Chapter-6: Interfaces

PART-II: Sample Short Questions & Answers

1. Define an interface.

An interface in Java is a collection of abstract methods (without body) that define a contract for classes to implement. It provides multiple inheritance and abstraction.

👉 Example:

```
interface Animal { void sound(); }
```

---

2. How to declare an interface?

An interface is declared using the interface keyword. It contains only constants and abstract methods (before Java 8).

👉 Example:

```
interface Shape {
  void draw();
}
```

---

3. What is abstract class?

An abstract class is a class that may contain both abstract (unimplemented) and concrete (implemented) methods. It cannot be instantiated directly.

👉 Example:

```
abstract class Animal { abstract void eat(); }
```

---

4. Write any 2 uses of interface.

- To achieve **multiple inheritance**.
- To ensure **loose coupling** and abstraction in applications.

---

5. What is the keyword used to extend interface?

Interfaces use the extends keyword to extend another interface.

👉 Example:

```
interface A { void show(); }
interface B extends A { void display(); }
```

---

6. How to extend multiple interfaces?

In Java, one interface can extend multiple interfaces using commas.

👉 Example:

```
interface A { void show(); }
interface B { void print(); }
interface C extends A, B { void display(); }
```

---

7. Write syntax to declare interface.

```
interface InterfaceName {
  // constant fields
  // abstract methods
}
```

---

8. What specifier is used for interface?

By default, interface methods are public abstract, and fields are public static final.

---

9. Write short example to implement an interface.

👉 Example:

```
interface Animal { void sound(); }
class Dog implements Animal {
  public void sound() { System.out.println("Bark"); }
}
```

---

10. What is the keyword used to inherit an interface?

The implements keyword is used when a class inherits (implements) an interface.

👉 Example:

```
class Dog implements Animal { ... }
```

✅ PART-III: Sample Long Questions & Answers

1. Explain difference between interface and abstract class.

- **Abstract Class:** Can have abstract + concrete methods, constructors, instance variables, etc. Supports partial abstraction.
- **Interface:** Contains only abstract methods (until Java 8) and constants. Supports multiple inheritance.
  👉 Example:

```
abstract class Animal { abstract void eat(); void sleep(){System.out.println("Sleeping");} }
interface Flyable { void fly(); }
```

---

2. Define interface and briefly explain how to declare and implement an interface.

An interface defines a contract that a class must follow. It is declared using interface and implemented using implements.

👉 Example:

```
interface Vehicle { void drive(); }
class Car implements Vehicle {
  public void drive() { System.out.println("Car is driving"); }
}
```

---

3. How to extend interface and multiple interface explain with example.

- One interface can extend multiple interfaces.
- A class can implement multiple interfaces.

👉 Example:

```
interface A { void show(); }

interface B { void print(); }

interface C extends A, B { void display(); }

class Demo implements C {

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

  public void print(){System.out.println("Print");}

  public void display(){System.out.println("Display");}

}
```

---

4. Write a program that explains abstract class and interface.

👉 Example:

```
abstract class Animal {

  abstract void eat();

  void sleep(){ System.out.println("Sleeping"); }

}


interface Flyable {

  void fly();

}


class Bird extends Animal implements Flyable {

  void eat(){ System.out.println("Bird eats seeds"); }

  public void fly(){ System.out.println("Bird flies high"); }

}


public class Test {

  public static void main(String[] args) {

    Bird b = new Bird();

    b.eat();

    b.sleep();

    b.fly();

  }

}
```


---

5. Define Interface with example advantages of using interface.

- **Definition:** An interface is a blueprint of a class containing abstract methods.
- **Advantages:**
    1. Achieves multiple inheritance.
    2. Provides abstraction and loose coupling.
    3. Improves flexibility and scalability.

    👉 Example:

```
interface Payment { void pay(); }

class CreditCard implements Payment {

  public void pay(){ System.out.println("Payment by Credit Card"); }

}
```