Variables or methods declared inside the class.

Q] What is Pure OOP? why Java is Not Pure Object Oriented Programming?

Encapsulation, Abstraction, Polymorphism, inheritance, class & Object

Pure Object Oriented Programming language support or have features that treat everything within a program as an Object.

int y;

: Variable Declaration with Primitive data Types.

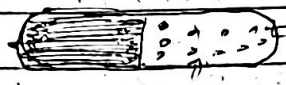
Variables can be stored in Java using primitive data types. In addition, the static keyword in Java allows us to use classes without the use of objects.

Q] What are the Differences Between Inheritance And Polymorphism?

→ Inheritance	Polymorphism
→ with inheritance, a derived class inherits the already existing class's features (base class).	Polymorphism allows class methods to exist in multiple forms.
→ Inheritance refers to using the structure & behavior of a parent class in a subclass	polymorphism intends on changing the behavior of parent class's method.
→ Inheritance can be of single, hybrid, multiple, hierarchical, multipath & multilevel types	Polymorphism has two types: compile time & run time polymorphism.
→ Inheritance supports code reusability & reduces lines of code	Polymorphism decides which form of a function to be invoked.

Q) What is Encapsulation?

Encapsulation is the process of binding data members & methods of a program together to do a specific job, without revealing unnecessary details.



class Binding variables & methods.

Q) What is Abstraction?

→ Abstraction is the method of hiding unnecessary details from the necessary ones. it is one of the main features of OOPs.

Q) Can we Instantiate an Abstract class?

→ we cannot instantiate an abstract class because it is abstract, & it is not complete.

when we add {} parenthesis while creating an object, the compiler considers it as an anonymous class, where {} denotes the body of the anonymous class.

Q) Can a class Inherit the Constructor of it's Base class ?

→ ② public Parent()

parent class	Inheritance →	child class

Whenever a child class extends parent class, the subclass inherits state & behavior in the form of variables & methods from its superclass but it does not inherit Constructor of supper class.

② If Constructors are inherited then child class would Contain a parent class Constructor which is against the Constructor Constraint.

Q) How is an Abstract class Different from an ~~iter~~ interface ?

→ Both Contains only the method declaration & not their implementation.

→ when an interface is implemented, the subclass must define all its methods & provide its implementation.

→ when an abstract class is inherited, the subclass does not need to provide the definition of its abstract method, Until & Unless the subclass is using it.

Page No. 1
Date

Q) what is Constructor chaining?

-> Constructor chaining is a sequence of invoking constructors of the same class. Upon initializing an Object.

new demo(8, 10); // invokes parameters constructor 3.

demo(int x, int y)
{
this(5);
}

Q) what is Singleton class?

-> Singleton class is a class that can have can have only one object at a time. After the first instantiation, the new object also points to the first instance.

-> Singleton class is capable of implementing interfaces, inheriting from other classes & allowing inheritance.

-> Static class on the other hand, cannot inherit its instance members which makes singleton classes more adaptable.

Q] What is the fundamental Difference Between Abstraction & Encapsulation?

~~Abstraction~~ ~~Encapsulation~~

Hiding details of Television Component. Circuitry: Encapsulation

Hiding Everything Except TV Screen & Control panel : Abstraction

-> Abstraction is about express external simplicity & Encapsulation is about hiding internal complexity.

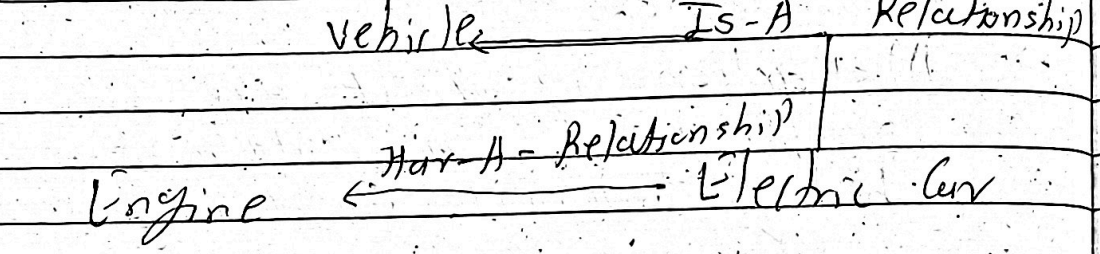
Q] What is Constructor and Destructor? Can we Overload them?

-> Constructor: It is a block of code similar to the method. A constructor will be invoked when an instance of the class is created. At the time of calling the Constructor, memory for an object is allocated in the memory space.

-> Destructor: It is the last function that is called before an object is destroyed.

Q] What is Composition ?

The composition is way to design or implement then has-a relationship. The has-a relationship is used to implement an instance variable that refers to another object.



Q] Copy Constructor Vs Assignment Operator.

-> The Copy Constructor & assignment operator (=) both are used to initialize one object using another object.

-> Copy Constructor allocates separate memory to both objects i.e. existing object & newly created object.

Q] What is Exception Handling ?

-> Exception is an event that occurs during execution of a program at runtime & causes in execution to halt halt.

-> Exception primarily occurs where the User wishes to do something but the program does not support.

-> Exceptions can be handled in the program ahead of time, preventing the execution from stopping.

