

Who Is this?

The this keyword



Who is "I"?

PAUSE AND TRY

Three students each say "I am first in line." Is the sentence true? Who does "I" mean each time?

Ana: "I am first in line."

Ben: "I am first in line."

Cy: "I am first in line."

Same words, a different speaker each time. A method is the same code, run for a different object each call.

By the end of class you can...

1

Explain that `this` holds a reference to the current object, and find which object it is in any call.

2

Use `this.name` to reach an instance variable that a parameter with the same name hides.

3

Pass `this` as an argument so another method can work with the current object.

4

Explain why class (static) methods, including `main`, have no `this`.

A white rounded square with a thin black border, containing the number '1' in a bold, black, sans-serif font.

1

PART 1

this is the current object

this holds a reference to the current object

THE CLASS

```
class Badge {
    private String owner;
    public Badge(String o) {
        owner = o;
    }
    public void show() {
        System.out.println(this.owner);
    }
}
```

IN MAIN (ANOTHER CLASS)

```
Badge red = new Badge("Ivy");
Badge blue = new Badge("Max");
blue.show();
red.show();
```

OUTPUT

Max
Ivy

Call	this refers to	this.owner
blue.show()	blue's object	"Max "
red.show()	red's object	" Ivy"

Testing what this holds

THE CLASS

```
class Badge {  
    private String owner;  
    public Badge(String o) {  
        owner = o;  
    }  
    public boolean isSame(Badge other) {  
        return this == other;  
    }  
}
```

IN MAIN (ANOTHER CLASS)

```
Badge red = new Badge("Ivy");  
Badge blue = new Badge("Ivy");  
Badge copy = blue;  
System.out.println(red.isSame(blue));  
System.out.println(blue.isSame(copy));
```

OUTPUT

```
false  
true
```



this is the reference before the dot. **other** is the reference in the parentheses. **this == other** is true only when they are the very same object.

this is not "the newest object"



To find **this**, look at the call that started the method, not at the order the objects were made.

FROM THE BADGE EXAMPLE

```
Badge red = new Badge("Ivy");
Badge blue = new Badge("Max");
blue.show();
red.show();
```

OUTPUT

Max
Ivy

Idea	True?
this is the object before the dot	Yes
this is the newest object	No: red.show() printed Ivy
this is the class	No: each object has its own owner

Whose coins?

PAUSE AND TRY

What are the two lines of output?

```
class Piggy {  
    private int coins;  
    public Piggy(int start) {  
        coins = start;  
    }  
    public void save(int amount) {  
        this.coins += amount;  
    }  
    public int getCoins() {  
        return coins;  
    }  
}
```

```
Piggy pig = new Piggy(10);  
Piggy hog = new Piggy(50);  
hog.save(5);  
pig.save(20);  
hog.save(1);  
System.out.println(pig.getCoins());  
System.out.println(hog.getCoins());
```

A large white square with rounded corners, tilted slightly to the right, containing the number '2' in a teal font.

2

PART 2

Reaching a hidden instance variable

this.name reaches past the parameter

WITHOUT THIS

```
class Book {  
    private String author;  
    public Book(String author) {  
        author = author;  
    }  
    public String getAuthor() {  
        return author;  
    }  
}
```

OUTPUT

null

WITH THIS

```
class Book {  
    private String author;  
    public Book(String author) {  
        this.author = author;  
    }  
    public String getAuthor() {  
        return author;  
    }  
}
```

OUTPUT

Lin

this.author is the object's variable. **author** alone is the parameter.

Both names in one line

THE CLASS

```
class Trip {  
    private int miles;  
    public Trip(int miles) {  
        this.miles = miles;  
    }  
    public void drive(int miles) {  
        this.miles = this.miles + miles;  
    }  
    public int getMiles() {  
        return miles;  
    }  
}
```

IN MAIN (ANOTHER CLASS)

```
Trip t = new Trip(100);  
t.drive(30);  
t.drive(12);  
System.out.println(t.getMiles());
```

OUTPUT

142

Call	miles (parameter)	this.miles after
drive(30)	30	130
drive(12)	12	142

Same name, two variables: **this.miles** is the object's, **miles** is the parameter.

this on the wrong side



The instance variable must be on the **left**, because the left side receives the value.

WRONG SIDE

```
public Book(String author) {  
    author = this.author;  
}
```

OUTPUT

null

RIGHT SIDE

```
public Book(String author) {  
    this.author = author;  
}
```

OUTPUT

Lin

Both versions use the same main: new Book("Lin"), then print getAuthor().

Did the tank fill up?

PAUSE AND TRY

What are the two lines of output?

```
class Tank {  
    private int fuel = 10;  
    public void refill(int amt) {  
        int fuel = this.fuel + amt;  
        System.out.println("Now " + fuel);  
    }  
    public int getFuel() {  
        return fuel;  
    }  
}
```

```
Tank t = new Tank();  
t.refill(20);  
System.out.println(t.getFuel());
```

A large, white, rounded square with a slight tilt, containing the number '3' in a bold, teal, sans-serif font.

3

PART 3

Passing this along

Pass this to share the current object

CLASS DOG

```
class Dog {  
    private boolean clean = false;  
    public void visit(Groomer g) {  
        g.groom(this);  
    }  
    public void wash() {  
        clean = true;  
    }  
    public boolean isClean() {  
        return clean;  
    }  
}
```

CLASS GROOMER

```
class Groomer {  
    public void groom(Dog d) {  
        d.wash();  
    }  
}
```

IN MAIN

```
Dog rex = new Dog();  
Groomer pat = new Groomer();  
rex.visit(pat);  
System.out.println(rex.isClean());
```

OUTPUT

true

Ask the other object, and pass yourself

THE CLASS

```
class Tree {  
    private int height;  
    public Tree(int height) {  
        this.height = height;  
    }  
    public boolean taller(Tree other) {  
        return this.height > other.height;  
    }  
    public boolean shorter(Tree other) {  
        return other.taller(this);  
    }  
}
```

IN MAIN (ANOTHER CLASS)

```
Tree oak = new Tree(30);  
Tree fir = new Tree(18);  
System.out.println(fir.shorter(oak));  
System.out.println(oak.shorter(fir));
```

OUTPUT

```
true  
false
```

other.taller(this): the roles of this and other swap.

Writing **this** as an argument passes a reference to the current object.

A large, bold, teal-colored number '4' is centered within a white square with rounded corners.

PART 4

Class methods have no this

No current object, no this

A STATIC METHOD USING THIS

```
class Tree {  
    private int height;  
    public Tree(int height) {  
        this.height = height;  
    }  
    public static int getHeight() {  
        return this.height;  
    }  
}
```

COMPILER SAYS

non-static variable this cannot be referenced from a static context

```
System.out.println(this.getHeight());
```

MAIN IS STATIC TOO

non-static variable this cannot be referenced from a static context

A class (static) method has **no this**. It can reach an object only through a parameter.

Which methods compile?

PAUSE AND TRY

Each method is added to a Tree class with private int height. Which compile?

A

```
public int getHeight() {  
    return this.height;  
}
```

B

```
public static int getHeight() {  
    return this.height;  
}
```

C

```
public static int size(Tree t) {  
    return t.height;  
}
```

D

```
public static void grow() {  
    this.height++;  
}
```

Words to know

this

A reference to the current object, inside an instance method or constructor.

Current object

The object a method was called on, or the one a constructor is building.

this.name

The current object's instance variable, even when a parameter has the same name.

Shadowed

An instance variable hidden by a parameter or local variable with its name.

Self-reference

An expression that uses this, such as `this == other` or `g.groom(this)`.

Static context

Inside a class method, where there is no current object and no this.

Say it in your own words



Explain to your partner what this means inside a method, and why a static method cannot use it.

SENTENCE STARTER

this refers to _____. A static method has no this because _____.

Next: the Who Is this? puzzle worksheet.