

Give It Back

Methods: how to write them



Ask it, or tell it?

PAUSE AND TRY

Which requests give you an answer back? Which ones change the bottle?

THINGS YOU CAN ASK A WATER BOTTLE

1. How many ounces are left?
2. Drink 10 ounces.
3. Are you empty?
4. Refill to the top.

A method is how an object answers a request. Which requests need the object to hand something back?

By the end of class you can...

- 1 Write a void method and call it as a statement on its own.
- 2 Write a mutator that uses its parameters to change instance variables.
- 3 Write a non-void method whose return value matches its return type.
- 4 Write an accessor and explain why the caller gets a copy.
- 5 Trace exactly where a method stops when return runs.
- 6 Explain why changing a primitive parameter never changes the argument.

A large, bold, black number '1' is centered within a white rounded square that has a slight 3D effect and a shadow.

1

PART 1

Void methods that change things

Two kinds of method headers

VOID: HANDS NOTHING BACK

HEADER

```
public void drink(int amount)
```

CALL

```
b.drink(10);
```

The call is a statement on its own line.

NON-VOID: RETURNS ONE VALUE

HEADER

```
public int getOunces()
```

CALL

```
int left = b.getOunces();
```

The call is a value: print it or store it.



Every header needs a **return type** before the name: **void** or the type of the one value that comes back.

A mutator changes the object

WATERBOTTLE.JAVA

```
class WaterBottle {  
    private int ounces;  
    public WaterBottle(int start) {  
        ounces = start;  
    }  
    public void drink(int amount) {  
        ounces = ounces - amount;  
        if (ounces < 0) {  
            ounces = 0;  
        }  
    }  
    public int getOunces() {  
        return ounces;  
    }  
}
```

MAIN

```
WaterBottle b = new WaterBottle(24);  
b.drink(10);  
System.out.println(b.getOunces());  
b.drink(30);  
System.out.println(b.getOunces());
```

OUTPUT

```
14  
0
```

Mutator: a method that changes instance variables.
Often void.

Changing the parameter changes nothing



A parameter disappears when the method ends. A mutator must put its result in an **instance variable**.

BUG: STORES IN THE PARAMETER

```
public void addTime(int coins) {  
    coins = minutes + coins * 15;  
}
```

AFTER ADDTIME(2), GETMINUTES() RETURNS

0

FIXED: STORES IN MINUTES

```
public void addTime(int coins) {  
    minutes = minutes + coins * 15;  
}
```

AFTER ADDTIME(2), GETMINUTES() RETURNS

30

Both start with minutes at 0.

Turn it up

PAUSE AND TRY

What does main print?

```
class Dimmer {  
    private int level;  
    public void up(int step) {  
        level = level + step;  
        if (level > 10) {  
            level = 10;  
        }  
    }  
    public int getLevel() {  
        return level;  
    }  
}
```

MAIN

```
Dimmer d = new Dimmer();  
d.up(3);  
d.up(9);  
System.out.println(d.getLevel());  
d.up(-4);  
System.out.println(d.getLevel());
```

YOUR OUTPUT

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

2

PART 2

Methods that hand a value back

The call becomes the returned value

RECTANGLE.JAVA

```
class Rectangle {  
    private int width;  
    private int height;  
    public Rectangle(int w, int h) {  
        width = w;  
        height = h;  
    }  
    public int area() {  
        return width * height;  
    }  
    public double ratio() {  
        return (double) width / height;  
    }  
}
```

MAIN

```
Rectangle r = new Rectangle(6, 4);  
int a = r.area();  
System.out.println(a + 1);  
System.out.println(r.ratio());
```

OUTPUT

```
25  
1.5
```

return evaluates the expression and sends back its **value**. It must fit the return type.

KEY IDEA

An accessor hands back a copy

MAIN

```
WaterBottle b = new WaterBottle(24);  
int left = b.getOunces();  
left = 0;  
System.out.println(left + " " + b.getOunces());
```

OUTPUT

```
0 24
```



Accessor: a non-void method that returns a copy of an instance variable's value. Changing the copy never changes the object.

TRYING TO CHANGE THE VALUE THROUGH THE CALL

```
b.getOunces() = 0;
```

COMPILER SAYS

```
error: unexpected type
```

A call is a value, not a variable.



The call must match the return type

USING A VOID CALL AS A VALUE

```
System.out.println(b.drink(5));
```

COMPILER SAYS

```
error: 'void' type not allowed here
```

Fix: `b.drink(5);` on its own line.

