

One Copy for Everyone

Class variables and methods



Whose number is it?

PAUSE AND TRY

A school has three buses. Which of these facts belongs to one bus, and which belongs to all of them?

The driver's name

Riders on this bus today

Riders on all buses today

The number of seats per bus (same for every bus)

By the end of class you can...

- 1 Declare a class variable with static and explain why every object shares one copy.
- 2 Use a public class variable from another class with the class name and a dot.
- 3 Use final to make a value that cannot change, and write a class constant.
- 4 Write class methods that read and change class variables and call other class methods.
- 5 Explain why a class method needs an object passed in to use instance data.

A large white square with rounded corners, tilted slightly to the right, containing a large blue number '1'.

PART 1

One shared copy

Instance variable or class variable?

Instance variable

ONE PER OBJECT

Every object gets its own copy. Changing one object's copy leaves the others alone.

```
private int riders;
```

Class variable

ONE PER CLASS

Declared with static. The class has exactly one copy, and every object shares it.

```
private static int totalRiders = 0;
```



Ask: **"Is this value about one object, or about all of them?"** If it is about all of them, it is a class variable.

Two buses, one total

THE CLASS

```
class Bus {  
    private int riders;  
    private static int totalRiders = 0;  
  
    public void board(int n) {  
        riders += n;  
        totalRiders += n;  
    }  
    public int getRiders() {  
        return riders;  
    }  
    public int getAll() {  
        return totalRiders;  
    }  
}
```

IN MAIN (ANOTHER CLASS)

```
Bus north = new Bus();  
Bus south = new Bus();  
north.board(10);  
south.board(6);  
north.board(4);  
System.out.println(north.getRiders());  
System.out.println(south.getRiders());  
System.out.println(south.getAll());
```

OUTPUT

```
14  
6  
20
```

20 comes from south, but it counts north's 14 too. There is only one totalRiders.

A counter without static counts to 1

WITHOUT STATIC

```
class Pet {  
    private int petCount = 0;  
  
    public Pet() {  
        petCount++;  
    }  
    public int getCount() {  
        return petCount;  
    }  
}
```

IN MAIN

```
Pet rex = new Pet();  
Pet tom = new Pet();  
Pet kit = new Pet();  
System.out.println(kit.getCount());
```

WITH STATIC

```
class Pet {  
    private static int petCount = 0;  
  
    public Pet() {  
        petCount++;  
    }  
    public int getCount() {  
        return petCount;  
    }  
}
```



Prints 1 without static: every Pet has its **own** counter, so each one only counts itself. Prints 3 with static. No error warns you.

Sip, sip, sip

PAUSE AND TRY

What are the two lines of output?

```
class Cup {  
    private int sips = 0;  
    private static int allSips = 0;  
    public void sip() {  
        sips++;  
        allSips++;  
    }  
    public int getSips() { return sips; }  
    public int getAll() { return allSips; }  
}
```

```
Cup a = new Cup();  
Cup b = new Cup();  
a.sip();  
b.sip();  
b.sip();  
System.out.println(a.getSips());  
System.out.println(a.getAll());
```

A large, bold, teal-colored number '2' is centered within a white rounded square that has a slight 3D effect and a shadow.

PART 2

From outside the class, and final

Class name, dot, variable name

THE CLASS

```
class Score {  
    public static int highScore = 0;  
  
    public Score(int points) {  
        if (points > highScore) {  
            highScore = points;  
        }  
    }  
}
```

IN MAIN (ANOTHER CLASS)

```
System.out.println(Score.highScore);  
Score a = new Score(70);  
Score b = new Score(95);  
Score c = new Score(80);  
System.out.println(Score.highScore);
```

OUTPUT

```
0  
95
```



A **public** class variable is used from another class as **ClassName.variableName**. Inside its own class, the name alone works. It exists even before any object is made.

final: a value that cannot change

```
public static final int MAX_SIZE = 5;
```

public anyone may read it

static one copy, for the class

final it can never change

IN MAIN

```
Team bears = new Team(3);  
System.out.println(Team.MAX_SIZE);  
System.out.println(bears.openSpots());
```

THE CLASS

```
class Team {  
    public static final int MAX_SIZE = 5;  
    private int players;  
    public Team(int p) {  
        players = p;  
    }  
    public int openSpots() {  
        return MAX_SIZE - players;  
    }  
}
```

OUTPUT

```
5  
2
```

public does not mean "anyone can change it"



public decides who may **read** it. **final** stops **everyone** from changing it, including its own class.

