**write a java code to check if a number is prime?**

Public static boolean isPrime(int n){

if(n <= 1) return false;

for(int i = 2; i \* i < =n ;i++){

If(n % i == 0){

return false;

}

}

return true;

}

**What are method overloading and method overriding?**

**Overloading:**

Overloading occurs when two or more methods in one class have the same method name but different parameters. Like the constructors.

**Overriding:**

Overriding occurs when two methods have the same method name and parameters. One of the methods is in the parent class, and the other is in the child class. Overriding allows a child class to provide the specific implementation of a method that is already present in its parent class.

|  |  |
| --- | --- |
| Overloading | Overriding |
| The binding of overloaded method call to its definition has happens at compile-time | The binding of overridden method call to its definition happens at runtime |
| Static methods can be overloaded which means a class can have more than one static method of the same name | Static methods cannot be overridden |
| Method overloading is performed within class | Method overriding occurs in two classes that have IS-A (inheritance) relationships |
| Argument list should be different while doing method overloading | Argument list should be same in method Overriding |
| Return type of method does not matter in case of method overloading, it can be the same or different | In case of method overriding the overriding method can have more specific return type |

**What is OOP and fundamental concept of OOP?**

**OOP** stands for object oriented programming. It is a type of programming which is based on objects rather than functions and procedures. Each object is an instance of a class. It has four main features which are abstraction, inheritance, encapsulation, and polymorphism.

**Encapsulation:**

Encapsulation refers to wrapping object state(member variables) and behavior(member methods) into a single unit. If you are creating a class, you are doing encapsulation. Encapsulation in java is achieved by using access modifier like private, public, protected and default. If you want to expose some methods to a user, we could use public. If you want to hidden some methods to a user, we could use private.

**Abstraction**:

Abstraction is the concept of hiding the internal implementation and complex logic from a user. You can start a car by turning the key or pressing the start button. You don’t know how the engine is getting started. All the internal processes are hidden from the driver. There are two ways to achieve abstraction. One is abstract class and anther one is interface.

**Inheritance:**

Inheritance allows one class acquires the properties and functionalities of another class. The advantage of Inheritance is providing the reusability of code and each subclass can defines its unique features, the rest of the features can be inherited from the parent class. The disadvantage of Inheritance is two classed are tightly coupled, so we could use composition instead of inheritance to get a loose coupling.

**Polymorphism**:

Polymorphism is the concept where an object behaves differently in different situations. There are two types of polymorphism. One is compile time polymorphism which is achieved by method overloading. Another one is runtime polymorphism which is achieved by method overriding. Example: employee, software, manager.

**Abstraction vs Encapsulation**

Abstraction is more about hiding the details at the design level, Encapsulation is more about hiding the details at implementation level. For example, we have a vehicle, we know it can move, it might move using tiers, it might fly, all we need to know is it can move. Then for encapsulation, for example, when we implement a HashMap, to solve the collision, we either use a linked list, or we use a red-black tree for each bucket. We can change the internal implementation without affecting the clients who use the HashMap.

**Difference between Abstract class and Interface?**

**Abstract class:**

A class that is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body). It needs to be extended by implementing all abstract methods. It CAN NOT be instantiated.

**Interface:**

The interface keyword is used to declare interface. Interfaces can contain only constants and abstract methods.Like abstract classes, Interfaces cannot be instantiated, they can only be implemented by classes or extended by other interfaces.

|  |  |
| --- | --- |
| Abstract class | Interface |
| Abstract class doesn't support multiple inheritance, it can extend only one class or one abstract class at a time | Interface supports multiple inheritance, it can extend any number of interfaces at a time |
| Abstract class can have final, non-final, static and non-static variables with any access specifier | Interface has only static, public and final variables |
| An abstract class can have both abstract and concrete methods | An interface can have only abstract methods. Since Java 8, it can have default and static methods also |
| The **abstract** keyword is used to declare abstract class | The **interface** keyword is used to declare interface |
| An abstract class can be **extended** using keyword "extends" | An interface class can be **implemented** using keyword "implements" |

**what is the difference between Static and no static variable or function in Java?**

What is promise in JavaScript?

what is callback function in JavaScript?

What's the difference between primary key, foreign key and unique key?

What is the difference between inner join and left outer join?

Different joins

What is DOM

What is join in Database

What is early and late binding?  
What is a copy constructor?  
What is static and dynamic binding?  
Data mirroring, replications.

sub query and its types

view

Inner join a Database integrity.

Unique Key, Primary Key, Foreign Key, Temp Table, Table Variable, Trigger, Stored Procedure

Difference between Array and ArrayList

How garbage collector works