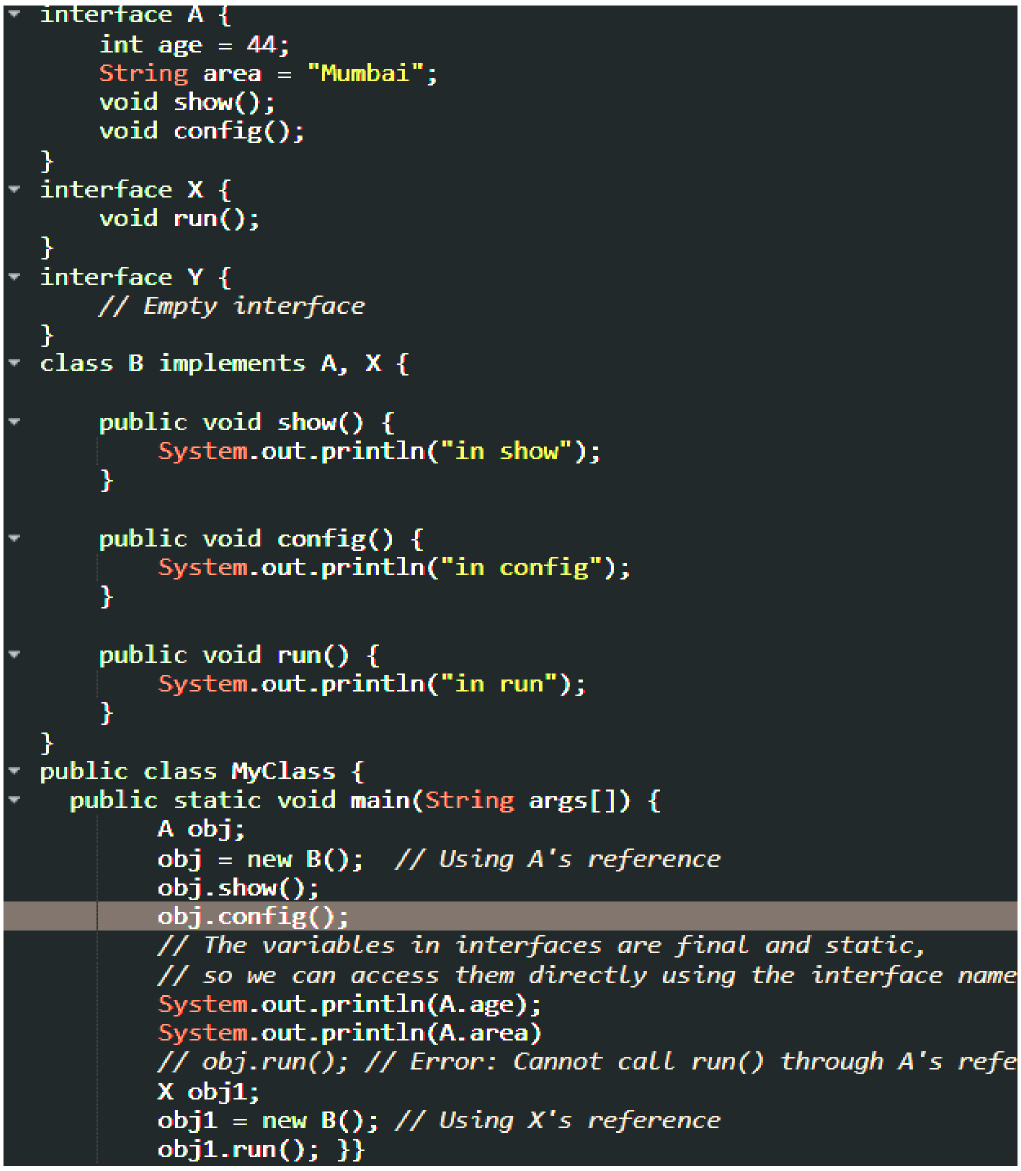
10.7-More on Interfaces

In previous sections, we've explored interfaces in Java in depth. Now, let's dive into more advanced topics regarding interfaces. A common question is: **Can a class implement multiple interfaces?**

Multiple Interface Implementation: Example

In Java, a class can implement multiple interfaces. This is unlike class inheritance, where a class can only extend one class. Implementing multiple interfaces allows for a more flexible design.

