

Where Names Live

Scope and access



Two methods, one name

PAUSE AND TRY

Two methods in the same class each declare a variable named total. Do they clash? Can one method see the other's total?

METHOD 1

```
public int itemCost() {  
    int total = 4 * 3;  
    return total;  
}
```

METHOD 2

```
public int withTax() {  
    int total = 12 + 1;  
    return total;  
}
```

By the end of class you can...

- 1 Find the scope of any variable: where in the code its name can be used.
- 2 Explain why a local variable cannot be used outside its block, or remember values between calls.
- 3 Explain why parameters are local variables, and save a parameter in an instance variable.
- 4 Tell "cannot find symbol" from "has private access", and keep public and private off local variables.
- 5 Predict which variable a name means when a local name matches an instance variable.

A large, white, rounded square containing the number '1' in a bold, teal, sans-serif font.

1

PART 1

Scope and local variables

Scope: where a name can be used

THE CLASS

```
class Recipe {  
    private int servings;  
    public Recipe(int s) {  
        servings = s;  
    }  
    public void showFlour() {  
        int cups = servings * 2;  
        System.out.println(cups + " cups");  
    }  
    public void showEggs() {  
        int eggs = servings + 1;  
        System.out.println(eggs + " eggs");  
    }  
}
```

Variable	Declared in	Can be used
servings	class body	every method
s	constructor header	constructor only
cups	showFlour	rest of showFlour
eggs	showEggs	rest of showEggs



A **block** is code between { and }. A **local variable** can be used from its declaration to the end of its block. That region is its **scope**.

Reaching into another method

INSIDE RECIPE

```
public void showFlour() {  
    int cups = servings * 2;  
    System.out.println(cups + " cups");  
}  
public void showEggs() {  
    int eggs = cups / 2;  
    System.out.println(eggs + " eggs");  
}
```

COMPILER SAYS

```
error: cannot find symbol  
    symbol:   variable cups  
    location: class Recipe
```



cups belongs to showFlour. Its scope ends at that method's closing brace.



Java checks scope while compiling, before anything runs. "Symbol" means a name. To share a value between methods, store it in an **instance variable**.

A local variable starts over on every call

LOCAL: STARTS OVER

```
class Doorbell {  
    public int ring() {  
        int rings = 0;  
        rings++;  
        return rings;  
    }  
}
```

OUTPUT

```
1  
1  
1
```

INSTANCE: REMEMBERS

```
class Doorbell {  
    private int rings = 0;  
    public int ring() {  
        rings++;  
        return rings;  
    }  
}
```

OUTPUT

```
1  
2  
3
```

IN MAIN

```
Doorbell d = new Doorbell();  
System.out.println(d.ring());  
System.out.println(d.ring());  
System.out.println(d.ring());
```



If a value must still be there **next call**, it belongs in an instance variable, not a local variable.

Follow the cash

PAUSE AND TRY

What are the three lines of output?

```
class Wallet {  
    private int cash = 20;  
    public int spend(int cost) {  
        int left = cash - cost;  
        cash = left;  
        return left;  
    }  
    public int refund(int amount) {  
        int left = cash + amount;  
        return left;  
    }  
}
```

```
Wallet w = new Wallet();  
System.out.println(w.spend(5));  
System.out.println(w.refund(3));  
System.out.println(w.spend(5));
```

A large, white, rounded square containing the number '2' in a bold, teal, sans-serif font.

2

PART 2

Blocks inside methods

Inner blocks see out; outer blocks cannot see in

THE CLASS

```
class Scale {  
    public int heavier(int a, int b) {  
        int winner = a;  
        if (b > a) {  
            winner = b;  
        }  
        return winner;  
    }  
}
```

IN MAIN

```
Scale s = new Scale();  
System.out.println(s.heavier(3, 8));  
System.out.println(s.heavier(9, 2));
```

OUTPUT

8
9

METHOD BODY

a, b, winner

IF BODY

can use a, b, winner

A variable declared here ends at its }

return winner; is out here

Code in an inner block can use the outer block's variables, but not the other way around.

