

NAME _____

DATE _____

PERIOD _____

UNIT 3 · TOPIC 3.4 · WORKSHEET

Starting State

Fill in brand-new objects, rebuild a pond program, and catch the constructor that forgot its job.

About 30 minutes

10 questions

Cumulative through 3.4

TOPIC

3.4

**DIRECTIONS**

Work each puzzle in pencil. When code prints, write exactly what it prints. Every class keeps its instance variables `private`, and its short void methods, like `show()`, just print the object's state. Some puzzles include pieces you won't need.

PART A · 2 QUESTIONS

A

Warm up your brain

**THINK FIRST**

A brand-new object's constructor skips one of its `int` instance variables. Does that variable hold nothing, a random leftover number, or something definite? Guess in the margin; question 1 settles it.

1 PENCIL CHECK

A program runs `Kite k = new Kite("red", 5);`. Write the value each instance variable of `k` holds when the constructor finishes. Then answer the last row.

```
class Kite {
    private String color;
    private int length;
    private double windSpeed;
    private boolean flying;
    private String owner;
    public Kite(String kiteColor, int feet) {
        color = kiteColor;
        windSpeed = feet / 2;
    }
}
```

Instance variable	Value after the constructor
color	
length	
windSpeed	
flying	
owner	
Would new Kite() compile? (yes or no)	

2 YOU'RE THE JVM

You are the Java Virtual Machine. This code uses the `Kite` class from question 1. What prints?

```
Kite a = new Kite("red", 5);
Kite b = new Kite("red", 5);
Kite c = a;
System.out.println((a == b) + " " + (a == c));
```

- A** true true
 B false true
- C** false false
 D true false

PART B · 3 PUZZLES

B Puzzles

3 CODE TILES

The pond's main fell off the fridge. Rebuild it from four tiles so it prints the target output.

```
class Frog {
    private String name;
    private int age;
    public Frog(String frogName, int years) {
        name = frogName;
        age = years;
        System.out.println(name + " hatched, age " + age);
    }
    public Frog(String frogName) {
        name = frogName;
        age = 0;
        System.out.println(name + " is brand new");
    }
}
```

Frog pip = new Frog("Pip", 2);	System.out.println("Pond closes");
Frog lily = new Frog("Lily");	Frog pip = new Frog(2, "Pip");
System.out.println("Pond opens");	Frog lily = new Frog();

Target output

```
Pond opens
Lily is brand new
Pip hatched, age 2
Pond closes
```

YOUR MAIN METHOD



FAIR QUESTION

A constructor has no return type. So what does new hand back?

A reference to the new object. new sets aside memory for the object, runs the constructor to fill in its starting state, and then the whole new expression evaluates to the object's reference, which = stores in the variable. The constructor's only job is the starting state, so it never needs a return type.

4 SNIPPET POOL

A main method runs `Canoe trip = new Canoe("Mo", 3);` and then `trip.printReceipt();`. Fill the holes from the pool so it prints `Mo: 3 hours, $37.5` (a canoe costs \$12.50 an hour). Watch the order of the lines.

```
class Canoe {
    private String renter;
    private int hours;
    private double cost;
    public Canoe(String who, int time) {
        renter = (1);
        cost = (2);
        hours = (3);
    }
    public void printReceipt() {
        System.out.println(renter + ": " + hours + " hours, $" + cost);
    }
}
```

THE POOL

Each snippet can be used only once, and not every snippet is needed.

who	"who"	renter	time
hours	time * 12.5	hours * 12.5	

5 YOU'RE THE COMPILER

You are the compiler. Each snippet runs in a main method that can use the two classes below. Decide whether it compiles. If it does, write what it prints. If it doesn't, say why in a few words.

```
class Fan {
    private int speed;
    public Fan(int startSpeed) {
        speed = startSpeed;
    }
    public void show() {
        System.out.println("fan " + speed);
    }
}

class Lamp {
    private int watts;
    private boolean on;
    public void show() {
        System.out.println("lamp " + watts + " " + on);
    }
}
```

Snippet	Compiles?	Output, or why not
1 Fan f = new Fan();		
2 Lamp m = new Lamp(); m.show();		
3 Fan f = new Fan(3); f.show();		
4 Lamp m = new Lamp(60);		
5 Fan f = new Fan(2.5);		



WATCH OUT

`public void Robot(String model)` is **not** a constructor, even inside a class named `Robot`. The word `void` is a return type, and a constructor has none, so Java treats this as an ordinary method that happens to share the class's name. `new Robot("R2")` then fails to compile because no constructor takes a `String`.

PART C · 5 QUESTIONS

C

Catch the mistake

6

ONE-TOKEN FIX

A main method runs `Toad pip = new Toad("Pip", 4);` and then `pip.report();`. It prints `Pip caught 0 flies`. It should print `Pip caught 4 flies`. **Delete** exactly one token to fix it, and explain why it printed 0.

```

1  class Toad {
2      private String name;
3      private int flies;
4
5      public Toad(String who, int caught) {
6          name = who;
7          int flies = caught;
8      }
9      public void report() {
10         System.out.println(name + " caught " + flies + " flies");
11     }
12 }
```

THE TOKEN, AND WHY IT PRINTED 0

7

FACE-OFF

Tonight: an Instance Variable and a Parameter argue about who matters more. Fill in the blanks.

**INSTANCE
VARIABLE**

I'm declared in the class, and I last as long as my _____ does.

PARAMETER

I start with a _____ of the argument, and I vanish when the constructor _____.

**INSTANCE
VARIABLE**

So the constructor must assign you to me, or I keep my _____ value.

PARAMETER

Take my name and watch: in `size = size;` both sides mean the _____.

**INSTANCE
VARIABLE**

Rude. Until Topic 3.9, use a _____ name, like `startSize`.

8 TWO TRUTHS AND A LIE

A Scanner is **mutable**: each read moves it forward, so its state changes. Two of the claims about this program are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
import java.util.Scanner;

public class Raffle {
    public static void main(String[] args) {
        Scanner tickets = new Scanner("12 40 7 33");
        Drawing first = new Drawing(tickets);
        System.out.println("main reads " + tickets.nextInt());
    }
}

class Drawing {
    private Scanner box;
    private int winner;

    public Drawing(Scanner ticketBox) {
        box = ticketBox;
        winner = box.nextInt();
        System.out.println("winner " + winner);
    }
}
```

- **1** After the constructor runs, `box` and `tickets` refer to the same Scanner object.
- **2** The program's last line of output is `main reads 40`.
- **3** `tickets` was passed by value, so the `Drawing` got its own copy of the Scanner, and its read can't change what main reads next.

THE LIE, REWRITTEN

9 COPY OR SHARE?

A `Playlist` is mutable: songs can be added after it's built. It has a constructor `Playlist(Playlist original)` that builds a **new** `Playlist` with the same songs. A `Party` class has `private Playlist music;`. Which line belongs in the constructor `public Party(Playlist startMusic)?`

- A** `music = startMusic;`
- B** `startMusic = music;`
- C** `Playlist music = new Playlist(startMusic);`
- D** `music = new Playlist(startMusic);`

10 WHO AM I?

Some ideas from today are playing a party game. