this keyword to refer current class instance variable

this keyword can be **used to refer current class instance variable.**

If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

a) **if we use same variables for instance variables and parameters has no effect**

b) If we use **different variables for instance variables and parameters has effect**

c) we can use **this keyword if we have same variables for instance variables and parameters**

```
package com.dl.thiskeyword.caseone;

if we use  same variables for instance variables and parameters has no effect
public class CreateAccount {

int userId; // instance v
String firstName;
String lastName;
String email;
long contactNo;

public CreateAccount(int userId, String firstName, String lastName, String email, long contactNo) {

userId = userId; // The assignment to variable userId has no effect
firstName = firstName;
lastName = lastName;
email = email;
contactNo = contactNo;
}
}
```

```
package com.dl.thiskeyword.casetwo;

If we use different variables for instance variables and parameters has effect

public class CreateAccount {

int userId; // instance v
String firstName;
String lastName;
String email;
long contactNo;

public CreateAccount(int uid, String fName, String lName, String mail, long cNo) {
userId = uid;
firstName = fName;
lastName = lName;
email = mail;
contactNo = cNo;
}
}
```

```
package com.dl.thiskeyword.casethree;

When we are using this keyword if we have same variables for instance variables and parameters
public class CreateAccount {

int userId; // instance v
String firstName;
String lastName;
String email;
long contactNo;

public CreateAccount(int userId, String firstName, String lastName, String email, long contactNo) {
this.userId = userId;
this.firstName = firstName;
this.lastName = lastName;
this.email = email;
this.contactNo = contactNo;
}
}
```

this keyword to invoke current class method

We can **invoke the method of the current class** by using this keyword.

If you don't use the this keyword, **compiler automatically adds this keyword while invoking the method**

```
package com.dl.thiskeyword.casefour;

public class Profile {
public void addToCart() {
System.out.println("Product Added to Cart");
}

public void get() {
System.out.println("Get the Products");
// this.addToCart();
addToCart(); // compiler automatically adds this keyword while invoking the method
}

public void remove() {
System.out.println("Remove the Products");
}

public static void main(String args[]) {
Profile p = new Profile();
p.get();
}
}
```

this() constructor to invoke current class constructor

this() constructor call can be used to invoke the current class constructor.

It is used to reuse the constructor.

In other words, it is used for constructor chaining

```
package com.dl.thiskeyword.casefour;

//this() : to invoke current class constructor
public class Student {
Student() {
System.out.println("Student Default Constructor");
}

Student(int id) {
this(); // calling other constructor of current class System.out.println(id);
}

public static void main(String args[]) {
new Student(101);
}
}
```

this keyword to pass as an argument in the method

this keyword can also be passed as an argument in the method.

```
package com.dl.thiskeyword.casefive;

//this keyword to pass as an argument in the method
public class Customer {

public void m1(Customer obj) {
System.out.println("M1 Method Invoked");
}

public void m2() {
m1(this);
}

public static void main(String args[]) {
Customer c = new Customer();
c.m2();
}
}
```

this keyword to pass as argument in the constructor call

We can pass this keyword in the constructor also.

It is useful if we have to use one object in multiple classes

```
package com.dl.thiskeyword.casesix;
//this: to pass as argument in the constructor call
public class ShopKeeper {

public void help() {

new Customer(this); // new Customer(ShopKeeper shopKepper)
new Owner(this); // new Owner(ShopKeeper shopKepper)
}

public static void main(String[] args) {
ShopKeeper c = new ShopKeeper();
c.help();
}
}
```

```
class Customer {

public Customer(ShopKeeper shopKepper) {
System.out.println("Customer Came for Prodcuts to buy");
}

}
```

```
class Owner {
public Owner(ShopKeeper shopKepper) {
System.out.println("Owner Came to get Shop Rent");
}
}
```

this keyword to pass as argument in the constructor call

We can pass this keyword in the constructor also.

It is useful if we have to use one object in multiple classes

```
public class A {

int id = 10;

public A() {
new B(this).IdNumber();
}

public static void main(String[] args) {
new A();
}

}
```

```
class B{

A  a;

public B(A a) {
this.a = a;
}

public void IdNumber() {
System.out.println(a.id);
}

}
```

this keyword can be used to return current class instance

We can **return this keyword as a statement from the method**.

In such case, return type of the method must be the class type (non-primitive).

```
return_type method_name(){
return this;
}
```

```
package com.dl.thiskeyword.caseseven;

//this keyword can be used to return current class instance
public class Product {

public Product get() {
Product p = new Product();
return p;
}

public Product update() {
return this;
}

public static void main(String[] args) {
Product product = new Product();
//returns different objects
System.out.println(product.get()); //
com.dl.thiskeyword.caseseven.Product@515f550a
System.out.println(product.get()); //
com.dl.thiskeyword.caseseven.Product@626b2d4a
System.out.println(product.get()); //
com.dl.thiskeyword.caseseven.Product@5e91993f

//returns same object
System.out.println(product.update()); //
com.dl.thiskeyword.caseseven.Product@1c4af82c
System.out.println(product.update()); //
com.dl.thiskeyword.caseseven.Product@1c4af82c
System.out.println(product.update()); //
com.dl.thiskeyword.caseseven.Product@1c4af82c
}
}
```

Proving this keyword

Let's prove that **this keyword refers to the current class instance variable.**

In this program, we are printing the reference variable and this, output of both variablesare same.

```
package com.dl.thiskeyword.caseeight;

//prove that this keyword refers to the current class instance variable
//prints same reference ID
public class Test {

public void m1() {
System.out.println(this); // com.dl.thiskeyword.caseeight.Test@515f550a
}

public static void main(String args[]) {
Test t = new Test();
System.out.println(t);// com.dl.thiskeyword.caseeight.Test@515f550a
t.m1();
}

}
```

Super Keyword:

Super keyword in Java is a **reference variable** which is used to refer **Parent** class object.

Super can be used to refer parent class **Variable**.
Super can be used to invoke parent class **Method**.
Super() can be used to invoke parent class **Constructor**.

```
package com.dl.superkeyword.variables;

class A{

int i = 10;
int j = 20;

}
class B extends A{

int i = 100;
int j = 200;

public B(int i, int j) {
System.out.println("Local Variables: " + i + " " + j);
System.out.println("Instance Variables Current Class: " + this.i + "  " + this.j);
System.out.println("Instance Variables Super Class: " + super.i + "  " + super.j);
}
}
public class Client {

public static void main(String[] args) {

new B(1000, 2000);
}
}
```

```java
package com.dl.superkeyword.methods;
class A {

public void m1() {

System.out.println("M1 Method of A Class");

}

}
```

```java
package com.dl.superkeyword.methods;
public class Client {

public static void main(String[] args) {

B b = new B();
b.m2();
}
}
```

```java
package com.dl.superkeyword.methods;
class B extends A {

public void m1() {
System.out.println("M1 Method of B Class");
}

public void m2() {
this.m1();
super.m1();
System.out.println("M2 Method of B Class");
}
}
```

```
M1 Method of B Class
M1 Method of A Class
M2 Method of B Class
```

```
package com.dl.superkeyword.constructors;

class A {
public A() {
System.out.println("Constructor A");
}

public A(int i) {
System.out.println("Construcotr with Parameters");
}
}

class B extends A {

public B() {
super(10);
System.out.println("Constructor B");
// super(10); // Constructor call must be the first statement in a constructor
}
}

public class Client {
public static void main(String[] args) {
new B();
}
}
```

```
Construcotr with Parameters
Constructor B
```