

Starting State

Constructors



A brand-new bike

PAUSE AND TRY

A bike-share app adds a new bike. What facts should it start with, and where does each value come from?

FACTS THE APP TRACKS

1. ID number
2. Battery level, 0 to 100
3. Rented right now?

Which facts are the same for every new bike? Which one must be told to the app?

By the end of class you can...

- 1 Describe an object's state and the has-a relationship with its instance variables.
- 2 Write a constructor that gives every instance variable a starting value.
- 3 Explain what happens in memory when new calls a constructor.
- 4 Use parameters, named differently from instance variables, to supply values.
- 5 State the default values and when Java supplies a default constructor.
- 6 Explain why a constructor should copy a mutable object it receives.

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

PART 1

State and the constructor

An object's state is its values right now



An object's **state** is the values of its instance variables at a given moment. The object **has a** value for each one: a **has-a relationship**.



The instance variables come from the class and are the same for every bike. The **values** differ from object to object and change over time.

Instance variable	Bike 417	Bike 52
<code>int id</code>	417	52
<code>double battery</code>	100.0	38.5
<code>boolean rented</code>	false	true

A bike has an ID. A bike has a battery level. A bike has a rented status. Each "has a" is an instance variable.

Anatomy of a constructor

BIKE.JAVA

```
class Bike {  
    private int id;  
    private double battery;  
    private boolean rented;  
  
    public Bike(int bikeId) {  
        id = bikeId;  
        battery = 100.0;  
        rented = false;  
        System.out.println(id + " " + battery);  
    }  
}
```

public

Constructors are always public in this course.

Bike

Exactly the class name.

no return type

Not even void. It always builds its own class.

(int bikeId)

Parameters: data the caller passes in.

{ ... }

Gives every instance variable a value.

What new Bike(417) does

MAIN

```
System.out.println("Start");  
Bike rental = new Bike(417);  
System.out.println("Done");
```

OUTPUT

```
Start  
417 100.0  
Done
```

The constructor finishes before main moves on.

1

Memory is set aside for a new Bike object, with its own id, battery, and rented.

2

The constructor body runs. bikeId holds a copy of 417, and every instance variable gets a value.

3

A reference to the new object is handed back and stored in rental.

A return type turns it into a method



A constructor has **no return type**, not even void. Put one in front and Java no longer treats it as a constructor.

BUS.JAVA

```
class Bus {  
    private int route;  
  
    public void Bus(int routeNumber) {  
        route = routeNumber;  
    }  
}
```

```
Bus morning = new Bus(12);
```

COMPILER SAYS

```
error: constructor Bus in class Bus  
cannot be applied to given types;  
    required: no arguments  
    found:    int
```

Fix: delete void, so the header is public Bus(int routeNumber).

Two tickets, one constructor

PAUSE AND TRY

What does this program print?

```
class Ticket {  
    private int row;  
    private double price;  
  
    public Ticket(int seatRow) {  
        row = seatRow;  
        if (row <= 5) {  
            price = 40.0;  
        } else {  
            price = 25.0;  
        }  
        System.out.println(row + " " + price);  
    }  
}
```

MAIN

```
Ticket a = new Ticket(3);  
Ticket b = new Ticket(12);
```

YOUR OUTPUT

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

2

PART 2

Parameters supply the values

From parameter to instance variable

```
class Backpack {  
    private String name;  
    private int pockets;  
    private boolean heavy;  
  
    public Backpack(String owner, double pounds) {  
        name = owner;  
        pockets = 4;  
        heavy = pounds > 15.0;  
        System.out.println(name + " " + pockets  
                             + " " + heavy);  
    }  
}
```

```
Backpack a = new Backpack("Ana", 12.5);  
Backpack r = new Backpack("Raj", 18.0);
```

Variable	Gets its value
name	copied from owner
pockets	fixed: always 4
heavy	computed from pounds

OUTPUT

```
Ana 4 false  
Raj 4 true
```

A parameter vanishes when the constructor ends. **Store it** or lose it.