Here's an example:



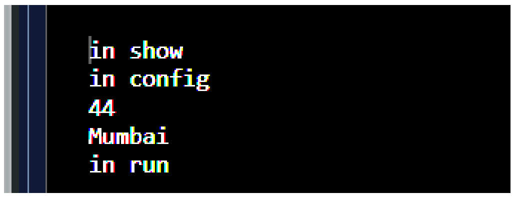
```java
interface A {
    int age = 44;
    String area = "Mumbai";
    void show();
    void config();
}
interface X {
    void run();
}
interface Y {
    // Empty interface
}
class B implements A, X {

    public void show() {
        System.out.println("in show");
    }

    public void config() {
        System.out.println("in config");
    }

    public void run() {
        System.out.println("in run");
    }
}
public class MyClass {
  public static void main(String args[]) {
        A obj;
        obj = new B();  // Using A's reference
        obj.show();
        obj.config();
        // The variables in interfaces are final and static,
        // so we can access them directly using the interface name
        System.out.println(A.age);
        System.out.println(A.area)
        // obj.run(); // Error: Cannot call run() through A's refe
        X obj1;
        obj1 = new B(); // Using X's reference
        obj1.run(); }}
```

Output:



```
in show
in config
44
Mumbai
in run
```

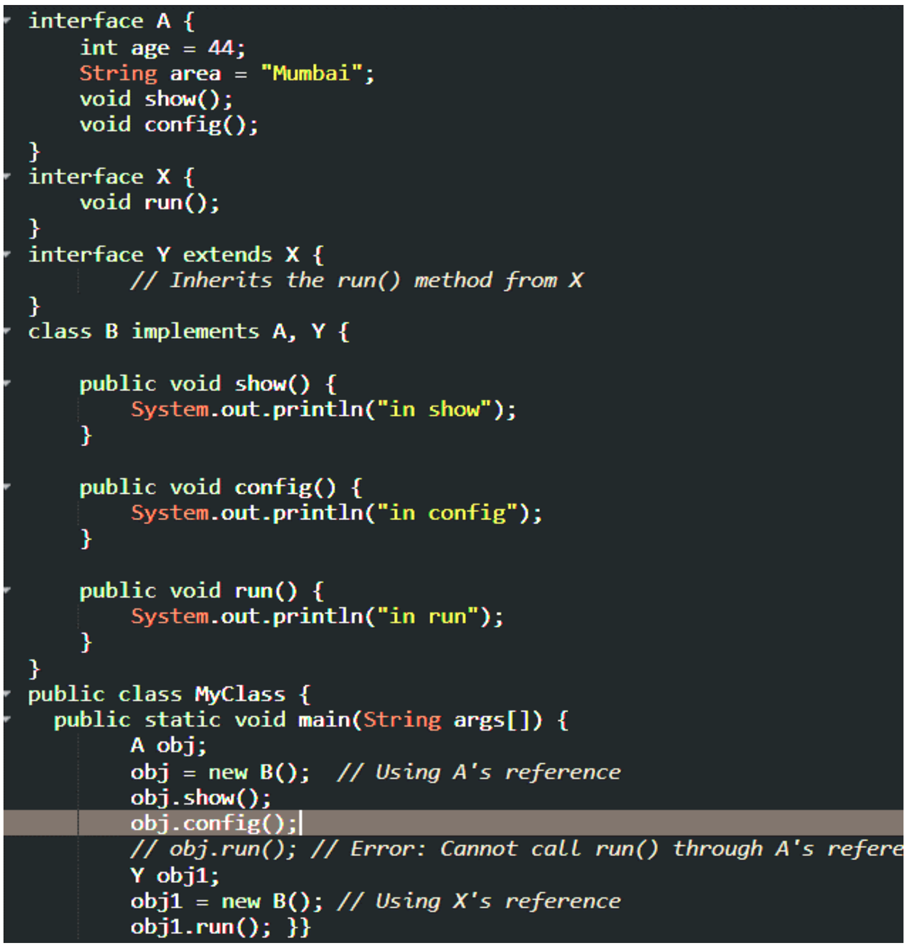
Explanation

- **Interface A**: Contains two abstract methods show() and config(), which are implemented by class B.
- **Interface X**: Contains one abstract method run(), also implemented by class B.
- **Interface Y**: This is an empty interface (also called a marker interface).

In Java, a class can only extend **one class**, but it can implement **multiple interfaces**. In this example, class B implements both A and X.

- The variables age and area in interface A are both **final** and **static**, which means they can be accessed directly through the interface name, without creating an object.

Multiple Inheritance Through Interfaces



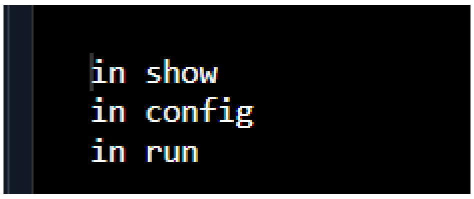