-> Try-catch is the most common method used for handling exception in the program.

Q] What is Coupling ? Why it's Important ?

-> In Programming, Separation of Concerns is known as Coupling. It means that an object cannot change or modify the state or behavior of other objects directly.

-> Loosely Coupled Objects : These objects are independent of one another & do not directly modify the state of other objects.

Page No. 10
Date

→ Tightly coupled objects:
Objects that depend on other objects & can modify the states of other objects are called tightly coupled.

Q] Explain Overriding & Overloading

→ Method Overloading is a concept in OOP which employs two or more method in class with the same name but different method signature.

→ Q. If a method with the same method signature is presented in both child & parent class is known as method overriding.

Q] Can you override the constructor?

→ You cannot override the constructor because it looks like a method but it is not. it lacks a return type & shares the same name as the class.

→ If you treat constructor as a method & write a super class's constructor in the sub-class, compiler expecting a return type, it will generate a compile-time error.

Q] Which Method will be Executed First: Static or Main Method ?

→ Static block will be executed even before the Compiler or interpreter looks for the main() method in the program. Hence, answer will be Static() method.

Q] What is an Interface ?

→ An interface define a contract for responsibilities (methods) of a class.

→ An interface is a contract: the guy writing the interface says, "hey, I accept things looking that way".

→ Interface represents common actions between multiple classes.

→ eg. Map interface, Collection interface.

Q] What is an Abstract class ?

An abstract class is a class that cannot be instantiated, but must be inherited from. An abstract class may be fully implemented, but is more usually partially implemented or not implemented at all, thereby encapsulating common functionality for inherited class.

Q) When do you use an Abstract class ?

→ If you want to provide common, implemented functionality among all implementations of your component, use an abstract class.
Abstract classes allow you to partially implement your class.

→ An example of an abstract class in the JDK is AbstractMap, which is part of the Collections framework. Its subclasses (which includes HashMap, TreeMap, & ConcurrentHashMap) share many methods (including get, put, isEmpty, containsKey, containsValue) that AbstractMap defines.

→ example abstract method :
public abstract Set<> entrySet();

→ Another Example → Spring AbstractController

In code below "AbstractClassExample ex = new AbstractClassExample();" give a compilation error because AbstractClassExample is declared with keyword abstract.

Q] Compare Abstract class Vs Interface?

Syntactical Difference.

1] Method & members of an abstract class can have any visibility. All method of an interface must be public.

2] A ~~Create~~ Concrete child class of an abstract class must define all the abstract methods. An abstract child class can have abstract method.

An interface extending another interface need not provide default implementation for methods inherited from the parent interface.

3] A child class can only extend a single class. An interface can extend multiple ~~inst.~~ interfaces. A class can implement multiple interfaces.

4] A child class can define abstract methods with same or less restrictive visibility. whereas a class implementation an interface must define all interface method as public.

Q] What is a Constructor?

→ Constructor invoke whenever we create an instance (object) of a class. we cannot create an object without a Constructor.

Constructor has the same name as the class & no return type it can accept any number of parameters.

```
class Animal {
    String name;

public Animal (String name) {
    this.name = name;
}
public static void main (String[]
args ) {

    Animal ani = new Animal ("A");
}
}
```

Q] what is default Constructor ?

→ Default Constructor is the Constructor that is provided by the Compiler. it has no arguments. In the example below,

There are no Constructor define in the Animal class. Compiler provides us with a default Constructor. which helps us create an instance of animal class.

new Topic

1 Habitable Planet

```
public class HabitablePlanetApp {
    private static final String SOLID = "solid";
    private static final int Avg = 15;
    private static final double ERT =
                                0.00001581;
    public static void main (String[] args) {
    ArrayList <Planet> pla = PlanetRepository
                            .getPlanets();
    System.out.println(
            pla.stream().distinct().count());

    Optional <Planet> theBigger =
            pla.stream().max(
            Comparator.comparing (Planet
                        :: getSize));
    System.out.println(
                theBigger.get().getName()
    + " " + theBigger.get().getSize());

    Optional <Planet> theCol =
            pla.stream().min(
    Comparator.comparing (Planet :: getTemper
                            -ature));
    System.out.println(
            theCol.get().getName() + " " +
            theCol.get().getTemperature());
```

System.out.println(" top 10 ");

Planets.pla.stream()
.filter(planet -> SOLD.equal
(planet.getType())
.sorted(Comparator.comparing(
Planet p) ->
Math.abs(p.getTemperature()
- AVE))
.thenComparing(Planet::getDistance-
fromEarth))
.limit(10)
.forEach(System.out::println);

System.out.println(" Earth from the
Sun. ");

pla.stream().sorted(
Comparator.comparing(p -> Math.abs
(p.getclosestStarDistance() -
EARTH))
.limit(1)
.forEach(System.out::println);

}
}

② // public class Planet {

private String name;
private String type;
private int size;
private double closestStarDistance;
private int temperature;
private double distanceFromEarth;

// getter setter constructor
// & hashCode & equal.
// @Override

}

③ public class // Planet Repository {

public static ArrayList<Planet>
getPlanets() {

ArrayList<Planet> planet = new ArrayList();

planet.add(new Planet("Aks", 24, 0.004,
-270, "solid", 1304);

// data.

return planet;

}

}