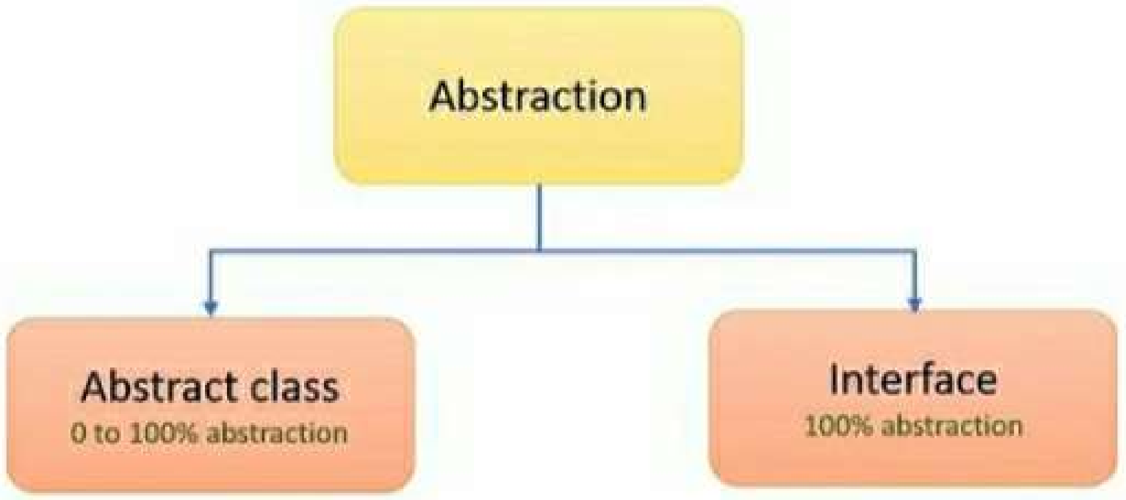
JAVA

ABSTRACTION

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.



