



**this reference**

By Rahul Barve



## **'this' reference**

- Every class member function gets a hidden parameter known as `this` reference.
- `this` is a keyword in Java.
- It always refers to the object that is currently invoked.



## **'this' reference**

```
class Box {  
    int length, width, height;  
    Box() {  
        length = 10;  
        //Is equivalent to  
        //this.length = 10;  
        ...  
    }  
}
```

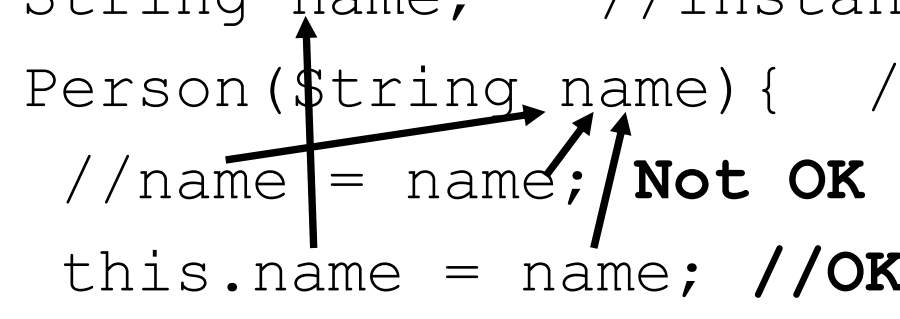


## **'this' reference**

- It becomes mandatory to use `this` reference when local variable names conflict with instance variable names.
- It is used to resolve the scope.

# 'this' reference

```
class Person {  
    String name;    //Instance Variable  
    Person(String name){    //Local Variable  
        //name = name; Not OK  
        this.name = name; //OK  
    }  
}
```





# Static Members

By Rahul Barve



# Static Members

- Whatever declarations, definitions are done within a class are collectively called as members of a class e.g. variables, methods and constructors.



# Static Members

- Members of a class are of 2 types:
  - Non-Static – These are associated with a particular instance of a class.
  - Static – These are not associated with any particular instance; rather they are associated with the whole class.





# Static Members

- Static members are either variables or methods.
- Constructors cannot be declared `static`.



# Static Members

- A static variable has a single copy irrespective of the number of objects created.
- Hence they are also known as class variables.



# Static Members

```
class Person {  
    //Instance Variables  
    String name;  
    int age;  
  
    //Class Variables  
    static float avgAge;  
    static int personCount;  
    ...  
}
```



# Static Members

- A `static` modifier is also used to introduce global variables.
- E.g.

```
public class Math {  
    public static float PI = 3.14f;  
    //Can be accessed from anywhere  
    //by using Math.PI  
}
```



# Static Members

- Like variables, methods can also be declared as `static`.
- Can be invoked without any object.



# Static Members

```
class Math {  
    static int getFactorial(int num) {  
        //Code to get factorial  
    }  
}
```

```
//int fact = Math.getFactorial(5);
```



# Static Members

- Static methods can access only static members.
- `this` reference is not accessible within static methods.
- `main()` is a static method because it is required to be called without object creation as it's an entry point of the application.



# Static Initialization Blocks

By Rahul Barve





# Static Initialization Blocks

- An arbitrary block of code.
- Executes when the class is loaded.
- Used for initializing static variables.



# Static Initialization Blocks

```
class MyClass {  
    static {  
        //Some Code  
    }  
}
```



# **Variables in Java**

By Rahul Barve



# Variables in Java

- Variables in Java are divided into 3 types:
  - Class Variables (Static Variables)
  - Instance Variables (Non-Static Variables)
  - Local Variables (Must be initialized before usage)



# Lets Summarize

- Classes and Objects
- Access Modifiers
- Methods, Overloading of Methods
- Constructors
- `this` Reference
- Static Members
- Types of Variables