Three ways to never set an instance variable

SAME NAME

```
public Phone(int battery) {  
    battery = battery;  
}
```

Both sides mean the parameter, so it is copied into itself.

stays 0

BACKWARD

```
public Phone(int level) {  
    level = battery;  
}
```

= stores right into left, so this overwrites the parameter.

stays 0

NEW LOCAL VARIABLE

```
public Phone(int level) {  
    int battery = level;  
}
```

Declares a new variable that vanishes when the constructor ends.

stays 0

```
public Phone(int level) {  
    battery = level;  
}
```

Correct: a different name, with the instance variable on the left of =. `new Phone(80)` stores 80.

Which one stores the volume?

PAUSE AND TRY

Speaker has an instance variable volume. Which constructor sets it?

- A** `public Speaker(int volume) { volume = volume; }`
- B** `public Speaker(int level) { volume = level; }`
- C** `public void Speaker(int level) { volume = level; }`
- D** `public Speaker(int level) { level = volume; }`



3

PART 3

Defaults and the default constructor

KEY IDEA

Every instance variable starts with a default

Type	Default
int	0
double	0.0
boolean	false
any reference type	null

MAIN RUNS

```
Notebook b = new Notebook("Art");
```

OUTPUT

```
0 0.0  
false null
```

```
class Notebook {  
    private String subject;  
    private int pages;  
    private double price;  
    private boolean spiral;  
    private String owner;  
  
    public Notebook(String topic) {  
        subject = topic;  
        System.out.println(pages + " " + price);  
        System.out.println(spiral + " " + owner);  
    }  
}
```

A constructor should still give **every** instance variable a value on purpose.



KEY IDEA

The default constructor

NO CONSTRUCTOR WRITTEN

```
class Timer {  
    private int seconds;  
  
    public void show() {  
        System.out.println(seconds);  
    }  
}
```

MAIN

```
Timer t = new Timer();  
t.show();
```

0

ONE CONSTRUCTOR WRITTEN

```
class Timer {  
    private int seconds;  
  
    public Timer(int start) {  
        seconds = start;  
    }  
}
```

MAIN

```
Timer t = new Timer();
```

```
error: constructor Timer in class  
Timer cannot be applied to given types;
```

Two constructors, one forgets

PAUSE AND TRY

What does this print? Which constructor runs each time?

```
class Plant {  
    private int height;  
    private boolean wet;  
    public Plant(int inches) {  
        height = inches;  
        wet = true;  
    }  
    public Plant() {  
        height = 5;  
    }  
    public void show() {  
        System.out.println(height + " " + wet);  
    }  
}
```

MAIN

```
Plant a = new Plant(8);  
Plant b = new Plant();  
a.show();  
b.show();
```

YOUR OUTPUT

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

PART 4

Mutable objects as parameters

Copying a reference is not copying the object

```
class Quiz {  
    private Scanner source;  
    private String first;  
  
    public Quiz(Scanner wordList) {  
        source = wordList;  
        first = source.next();  
        System.out.println("Quiz: " + first);  
    }  
}
```

MAIN

```
Scanner words = new Scanner("cat dog fox");  
Quiz q = new Quiz(words);  
System.out.println("main: " + words.next());
```

OUTPUT

```
Quiz: cat  
main: dog
```



An object whose state can change is **mutable**. `source = wordList`; copies the **reference**, so the quiz and main share one Scanner. A read through either one moves it for both.

Store a copy of a mutable object



When a constructor receives a mutable object, initialize the instance variable with **a copy**, not the reference itself.

```
public Order(Address destination) {  
    shipTo = new Address(destination);  
    System.out.println(shipTo == destination);  
}
```

COPY (DO THIS)

Output: `false`

```
public Shared(Address destination) {  
    shipTo = destination;  
    System.out.println(shipTo == destination);  
}
```

SHARE (AVOID)

Output: `true`

Address has a constructor, `Address(Address original)`, that builds a new Address with the same street.

Words to know

State

The values of an object's instance variables at a given moment.

Has-a relationship

An object has each of its instance variables: a bike has an ID.

Constructor

Class name, no return type; sets a new object's initial state.

Default value

0, 0.0, false, or null, before any code assigns a value.

Default constructor

The no-parameter one Java supplies when a class has none.

Mutable

An object whose state can change after it is built.

Born ready



Tell your partner what a constructor must do, and one way it can go wrong.

SENTENCE STARTER

A constructor gives _____ a starting value. If it skips one, that variable holds _____.

Next: the Starting State puzzle worksheet.