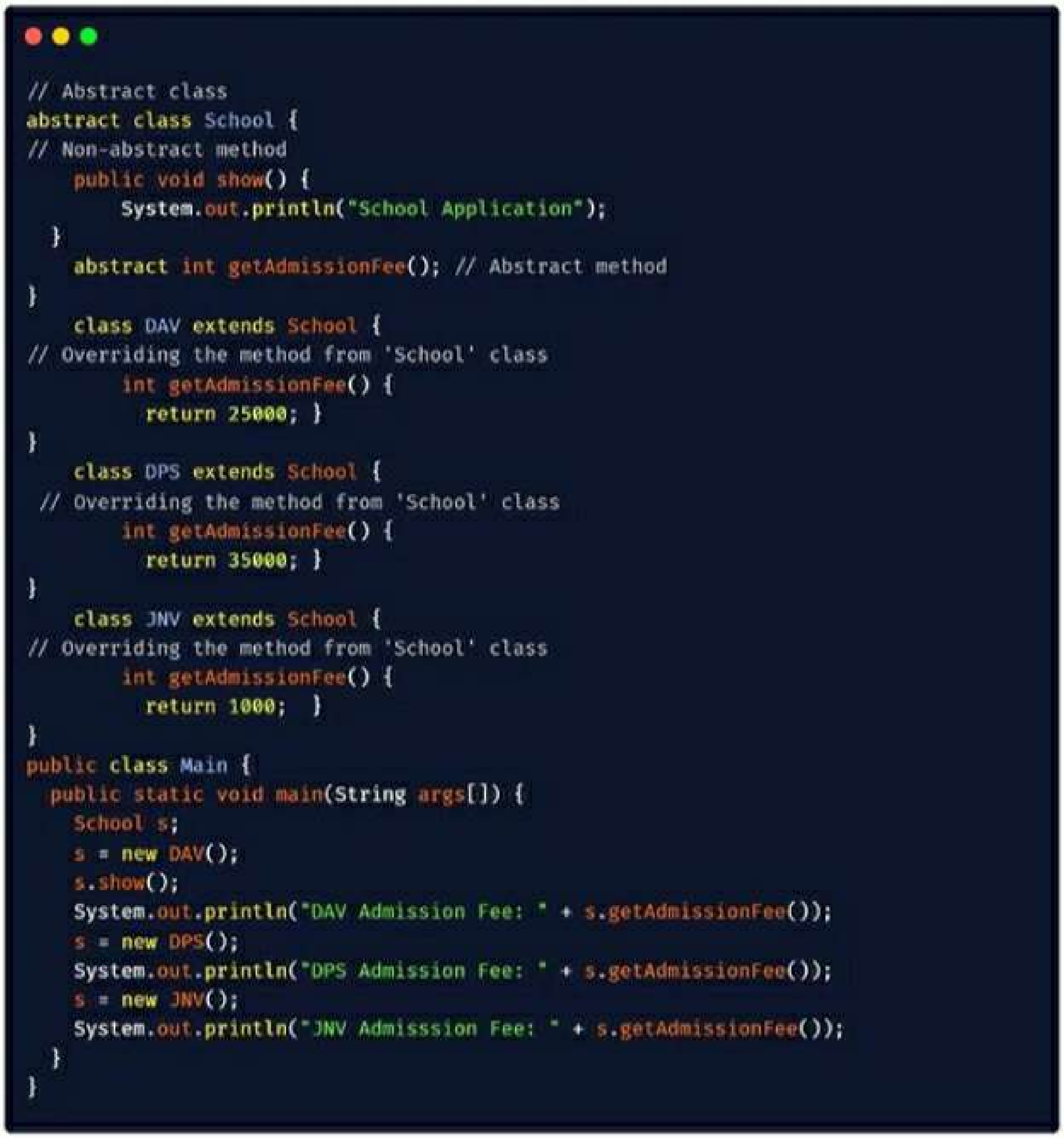
Abstract class

A class which is declared as abstract is known as an abstract class.

Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods.

A method which is declared as abstract and does not have implementation is known as an **abstract method.**



```
// Abstract class
abstract class School {
// Non-abstract method
    public void show() {
        System.out.println("School Application");
  }
    abstract int getAdmissionFee(); // Abstract method
}
    class DAV extends School {
// Overriding the method from 'School' class
        int getAdmissionFee() {
          return 25000; }
}
    class DPS extends School {
 // Overriding the method from 'School' class
        int getAdmissionFee() {
          return 35000; }
}
    class JNV extends School {
// Overriding the method from 'School' class
        int getAdmissionFee() {
          return 1000;  }
}
public class Main {
  public static void main(String args[]) {
    School s;
    s = new DAV();
    s.show();
    System.out.println("DAV Admission Fee: " + s.getAdmissionFee());
    s = new DPS();
    System.out.println("DPS Admission Fee: " + s.getAdmissionFee());
    s = new JNV();
    System.out.println("JNV Admisssion Fee: " + s.getAdmissionFee());
  }
}
```

Interface in Java

An interface in Java is a blueprint of a class. It has static constants and abstract methods. Java Interface also **represents the IS-A relationship.**

Why use Java interface?

There are mainly three reasons to use interface.