IN MAIN

```
Team.MAX_SIZE = 6;
```

COMPILER SAYS

cannot assign a value to static final variable
MAX_SIZE

Declaration	Read from main?	Change from main?
public static int count	Yes: Team.count	Yes
private static int count	No: private access	No
public static final int MAX	Yes: Team.MAX	No: final

A large, white, rounded square containing a bold, teal number '3'.

3

PART 3

Writing class methods

Class methods work on class data

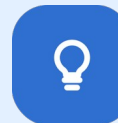
```
class Arcade {  
    public static final double FEE = 0.25;  
    private static int tokensUsed = 0;  
  
    public static void play(int tokens) {  
        tokensUsed += tokens;  
    }  
  
    public static int getUsed() {  
        return tokensUsed;  
    }  
  
    public static double spent() {  
        return getUsed() * FEE;  
    }  
}
```

IN MAIN (ANOTHER CLASS)

```
Arcade.play(8);  
Arcade.play(4);  
System.out.println(Arcade.getUsed());  
System.out.println(Arcade.spent());
```

OUTPUT

```
12  
3.0
```



A **class method** has static in its header. It can read and change class variables and call other class methods.

Score the quiz

PAUSE AND TRY

What prints? Why is no Quiz object ever needed?

```
class Quiz {  
    private static int asked = 0;  
    private static int points = 0;  
    public static void answer(int s) {  
        asked++;  
        points += s;  
    }  
    public static int avg() {  
        return points / asked;  
    }  
}
```

```
Quiz.answer(9);  
Quiz.answer(6);  
Quiz.answer(8);  
System.out.println(Quiz.avg());
```

A large, bold, teal number '4' is centered within a white rounded square that has a slight 3D effect and a shadow.

PART 4

What class methods cannot reach

A class method has no object of its own

THE CLASS

```
class Dog {  
    private String name;  
  
    public Dog(String n) {  
        name = n;  
    }  
  
    public static String greet() {  
        return "Hi, " + name;  
    }  
}
```

```
System.out.println(Dog.greet());
```

COMPILER SAYS

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

Inside a class method...	Allowed?
Use a class variable	Yes
Call another class method	Yes
Use an instance variable by name	No
Call an instance method by name	No
Use them through an object parameter	Yes

Hand the class method an object

```
class Dog {  
    private String name;  
    private int age;  
    public Dog(String n, int a) {  
        name = n;  
        age = a;  
    }  
    public static String greet(Dog d) {  
        return "Hi, " + d.name;  
    }  
    public static Dog older(Dog a, Dog b) {  
        if (a.age >= b.age) {  
            return a;  
        }  
        return b;  
    }  
}
```

IN MAIN (ANOTHER CLASS)

```
Dog max = new Dog("Max", 4);  
Dog bo = new Dog("Bo", 7);  
System.out.println(Dog.greet(max));  
Dog big = Dog.older(max, bo);  
System.out.println(Dog.greet(big));
```

OUTPUT

```
Hi, Max  
Hi, Bo
```



Through a parameter, a class method may use **d.name** or **a.age**. Now Java knows which dog.

Which bodies compile?

PAUSE AND TRY

Dog has: `private int age;` `private static int dogCount;` `getAge()` (instance); `countDogs()` (static). Which can go in a static method with no parameters?

A `return dogCount;`

B `return age;`

C `return getAge();`

D `return countDogs();`

Words to know

Class variable

Declared static. One copy that belongs to the class, shared by all objects.

Instance variable

Each object has its own copy.

static

Makes a variable or method belong to the class, not to one object.

final

The variable cannot be changed after it gets its value.

Class constant

A public static final variable, named in capitals.

Class method

Has static in its header. Called with the class name; no object needed.

One object, or all of them?



A music app has a Song class. Which should be static: each song's title, or the number of songs in the app? Explain to your partner.

SENTENCE STARTER

_____ should be static because it describes _____, not one _____.

Next: the One Copy for Everyone puzzle worksheet.