1. What is the purpose of the `this` keyword in Java?
   A. To refer to a static variable
   B. To call the superclass constructor
   C. To refer to the current object
   D. To access local variables
2. Which of the following scenarios requires the use of the `this` keyword?
   A. Accessing a static method
   B. Resolving variable name conflicts between instance variables and method parameters
   C. Instantiating a class
   D. Throwing an exception
3. What does `this()` refer to in a constructor?
   A. A reference to the parent class constructor
   B. A call to another constructor in the same class
   C. A method call
   D. A static method in the class
4. Which of the following is true about `this` keyword in Java?
   A. `this` can refer to any object in memory
   B. `this` is used only in static methods
   C. `this` refers to the current class instance
   D. `this` can be used to define static blocks
5. In which method is the `this` keyword most commonly used to differentiate between instance variables and parameters?
   A. Static method
   B. Constructor
   C. Final method
   D. Abstract method
6. What will happen if `this()` is not the first statement in a constructor?
   A. Compilation will fail
   B. A runtime exception is thrown
   C. The constructor will execute normally
   D. `this()` will be ignored
7. Which of the following cannot be used inside a static method?
   A. Method overloading
   B. this
   C. Method overriding
   D. return statement
8. What does `this.variableName` do in a constructor when there's a local variable with the same name?

A. Refers to a method in the class
B. Refers to the static variable
C. Refers to the instance variable
D. Causes a syntax error

9. What does the following line do: `this(10);` inside a constructor?
   A. Assigns 10 to an instance variable
   B. Calls a method named `this` with parameter 10
   C. Calls another constructor of the same class with 10 as argument
   D. Creates a new object
10. Which statement about `this` is **incorrect**?
   A. `this` can be passed as an argument to another method
   B. `this` can be used to return the current class object
   C. `this` can be used in static contexts
   D. `this` can be used to call current class methods
11. Which of the following is a valid use of `this` in Java?
   A. `this = new Object();`
   B. `this.staticMethod();`
   C. `this.toString();`
   D. `this->method();`
12. What does the `this` keyword refer to when used inside a non-static inner class?
   A. The outer class instance
   B. The inner class instance
   C. A superclass
   D. The JVM
13. How can one refer to the outer class instance from a non-static inner class using `this`?
   A. `this.outer()`
   B. `OuterClass.this`
   C. `super.this`
   D. `this@OuterClass`
14. In method chaining, how is `this` keyword useful?
   A. It breaks the method chain
   B. It allows returning the same object for chaining
   C. It creates a new instance every time
   D. It disables constructors
15. Why can't `this` be used in static methods?
   A. Static methods are faster
   B. Static methods belong to objects
   C. Static methods do not have access to instance context
   D. `this` is reserved for final methods

Absolutely. Below is a **detailed and guided version** of the same 5 coding tasks related to the `this` **keyword in Java**, now with:

- Proper context
- Instructions
- Constructor requirement
- Guidance on where and how to use `this`

Scenario Questions

1. Employee Salary Assignment

Objective: Understand how to use `this` to resolve variable shadowing inside a method and constructor.

Task:

- Create a class named `Employee`.
- Declare two instance variables: `String name` and `double salary`.
- Write a constructor that takes `name` and `salary` as parameters and assigns them to the instance variables using `this`.
- Write a method named `setSalary(String name, double salary)` that updates the employee's details using the `this` keyword.
- Write a method `display()` to print the values.

---

2. Product Comparison

Objective: Learn how to compare the current object (`this`) with another object passed as a parameter.

Task:

- Create a class named `Product`.
- Declare two instance variables: `int id` and `double price`.
- Write a constructor to initialize both variables using the `this` keyword.
- Write a method `boolean isSame(Product p)` that returns `true` if the current object's `id` is the same as the passed object's `id`. Use `this.id` for comparison.
- Write a test method to create two products and check if they are the same.

---

3. Coordinate Printer

Objective: Practice using `this` to print instance variables and understand object identity.

Task:

- Create a class named `Point`.
- Declare two instance variables: `int x` and `int y`.
- Write a constructor to initialize these variables using `this.x` and `this.y`.
- Create a method `void print()` that prints the values of `x` and `y` using the `this` keyword.
- Inside the `print()` method, also print `this` to display the memory address of the current object.

---

4. Student Detail Updater

Objective: Understand how to update object data using `this`, and observe how `this` is required when parameter names match instance variable names.

Task:

- Create a class named `Student`.
- Declare instance variables: `int rollNo` and `String name`.
- Create a constructor to initialize both variables.
- Create a method `void updateDetails(String name)` that updates the student's name using the `this` keyword and prints both the old name and new name.
- Write a `display()` method to show student details.

---

5. Object Return for Method Chaining

Objective: Learn how `this` can be used to return the current object for chaining multiple method calls.

Task:

- Create a class named `Box`.
- Declare an instance variable: `int length`.
- Write a constructor to set the length using the `this` keyword.

- Create a method `Box setLength(int length)` that updates the length using `this.length = length` and returns `this`.
- Write another method `void display()` to print the length.
- Demonstrate method chaining like:

```
box.setLength(10).display();
```