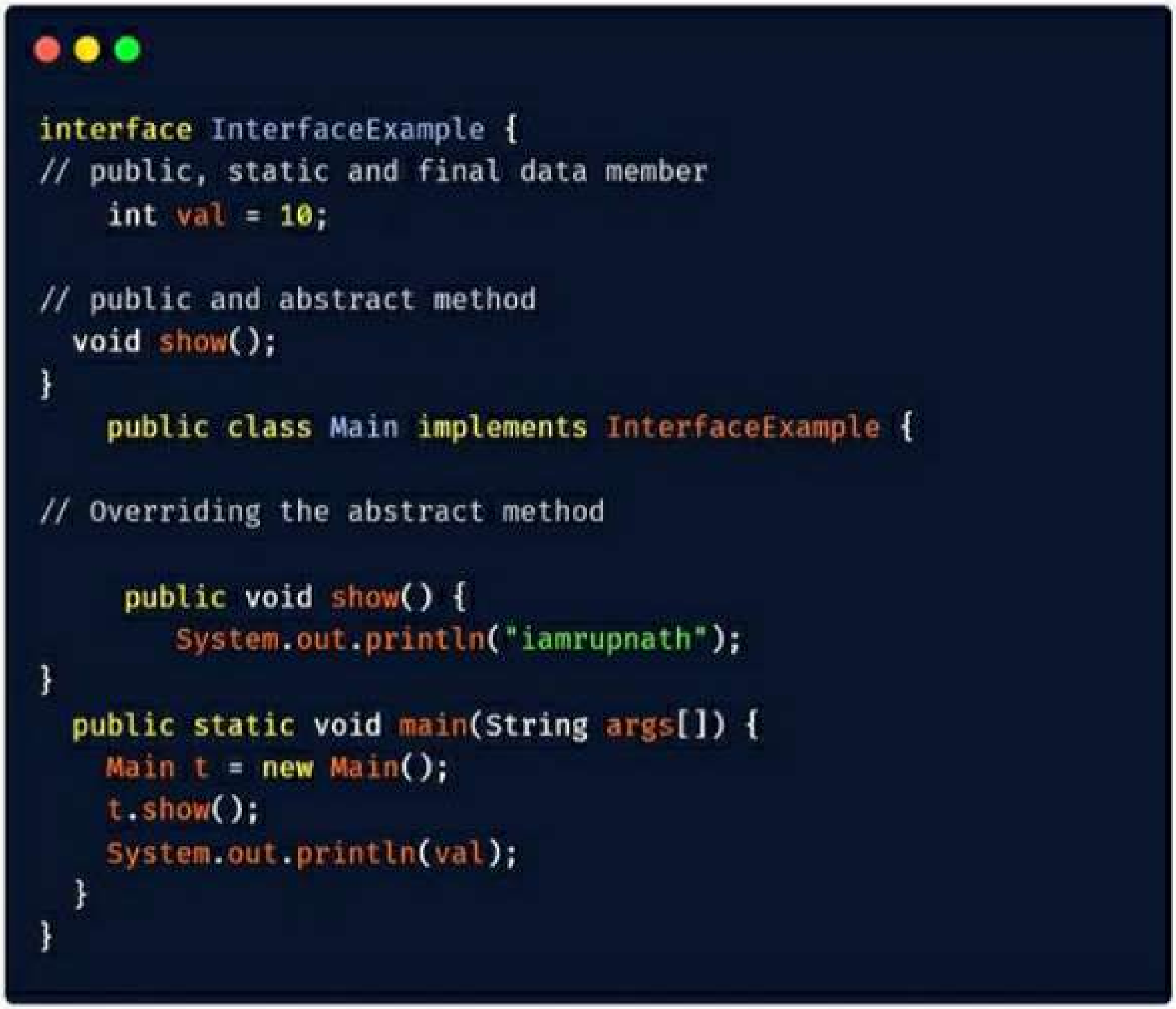
- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

Why use interfaces when we have abstract classes:

Abstraction is achieved by the use of interfaces and abstract classes. However, the reason for using interfaces is that abstract classes may contain non-final variables, while interface variables are public, final, and static.

