`this` Keyword

- Reference variable that refers to the current object.

Proving `this` keyword

```
class A {
  void m() {
    System.out.println(this); // prints the reference id
  }

  public static void main(String args[]) {
    A obj = new A();
    System.out.println(obj); // prints the reference id
    obj.m();
  }
}
```

```
Output:
A@6f75e721
A@6f75e721
```

Usage of `this` keyword

Refer current class instance variable

- If there is ambiguity between the instance variables and parameters, `this` keyword resolves it.

Understanding the problem without `this` keyword

```
// Student.java
class Student {
  int rollNo;
  String name;

  Student(int rollNo, String name) {
    rollNo = rollNo;
    name = name;
  }

  void display() {
    System.out.println(rollNo + " " + name)
  }
}

// Main.java
public class Main {
  public static void main(String args[]) {
    Student s1 = new Student(101, "Parth");
```

```
    Student s2 = new Student(102, "Lily");

    s1.display();
    s2.display();
  }
}
```

```
Output:
0 null
0 null
```

- Here, the parameters in constructor and the instance variables are same.

Solution of the above problem

```
// Student.java
class Student {
  int rollNo;
  String name;

  Student(int rollNo, String name) {
    this.rollNo = rollNo; // Using this keyword
    this.name = name; // using this keyword
  }

  void display() {
    System.out.println(rollNo + " " + name)
  }
}

// Main.java
public class Main {
  public static void main(String args[]) {
    Student s1 = new Student(101, "Parth");
    Student s2 = new Student(102, "Lily");

    s1.display();
    s2.display();
  }
}
```

```
Output:
101 Parth
102 Lily
```

- If instance variables and formal parameters are same, `this` keyword needs to be used to resolve the ambiguity.
- It is better approach to use meaningful names for variables.
- So we use same name for instance variables and parameters in real time and always use `this` keyword.

Invoke current class method

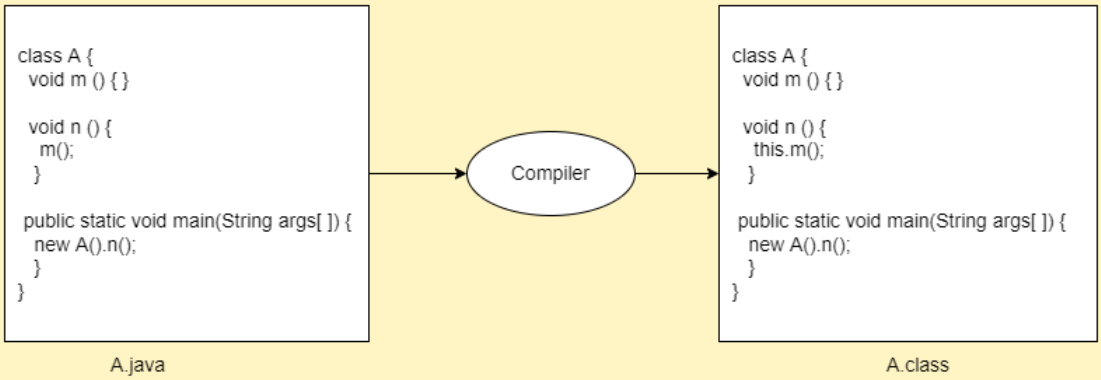
- We can invoke the current class method using `this` keyword.
- If we don't use `this` keyword, compiler automatically adds `this` keyword while invoking the method.

```
class A {
  void m() {
    System.out.println("Method m");
  }

  void n() {
    System.out.println("Method n");
    // m(); same as this.m()
    this.m();
  }
}

public class Main {
  public static void main(String args[]) {
    A obj  = new A();
    obj.n();
  }
}
```

```
Output:
Method n
Method m
```



`this()`

- It is used to invoke the current class constructor.
- Used for constructor chaining.
- Call to `this()` must be the first statement in constructor.

```
// Student.java
public class Student {
```

```
    int rollNo;
    String name;
    String course;
    double fee;

    Student(int rollNo, String name, String course) {
        this.rollNo = rollNo;
        this.name = name;
        this.course = course;
    }

    Student(int rollNo, String name, String course, double fee) {
        this(rollNo, name, course); // Reusing constructor
        this.fee = fee;
    }

    void display() {
        System.out.println(rollNo + " " + name + " " + course + " " + fee);
    }
}

// Main.java
public class Main {
      public static void main(String args[]) {
        Student s1 = new Student(101, "Parth", "Java");
        Student s2 = new Student(102, "Lily", "Python", 5000.0);

        s1.display();
        s2.display();
    }
}
```

```
Output:
101 Parth Java 0.0
102 Lily Python 5000.0
```