1. **What does the this keyword refer to in a Java class?**
   A) The class itself
   B) The current object instance
   C) A static reference
   D) The parent class

2. **Consider the following code:**

```java
public class Demo {
    int a = 10;
    void display() {
        System.out.println(this.a);
    }
}
```

What will be printed when `display()` is called on a `Demo` object?
   A) 0
   B) 10
   C) A memory address
   D) A compile-time error

3. **Examine the following setter method:**

```java
public class Employee {
    String name;
    public void setName(String name) {
        this.name = name;
    }
}
```

How does this help in the setName method?
   A) It distinguishes between the instance variable and the parameter
   B) It calls a static method
   C) It declares a new variable
   D) It resets the variable to its default value

4. **Look at the static method example:**

```java
public class StaticTest {
    static int count = 5;
    public static void printCount() {
        // Uncommenting the following line will cause a compile-time error:
        // System.out.println(this.count);
    }
}
```

Why would using `this` inside a static method cause an error?
   A) Because `this` is not accessible in a static context
   B) Because `this` can only be used in constructors
   C) Because static methods do not have parameters
   D) Because static methods cannot reference any class members

5. **Consider the code snippet:**

```java
public class Test {
    public void show() {
        System.out.println("Display method called");
    }
    public void display() {
        this.show();
    }
}
```

What does the statement `this.show();` do?

A) Calls a static method

B) Calls the `show()` method of the current object

C) Creates a new object

D) Calls a method from the parent class

---

6. **Which of the following statements about this is TRUE?**

A) It can only be used in constructors

B) It always refers to the current object

C) It is a reference to the parent class

D) It can be used in static methods

---

7. **Given the code below, what will be printed when `display()` is invoked?**

```java
public class Sample {
    String text = "Java";
    public void display() {
        String text = "Hello";
        System.out.println(this.text);
    }
}
```

A) Hello

B) Java

C) null

D) A compile-time error

---

8. **Examine the following inner class usage:**

```java
public class Outer {
    int x = 100;
    class Inner {
        int x = 50;
        void display() {
            System.out.println(Outer.this.x);
        }
    }
}
```

How does the inner class reference the outer class's instance variable?

A) Using `this.x`
B) Using `Outer.this.x`
C) Using `super.x`
D) Using `Inner.this.x`

---

9. **Consider this fluent-style method:**

```java
public class Fluent {
    int value;
    public Fluent setValue(int value) {
        this.value = value;
        return this;
    }
}
```

What design pattern is illustrated by returning `this` in the `setValue` method?
A) Singleton Pattern
B) Builder/Fluent Pattern
C) Factory Pattern
D) Adapter Pattern

---

10. **Why is this not allowed in a static method?**
A) Because static methods have no access to instance-specific data
B) Because static methods are only used for object creation

---

11. **In the context of inheritance, what is the difference between `this` and `super`?**
A) `this` refers to the current instance, while `super` refers to the immediate parent class
B) `this` is for static members, while `super` is for instance members
C) They both refer to the same object
D) `this` can only be used in inner classes

---

12. **Analyze the builder pattern example:**

```java
public class Car {
    String color;
    public Car setColor(String color) {
        this.color = color;
        return this;
    }
}
```

C) Because using this creates a new object
D) Because this must be used only in constructors

What is the benefit of returning this in the setColor method?

A) It allows for method chaining

B) It creates a new object

C) It resets the object's state

D) It overrides the default method

13. **Review the event handling example:**

```java
public class ButtonHandler implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        processEvent(this);
    }
    void processEvent(ButtonHandler handler) {
        // Process the event using handler
    }
}
```

In this context, what does passing `this` achieve?

A) It passes the current instance as the event source

B) It creates a new event

C) It calls a static method

D) It references a null object

14. **Consider the following code:**

```java
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
    public void compute() {
        int result = this.add(5, 10);
        System.out.println(result);
    }
}
```

How does `this.add(5, 10)` function in the `compute` method?

A) It calls a static method

B) It invokes the `add` method of the current object

C) It creates a new `Calculator` object

D) It references the parent class method

15. **Review the following method that resolves variable shadowing:**

```java
public class Employee {
    String id;
    public void updateId(String id) {
        this.id = id;
    }
}
```

How does the statement `this.id = id;` resolve the shadowing issue?

A) It assigns the parameter `id` to the instance variable

B) It assigns a new value to the parameter

C) It creates a new variable named `id`

D) It calls a method named `id`

16. **Examine this update method:**

```java
public class Sample {
    int data;
    public Sample update(int data) {
        this.data = data;
        return this;
    }
}
```

What is the purpose of returning `this` from the `update` method?

A) To enable method chaining

B) To create a new instance

C) To reset the object's state

D) To call a static method

17. **Look at the following fluent interface example:**

```java
public class Fluent {
    int count;
    public Fluent setCount(int count) {
        this.count = count;
        return this;
    }
    public Fluent increment() {
        this.count++;
        return this;
    }
}
```

What concept does this code demonstrate?

A) Method chaining

B) Method overloading

C) Static binding

D) Late binding

18. **Consider this example where** `this` **is passed to another method:**

```java
public class Processor {
    public void process() {
        log(this);
    }
    public void log(Processor p) {
        System.out.println("Logging processor: " + p);
    }
}
```

What is the main benefit of passing this to the log method?
A) It logs the current object's reference
B) It creates a new Processor object
C) It resets the object state
D) It calls a static log method

19. **Which scenario best illustrates method chaining using the this keyword?**
A) Each method returns a new object
B) Each method returns the current object to allow successive method calls

D) Using `this` exclusively in constructors

20. **How can** `this` **be used to differentiate between instance variables and parameters in a method?**
C) Using this to access static members
A) By automatically renaming the variables
B) By explicitly referencing the instance variable with this
C) By converting instance variables to static members
D) By using a different data type

21. **Consider the following code:**

```java
public class NumberPrinter {
    int number = 20;
    public void printNumber() {
        System.out.println(this.number);
    }
}
```

What will be the output when `printNumber()` is called on a `NumberPrinter` object?
A) 0
B) 20
C) A memory address
D) A compile-time error

22. **Can this be used inside a lambda expression to refer to the current instance?**
A) No, this is not allowed in lambda expressions
B) Yes, it refers to the lambda expression itself
C) Yes, it refers to the enclosing instance of the class
D) Yes, it creates a new instance of an anonymous class

23. **In a scenario with multiple inner classes, how can you refer to the outer class's instance?**

A) Using `this`

B) Using `OuterClassName.this`

C) Using `super.this`

D) Using `InnerClass.this`

24. **Which statement correctly describes returning `this` from a method?**

A) It always returns a new object

B) It provides a reference to the current object for method chaining

C) It returns a copy of the object

D) It returns a reference to the superclass

25. **Examine the following code snippet:**

```java
public class Counter {
    int count = 0;
    public void increment() {
        this.count++;
        System.out.println("Count is: " + this.count);
    }
}
```

What will be the output after calling `increment()` twice on a `Counter` object?

A) "Count is: 1" followed by "Count is: 2"

B) "Count is: 0" twice

C) "Count is: 2" followed by "Count is: 4"

D) A compile-time error