Syntax

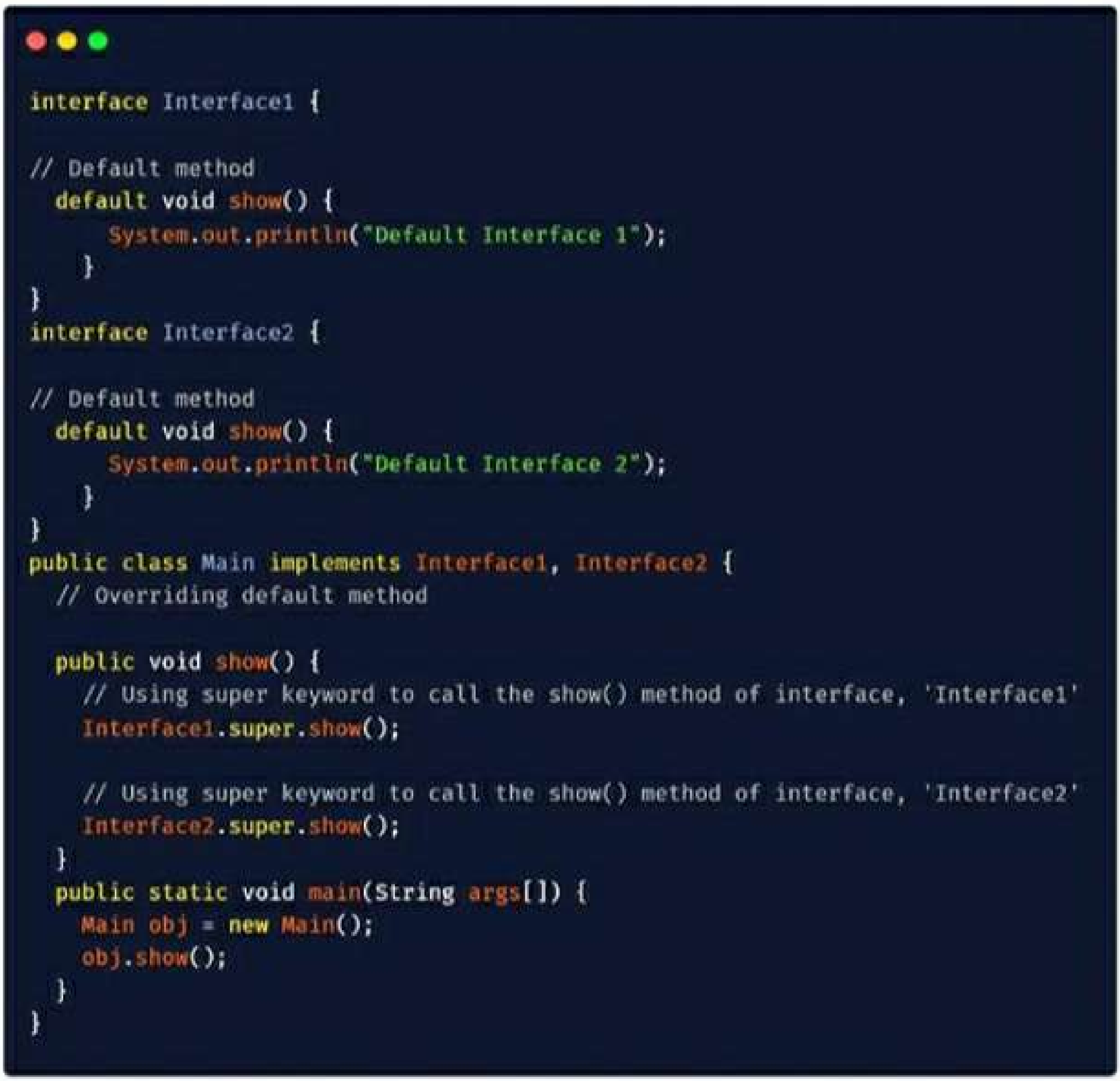
```
interface interface_name {
/* Constant Fields and Abstract Methods * /
}
```



```java
interface InterfaceExample {
// public, static and final data member
    int val = 10;

// public and abstract method
  void show();
}
    public class Main implements InterfaceExample {

// Overriding the abstract method

     public void show() {
        System.out.println("iamrupnath");
}
  public static void main(String args[]) {
    Main t = new Main();
    t.show();
    System.out.println(val);
  }
}
```

Multiple Inheritance in Java using an Interface

Because of the ambiguity problem, Java does not support multiple inheritance in the case of classes. However, we can use interfaces to implement multiple inheritance in Java since there is no ambiguity in the case of interfaces. It is because of its implementation provided by the implementation class



```java
interface Interface1 {

// Default method
  default void show() {
      System.out.println("Default Interface 1");
    }
}
interface Interface2 {

// Default method
  default void show() {
      System.out.println("Default Interface 2");
    }
}
public class Main implements Interface1, Interface2 {
  // Overriding default method

  public void show() {
    // Using super keyword to call the show() method of interface, 'Interface1'
    Interface1.super.show();

    // Using super keyword to call the show() method of interface, 'Interface2'
    Interface2.super.show();
  }
  public static void main(String args[]) {
    Main obj = new Main();
    obj.show();
  }
}
```

Output:

```
Default Interface 1

Default Interface 2
```

Advantages of Abstraction

- It reduces the complexity of viewing things.
- Avoids code duplication and increases reusability.
- It improves the maintainability of the application.
- It improves the modularity of the application.
- Can change internal implementation of class independently without affecting the user.
- Helps to increase the security of an application or program as only essential details are provided to the user.
- The enhancement will become very easy because without affecting end-users we can able to perform any type of changes in our internal system.