Detailed Notes on Abstraction and Interfaces in Java

Abstraction in Java

1. **Definition**: Hides complex implementation details and shows only essential functionalities to users.
2. **Purpose**:
    - Represent essential features without including background details.
    - Simplify user interaction by removing unnecessary details.

How to Achieve Abstraction:

1. **Abstract Class**: Provides 0-100% abstraction.
2. **Interface**: Provides 100% abstraction.

Abstract Classes

1. **Definition**:
    - Declared with the `abstract` keyword.
    - Cannot instantiate objects directly.
    - Can include both abstract and concrete methods.
2. **Key Points**:
    - `abstract` is a non-access modifier for classes, methods, and inner classes.
    - Subclasses must implement abstract methods.
    - Variables cannot be abstract.
    - Commonly used for sharing behavior across multiple subclasses.
3. **Abstract Methods**:
    - Declared using `abstract` but without a body.
    - Syntax: `abstract returnType methodName(arguments);`
4. **Use Cases**:
    - Share methods across non-abstract subclasses with their own implementations.
    - Promote code reusability.
5. **Important Points**:
    - Constructors in abstract classes are invoked through subclass constructors.
    - Abstract classes can include private, final, and static methods.
    - Does not support multiple inheritance but can implement interfaces.

Advantages of Abstract Classes

1. Simplifies code by enabling flexible implementation of methods.
2. Encourages code management and reusability.

Interfaces in Java

1. **Definition**:

- Collection of abstract methods and constants.
- Provides complete abstraction.

2. **Key Features**:
   - All methods are `public` and `abstract` by default.
   - Variables are `public`, `static`, and `final` (constants).
   - Can be implemented by multiple classes, enabling multiple inheritance.
3. **Rules**:
   - Cannot instantiate directly.
   - Must be implemented by classes to provide method definitions.
   - Cannot include instance variables or constructors.
4. **Syntax**:

```
public interface InterfaceName {
    int CONSTANT = value;
    void methodName();
}
```

5. **Additional Features (Java 8+)**:
   - Default methods with implementation.
   - Static methods.
   - Private methods (Java 9+).

Why Use Interfaces?

1. Helps expose project APIs to third parties.
2. Supports full abstraction and multiple inheritance.
3. Enables polymorphism by allowing different classes to implement the same interface with unique behaviors.

Key Points on Interfaces

- Interfaces can extend multiple other interfaces.
- Implementation classes must define all methods in the interface.
- Polymorphism: Multiple classes implementing the same interface can behave differently based on their implementations.

MCQ Question Bank: Abstraction and Interfaces in Java

Abstraction in Java

1. **What is the primary purpose of abstraction in Java?**
   a) To achieve multiple inheritance
   b) To hide unnecessary details from users
   c) To make code less readable
   d) To improve code performance
   Answer: b
2. **Which of the following techniques is used to achieve abstraction in Java?**
   a) Concrete Classes and Objects
   b) Abstract Classes and Interfaces

c) Constructors and Destructors
d) Packages and Modules
Answer: b

3. **How much abstraction can be achieved using an abstract class in Java?**
a) 0%
b) 50%
c) 0 to 100%
d) 100%
Answer: c

Abstract Classes

4. **Which keyword is used to declare an abstract class in Java?**
a) static
b) abstract
c) final
d) class
Answer: b

5. **What happens if a class contains an abstract method?**
a) The class must be declared as abstract.
b) The class cannot have any constructors.
c) The class becomes final.
d) The class must extend another abstract class.
Answer: a

6. **Which of the following is true about abstract methods?**
a) They have a method body.
b) They can be declared private.
c) They do not contain a method body.
d) They must be declared as static.
Answer: c

7. **What is a non-abstract class also known as?**
a) Abstract Class
b) Concrete Class
c) Interface
d) Static Class
Answer: b

Interfaces

8. **What is the default access modifier for methods in an interface?**
a) private
b) protected
c) public
d) default
Answer: c

9. **Which of the following is true about variables declared in an interface?**
a) They are private by default.
b) They are non-final by default.

c) They are static and final by default.
d) They cannot be initialized.
Answer: c

10. **Which version of Java introduced default methods in interfaces?**
    a) Java 6
    b) Java 7
    c) Java 8
    d) Java 9
    Answer: c

11. **Why are interfaces used in Java?**
    a) To provide partial abstraction
    b) To support multiple inheritance
    c) To implement final methods
    d) To make variables mutable
    Answer: b

12. **What is the syntax to implement multiple interfaces in a class?**
    a) `class MyClass extends Interface1, Interface2`
    b) `class MyClass implements Interface1, Interface2`
    c) `class MyClass inherits Interface1, Interface2`
    d) `class MyClass uses Interface1, Interface2`
    Answer: b

General Concepts

13. **Which of the following statements about abstract classes is false?**
    a) An abstract class can have final methods.
    b) An abstract class can implement multiple interfaces.
    c) Abstract methods can be static.
    d) Abstract classes can have constructors.
    Answer: c

14. **Which of the following is NOT allowed in an interface?**
    a) Static methods
    b) Constructors
    c) Default methods
    d) Nested interfaces
    Answer: b

15. **What is the term for when multiple classes implement the same interface differently?**
    a) Inheritance
    b) Polymorphism
    c) Encapsulation
    d) Abstraction
    Answer: b

This MCQ bank can help test understanding and reinforce the concepts of abstraction and interfaces in Java.