

Under Construction

Object creation and storage (instantiation)



You have typed this line for weeks

PAUSE AND TRY

What do you think each part of this line does?

```
Scanner console = new Scanner(System.in);
```

Scanner

?

console

?

new

?

Scanner(System.in)

?

By the end of class you can...

- 1 Create an object with new and a call to one of the class's constructors.
- 2 Declare a variable of the right type to hold a reference, or null when there is no object.
- 3 Read a constructor header and write its signature.
- 4 Decide which overloaded constructor a call runs, and whether the arguments match.
- 5 Trace a constructor call: copied argument values, the jump in, and the return.

1

PART 1

new and constructors

Building an object takes new and a constructor

```
String greeting = new String("Good morning");
```

Piece	What it is
String greeting	Declares a variable that can hold a reference to a String
new	Asks Java to build a brand-new object
String(...)	The constructor call: same name as the class
"Good morning"	The value passed in: the new object's starting characters



A **constructor** is code in a class that builds a new object. It has **the same name as its class**. Creating an object is called **instantiation**.

Same shape, different class

Type	Name	=	new	Constructor call
String	greeting	=	new	String("Good morning");
Scanner	console	=	new	Scanner(System.in);
Scanner	reader	=	new	Scanner("85 92");



The pattern never changes: **Type name = new Type(...);** The variable's type matches the class you are building.

Forgetting new



A constructor only builds an object when **new** comes right before it.

DOES NOT COMPILE

```
String greeting = String("Good morning");  
System.out.println(greeting);
```

FIXED

```
String greeting =  
    new String("Good morning");
```

COMPILER SAYS

```
error: cannot find symbol  
  symbol:   method String(String)
```

OUTPUT

```
Good morning
```

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

2

PART 2

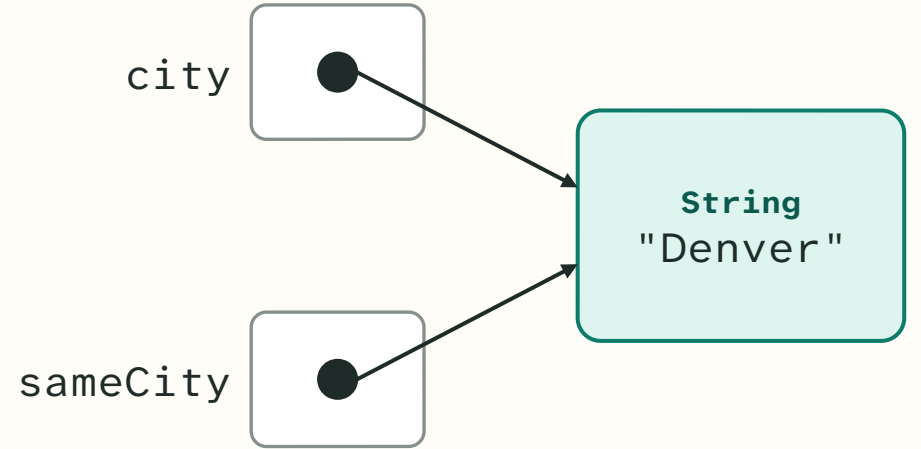
Where the object lives

Two variables, one object

```
String city = new String("Denver");  
String sameCity = city;  
System.out.println(city);  
System.out.println(sameCity);
```

OUTPUT

```
Denver  
Denver
```



one object, two arrows

A reference variable holds a **reference** (think: the object's address), not the object. Count the **news** to count the objects.

WATCH OUT

An empty object is not null

```
String empty = new String();  
String missing = null;  
System.out.print("[");  
System.out.print(empty);  
System.out.println("]");  
System.out.print("[");  
System.out.print(missing);  
System.out.println("]");
```

OUTPUT

```
[]  
[null]
```



new String() is an object with no characters. **null** means there is no object, and no constructor ran.

DOES NOT COMPILE

```
int count = null;
```

incompatible types: <null> cannot be converted to int

Which ones compile?

