Practical 8 Questions

Q1. What is multiple inheritance?

→ A class inheriting from 2 or more parent classes.

Q2. Why do we not use the concept of implement for inheritance using classes in java?

→ We use the keyword ‘implement’ for interface. Since interface has abstract methods inside it (that is just the method name and no body), there is no point of using interface for inheritance as we will have to define the body of the methods every time we implement the interface; which very much contrasts the purpose of inheritance (that is inheritance helps us not to write the same piece of code again and again).

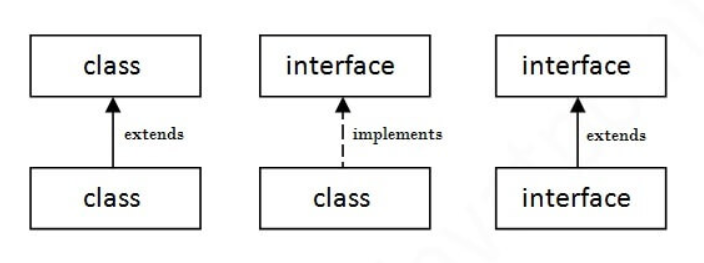
Q3. What is interface?

→ An interface in Java is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Since Java 8, we can have default and static methods in an interface.

Since Java 9, we can have private methods in an interface.

**interface** printable{

**void** print();

}

**class** A6 **implements** printable{

**public** **void** print(){

System.out.println("Hello");

}

**public** **static** **void** main(String args[]){

A6 obj = **new** A6();

obj.print();

}

}

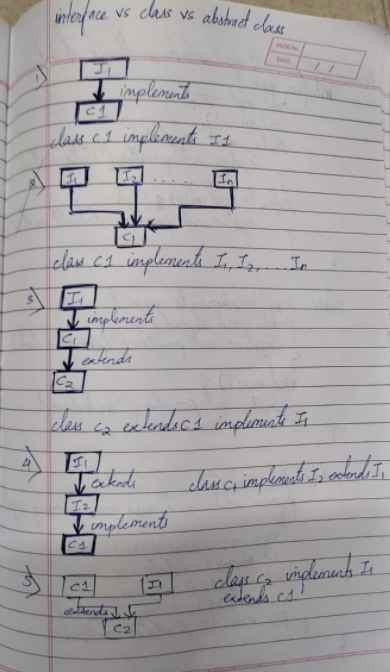
O/P : Hello

Q4. What are the contents in interface?

→ A java interface contains static and final variables. Interface can have only abstract methods (It cannot have a body). It can have static methods and default methods. All the methods are public and abstract; and all the fields are public, static and final.

Q5. How to implement interface? Explain every method using example.

→



Q6. Can we inherit inheritance?

→ No we cannot.

Q7. What is the difference between class and interface?

Graphical user interface, text, application

Description automatically generated→

Q8. Explain the concept of default method in interface?

→ **interface** In1 {

**final** **int** a = 10;

**default** **void** display() {

         System.out.println("hello");

     }

}

**class** TestClass **implements** In1 {

**public** **static** **void** main (String[] args) {

         TestClass t = **new** TestClass();

         t.display();

     }

}

Output : hello

Q9. Explain the concept of static method in interface?

→ **interface** In1

{

**final** **int** a = 10;

**static** **void** display()

    {

        System.out.println("hello");

    }

}

// A class that implements the interface.

**class** TestClass **implements** In1

{

    // Driver Code

**public** **static** **void** main (String[] args)

    {

        In1.display();

    }

}

Output : hello

We can call the interface method without creating the interface object.

Q10. Explain the difference between abstract class and interface?

→

|  |  |
| --- | --- |
| ABSTRACT CLASS | INTERFACE |
| 1. Abstract class can have abstract and non abstract methods. | Interfaces can have only abstract methods. Since Java 8, it can have default and static methods also. |
| 1. Abstract class doesn’t support multiple inheritance. | Interface supports multiple inheritance. |
| 1. Abstract class can have final, non final, static and non static variables. | Interface has only static and final variables. |
| 1. Abstract class can provide implementation of interface. | Interface cannot provide implementation of abstract class. |
| 1. The abstract keyword is used to declare abstract class. | The interface keyword is used to declare interface. |
| 1. Example:   Public abstract class c1{  Public abstract void draw();  } | Example:  Public interface draw{  Void draw();  } |

Q. Can we make constructor as static and final?

→ No

Q. Can we access the static fields and methods using this keyword?

→ Yes

Q. Can static methods access non static methods of the class?

→ No

Q. Can final methods access static members of the class?

→ Yes

Q. Can we override the final method?

→ No

Q. Can we overload the final method?

→ Yes

Q. Can we overload the static method?

→ Yes

Q. Can we override the static method?

→ Yes

Q. Can we inherit the final class?

→ Yes