```
interface A {
    int age = 44;
    String area = "Mumbai";
    void show();
    void config();
}
interface X {
    void run();
}
interface Y extends X {
        // Inherits the run() method from X
}
class B implements A, Y {

    public void show() {
        System.out.println("in show");
    }

    public void config() {
        System.out.println("in config");
    }

    public void run() {
        System.out.println("in run");
    }
}
public class MyClass {
  public static void main(String args[]) {
        A obj;
        obj = new B();  // Using A's reference
        obj.show();
        obj.config();
        // obj.run(); // Error: Cannot call run() through A's refere
        Y obj1;
        obj1 = new B(); // Using X's reference
        obj1.run(); }}
```

Output:



```
in show
in config
in run
```

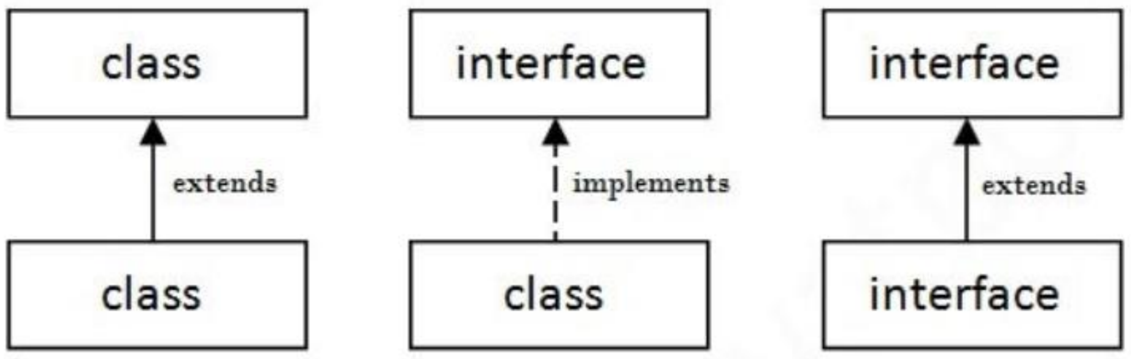
Let's extend the previous example with multiple interface inheritance. Suppose we have interface X, and now Y extends X. The run() method will automatically be inherited by Y, and class B will need to implement it.

Explanation

- **Interface Y extends X**: Y inherits the run() method from X. When class B implements Y, it must still implement the run() method because Y inherits it from X.
- In the Demo class, if we use a reference of type Y, we can call the run() method without issues.

Summary of Key Concepts

- **Class to Class**: Use the extends keyword.
- **Class to Interface**: Use the implements keyword.
- **Interface to Interface**: Use the extends keyword.



A class can implement multiple interfaces at the same time. In the example above, class B implements both A and Y (which extends X). The run() method must be implemented in class B because Y inherits this method from X.