RETURNING A DOUBLE FROM AN INT METHOD

```
public int half() {  
    return ounces / 2.0;  
}
```

COMPILER SAYS

```
error: incompatible types: possible  
lossy conversion from double to int
```

Fix: return type `double`, or `ounces / 2`.



Ask two questions: does this method return anything? If so, does the returned value fit the **return type**?

Pick the header

PAUSE AND TRY

Which header fits this body inside `WaterBottle`?

```
????? {  
    return ounces > 0;  
}
```

A `public void hasWater()`

B `public int hasWater()`

C `public boolean hasWater()`

D `public hasWater()`

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

PART 3

Return ends the method

The first return that runs wins

LIBRARYBOOK.JAVA

```
class LibraryBook {  
    private int daysLate;  
    public LibraryBook(int days) {  
        daysLate = days;  
    }  
    public int fine() {  
        if (daysLate <= 0) {  
            return 0;  
        }  
        if (daysLate > 20) {  
            return 500;  
        }  
        return daysLate * 25;  
    }  
}
```

MAIN

```
LibraryBook early = new LibraryBook(-2);  
LibraryBook late = new LibraryBook(3);  
LibraryBook lost = new LibraryBook(30);  
System.out.println(early.fine());  
System.out.println(late.fine());  
System.out.println(lost.fine());
```

OUTPUT

```
0  
75  
500
```

A return inside an if ends the whole method.

A return inside a loop

INSIDE CLASS SAVINGS (WEEKLY IS 15)

```
public int weeksUntil(int goal) {  
    int total = 0;  
    for (int week = 1; week <= 52; week++) {  
        total = total + weekly;  
        if (total >= goal) {  
            return week;  
        }  
    }  
    return -1;  
}
```

```
Savings s = new Savings(15);  
System.out.println(s.weeksUntil(100));  
System.out.println(s.weeksUntil(1000));
```

OUTPUT

7
-1

week	total	>= 100?
1	15	no
...	...	no
6	90	no
7	105	yes: return 7

Goal 1000 is never reached, so return -1 runs.

WATCH OUT

Every path needs a return, and nothing after it

CODE AFTER RETURN

```
public int yearly() {  
    return weekly * 52;  
    System.out.println("done");  
}
```

COMPILER SAYS

error: unreachable statement

Fix: move the println above the return.

A PATH WITH NO RETURN

```
public boolean isRich() {  
    if (weekly > 100) {  
        return true;  
    }  
}
```

COMPILER SAYS

error: missing return statement

Fix: add return false; after the if.

A non-void method must return a value on **every** path through it.



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

PART 4

Parameters get copies

Changing a parameter never changes the argument

ARCADE.JAVA

```
class Arcade {  
    private int tickets;  
    public Arcade() {  
        tickets = 0;  
    }  
    public void win(int points) {  
        points = points / 10;  
        tickets = tickets + points;  
    }  
    public int getTickets() {  
        return tickets;  
    }  
}
```

MAIN

```
Arcade a = new Arcade();  
int score = 250;  
a.win(score);  
int t = a.getTickets();  
System.out.println(score + " " + t);
```

OUTPUT

250 25

score (main)

250

points (win)

250 then 25

Two separate variables. Only the copy changes.

Two copies in one program

PAUSE AND TRY

What does main print? Find every copy.

```
class Pump {  
    private int gallons;  
    public Pump(int start) {  
        gallons = start;  
    }  
    public void fill(int amount) {  
        gallons = gallons + amount;  
        amount = 0;  
    }  
    public int getGallons() {  
        return gallons;  
    }  
}
```

MAIN

```
Pump p = new Pump(3);  
int extra = p.getGallons();  
extra = extra * 2;  
p.fill(extra);  
int g = p.getGallons();  
System.out.println(extra + " " + g);
```

YOUR OUTPUT

Words to know

Void method

Returns nothing; void comes before its name. Its call is a statement.

Non-void method

Returns exactly one value; its return type replaces void.

Mutator

Changes an object's instance variables. Often void.

Accessor

Non-void; returns a copy of an instance variable's value.

Return by value

return evaluates its expression and sends back that value.

Parameter

Receives a copy of a primitive argument's value.

Give it back, or change it?



Explain to your partner the difference between a mutator and an accessor, with a header for each.

SENTENCE STARTER

A mutator _____ the object and is usually _____, but an accessor returns _____.

Next: the Give It Back puzzle worksheet.