PAUSE AND TRY

Assume `import java.util.Scanner;` is at the top. Which declarations compile?

- A** `String title = null;`
- B** `double price = null;`
- C** `Scanner in = new Scanner(System.in);`
- D** `String reader = new Scanner(System.in);`



3

PART 3

Signatures and overloading

Reading a constructor header

```
Ticket(String movie, int seat, double price)
```

Piece	Meaning
Ticket	Same name as the class, so it is a constructor
String movie	1st value: a String, called movie inside
int seat	2nd value: an int, called seat inside
double price	3rd value: a double, called price inside

SIGNATURE

Ticket(String, int, double)

Signature = name + **ordered list of parameter types**. Parameter names are not part of it. A constructor header has no return type.

Which Lamp constructor runs?

THREE CONSTRUCTORS

```
Lamp()  
Lamp(int brightness)  
Lamp(String color, int brightness)
```



Constructors are **overloaded** when one class has several with **different signatures**. The arguments pick which one runs.

Call	Arguments	Runs
<code>new Lamp()</code>	none	<code>Lamp()</code>
<code>new Lamp(75)</code>	one int	<code>Lamp(int)</code>
<code>new Lamp("red", 40)</code>	a String, then an int	<code>Lamp(String, int)</code>

Pick the constructor

PAUSE AND TRY

Which constructor does each call run? Write "none" if it does not compile.

PIZZA HAS ONLY THESE

`Pizza(String size)``Pizza(String size, int toppings)`**A** `new Pizza("large", 2)`**B** `new Pizza(2, "large")`**C** `new Pizza()`**D** `new Pizza("small")`

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

PART 4

Arguments, copies, and the jump

Arguments must match: number, order, type

```
Ticket(String movie, int seat, double price)
```

Call	Problem	Compiler says (gist)
<code>new Ticket("Dune", 14, 12)</code>	none: 12 becomes 12.0	compiles
<code>new Ticket(14, "Dune", 12.5)</code>	order	int cannot be converted to String
<code>new Ticket("Dune", 14)</code>	number	argument lists differ in length
<code>new Ticket("Dune", 12.5, 14)</code>	type	lossy conversion from double to int

An int may go to a double. A double may not go to an int.

The jump in, the copy, the return

```
public class SeatCopy {  
    public static void main(String[] args) {  
        int mySeat = 14;  
        System.out.println("A: main starts");  
        Ticket t = new Ticket(mySeat);  
        System.out.println("C: back in main");  
        System.out.println(mySeat);  
    }  
}  
  
class Ticket {  
    Ticket(int seat) {  
        System.out.println("B: constructor");  
        seat = seat + 100;  
        System.out.println(seat);  
    }  
}
```

OUTPUT

```
A: main starts  
B: constructor  
114  
C: back in main  
14
```

Call by value: the parameter gets a copy, so mySeat stays 14. **Flow of control:** the whole constructor runs, then main picks up right after the call.

Line up at the movies

PAUSE AND TRY

Write the exact output, one line at a time.

IN MAIN

```
System.out.println("Line up");  
Ticket a = new Ticket(5);  
System.out.println("Buy popcorn");  
Ticket b = new Ticket(6);  
System.out.println("Find seats");
```

THE CLASS

```
class Ticket {  
    Ticket(int number) {  
        System.out.print("Ticket ");  
        System.out.println(number);  
    }  
}
```

Words to know

Constructor

Code with the class's name that builds a new object.

new

The keyword before a constructor call that creates an object.

null

What a reference variable holds when there is no object.

Signature

The name plus the ordered list of parameter types.

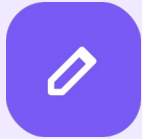
Overloaded

Several constructors, one name, different signatures.

Call by value

Each parameter starts as a copy of its argument.

Back to the warm-up line



Explain every part of this line to your partner:

```
Scanner console = new Scanner(System.in);
```

SENTENCE STARTER

The word new ____, the constructor ____, and console holds ____.

Next: the Under Construction puzzle worksheet.