Same name: in a row is fine, nested is not

NESTED: DOES NOT COMPILE

```
int i = 5;
for (int i = 0; i < 3; i++) {
    System.out.println(i);
}
```

COMPILER SAYS

```
error: variable i is already defined in
method main(String[])
```

While a variable is in scope, an inner block cannot declare another with the same name.

ONE AFTER ANOTHER: FINE

```
for (int i = 0; i < 2; i++) {
    System.out.println(i);
}
for (int i = 5; i < 7; i++) {
    System.out.println(i);
}
```

OUTPUT

```
0
1
5
6
```

Which names work here?

PAUSE AND TRY

Which of **n**, **sum**, **k**, and **sq** can be used at spot A? At spot B?

```
public int total(int n) {  
    int sum = 0;  
    for (int k = 1; k <= n; k++) {  
        int sq = k * k;  
        sum += sq;  
        // spot A  
    }  
    // spot B  
    return sum;  
}
```

SPOT A

n sum k sq

SPOT B

n sum k sq



3

PART 3

Parameters and access

A parameter is a local variable

THE CLASS

```
class Plant {  
    private int height;  
    public Plant(int start) {  
        height = start;  
    }  
    public void grow() {  
        height = height + start;  
    }  
}
```

COMPILER SAYS

```
error: cannot find symbol  
    symbol:   variable start  
    location: class Plant
```



start exists only while the constructor runs.



A **parameter** is a local variable of its constructor or method. A new copy is made on every call, and it is gone when the body ends.

Save the parameter in an instance variable

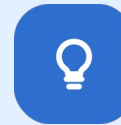
```
class Plant {  
    private int height;  
    private int step;  
    public Plant(int start) {  
        height = start;  
        step = start;  
    }  
    public void grow() {  
        height = height + step;  
    }  
    public int getHeight() {  
        return height;  
    }  
}
```

IN MAIN

```
Plant p = new Plant(4);  
p.grow();  
p.grow();  
System.out.println(p.getHeight());
```

OUTPUT

12



If other methods need a parameter's value, the constructor must **store it** in an instance variable.

Two different checks

THE CLASS

```
class Game {  
    private int score = 40;  
    public int getScore() {  
        return score;  
    }  
}
```

Code	Compiler says
<code>g.score</code> (in main)	<code>score</code> has private access in <code>Game</code>
<code>score</code> (in main)	cannot find symbol
<code>private int extra = 5;</code> (inside a method)	illegal start of expression



Scope decides where a name is visible. **public** and **private** decide which classes may use an instance variable or method. Local variables and parameters never take public or private.

A large white square with rounded corners, tilted slightly to the right, containing a large teal number '4' in the center.

4

PART 4

Same name, two variables

The closer declaration wins

PARAMETER NAMED TITLE

```
class Song {  
    private String title;  
    public Song(String title) {  
        title = title;  
    }  
    public String getTitle() {  
        return title;  
    }  
}
```

OUTPUT

null

PARAMETER NAMED T

```
class Song {  
    private String title;  
    public Song(String t) {  
        title = t;  
    }  
    public String getTitle() {  
        return title;  
    }  
}
```

OUTPUT

Sunrise

When a local variable or parameter has an instance variable's name, the name means the **local one** inside that method. It **shadows** the instance variable.

How fast is the car?

PAUSE AND TRY

What does this print?

```
class Car {  
    private int speed = 10;  
    public void setSpeed(int speed) {  
        speed = speed;  
    }  
    public void boost() {  
        int speed = 99;  
        speed++;  
    }  
    public int getSpeed() { return speed; }  
}
```

```
Car c = new Car();  
c.setSpeed(50);  
c.boost();  
System.out.println(c.getSpeed());
```

Words to know

Scope

The part of a program where a variable's name can be used.

Block

Code between a matching { and }: a method, loop, or if body.

Local variable

Declared in a block; usable only to the end of that block.

Parameter

A local variable of its method or constructor, set from the argument.

Access modifier

public or private on instance variables, constructors, and methods.

Shadowing

A local name that matches an instance variable hides it in that method.

Where does it live?



A game needs a high score that survives every round, and a counter used only while one round is scored. Which is local, and which is an instance variable? Explain to your partner.

SENTENCE STARTER

_____ is local because it only needs to last _____. _____ is an instance variable because _____.

Next: the Where Names Live puzzle worksheet.