Date
Page No.

4. B) Interfaces

⇒ An interface in Java is a collection of abstract methods that define a set of rules a class must follow.

- Can have :

(i) Only Abstract methods
- All methods are public abstract (by default)

(ii) Only constants
- All fields are public static final (by default)

(iii) No constructors ✗✗
(iv) No method bodies ✗✗
(v) No instance variables ✗✗
(vi) Can extend multiple interfaces
(vii) A class can implement multiple interfaces
(viii) Cannot be instantiated ✗✗

(i) Interface Syntax

```
interface InterfaceName {
// Constant (implicitly public static final)
   int SOME_CONSTANT = 10;

// Abstract method (implicity public abstract)
    void method1();
int method2(int param);
}
```

(ii) Implementing an Interface

```
class ImplementingClass implements InterfaceName {
    // Must implement ALL interface methods
    @Override
    public void method() {
        // implementation code
    }
    @Override
    public int method2(int param) {
        // implementation code here
        return param * 2;
    }
}
```

(iii) Key points and Rules.

1. Declared with the interface keyword.
2. All interface methods are implicitly public abstract (no private, static, or final methods pre-java 8).
3. All fields are implicitly public static final constants.
4. NO constructor allowed
5. A class can implement multiple interfaces using comma separation.
6. Classes must implement all methods of their interfaces or declare themselves abstract.
7. Interfaces can extend multiple interfaces simultaneously, eg. interface C extends A, B {---}.
8. Cannot instantiate interfaces directly (new InterfaceName() is illegal).
9. Interfaces are for defining capabilities/contracts, separate from implementation.

Date
Page No.

(iv) Interface Reference Variable

- You can declare variables of interface type and assign them instances of implementing class -

ex →

```
InterfaceName obj = new ImplementingClass();
obj.method1();
---
```

(Examples is in Github practice folder)