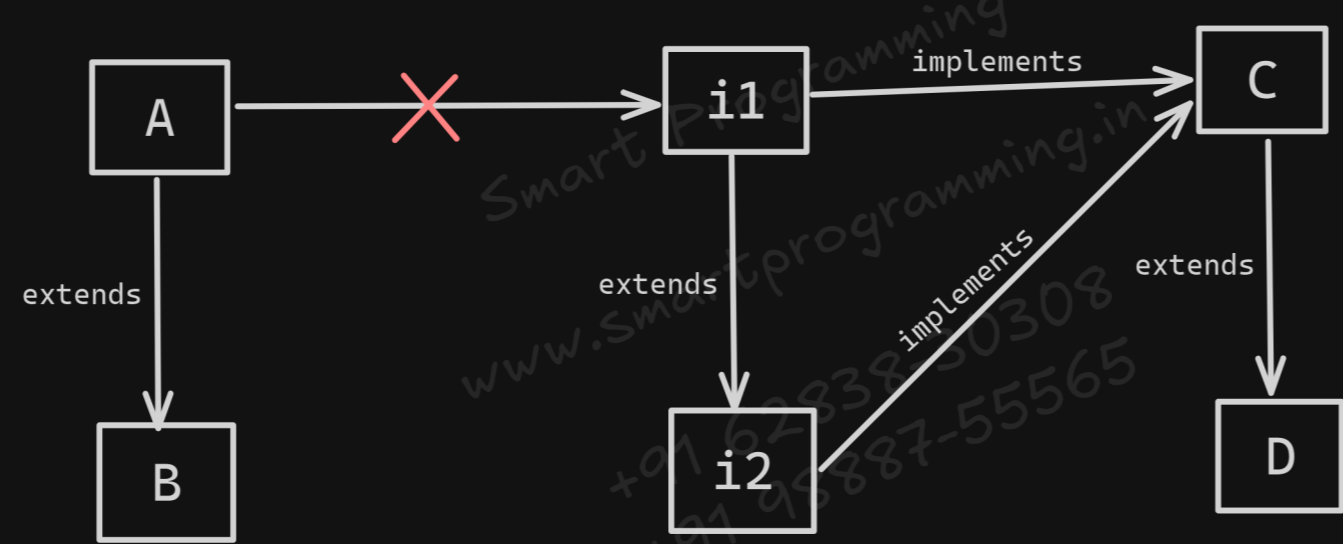


Inheritance



```
class B extends A
interface i2 extends i1
class C implements i1, i2
class C extends D implements i1, i2
```

```
interface i2 implements i2
interface i1 extends A
interface i1 implements A
class C extends A, B
```

Abstraction

- ⇒ It is the concept of :
 - hiding the implementation and showing the important details
- ⇒ Real World Example :-
 - Breaks in Car : breaks internal working is hided and breaks are visible
- ⇒ How to achieve abstraction :-
 - It is achieved by 2 ways :
 1. Using Abstract Class
 2. Using Interfaces
- ⇒ Abstraction is used to achieve "security"

Abstract Class

- ⇒ Abstract class is the class which contains both concrete methods and abstract methods
- ⇒ To create abstract class, we have to use "abstract" keyword
- ⇒ For example :

```
abstract class MainApp
{
    void m1()
    {
        //body
    }

    abstract void m2();
}
```

```
abstract class MainApp
{
    void m1()
    {
        //body
    }
}
```

```
abstract class MainApp
{
    abstract void m2();
}
```

- ⇒ Points to remember :-
 1. If any class has abstract method, then that class should be abstract class
 2. Its not compulsory to have abstract method in abstract class
 3. Abstract class cannot be instantiated (we cannot create an object of abstract class)
 4. If any class extends the abstract class, then it must override all the abstract methods

Abstract Method

- ⇒ Methods which does not have any implementation or body are known as abstract methods
- ⇒ To create abstract methods we have to use "abstract" keyword
- ⇒ For example :-

```
abstract void m1();
```

abstract method

```
void m1()
{
    //body
}
```

concrete method

Interfaces

- ⇒ Interface are similar to abstract class but having all the methods of abstract type
- ⇒ Syntax :

```
interface InterfaceName
{
    // public static final variables
    // public abstract methods
}
```

- ⇒ For example :

```
interface I1
{
    void m1();

    abstract void m2();

    public abstract void m3();

    void m4()
    {
    }

    protected abstract void m5();
}
```

- ⇒ Use of Interfaces :-
 1. It is used to achieve 100% abstraction
 2. It is used to achieve multiple inheritance
 3. It is used to achieve loose coupling

- ⇒ Updates for Interface :

- Till Java 7, interfaces can contain only abstract methods and constants.
- In Java 8, we can provide the implementation of methods using "default methods" and "static methods".
- In Java 9, we can provide the implementation of methods using "private methods".

Interview Question

Abstraction

1. What is abstraction in Java? How does it differ from encapsulation?
2. What are the two ways to achieve abstraction in Java?
3. How much abstraction is provided by an abstract class versus an interface?
4. Can an abstract class have constructors in Java? Why or why not?
5. Why can't we create an object of an abstract class?
6. What happens if an abstract class has only concrete methods?
7. Can an abstract class have private methods? Why or why not?
7. What happens if we do not implement all methods of an interface in an abstract class?
8. What is the use of the abstract keyword in Java?
9. Give a real-life example where abstraction is useful.
10. What is the difference between abstraction and encapsulation?

Interface in Java

11. What is an interface in Java, and why is it used?
12. How is an interface different from an abstract class?
13. Can an interface have variables? If yes, what are their default properties?
14. What types of methods can an interface contain in Java 8 and later versions?
15. Can an interface extend another interface? How does it work?
16. Can an interface have a constructor? Why or why not?
17. What happens if a class implements multiple interfaces with the same default method?
18. What happens if two interfaces have the same static method and a class implements both?
19. What is a functional interface in Java? Provide an example.
20. What are marker interfaces in Java? Give examples.
21. Provide a real-world example where an interface is useful.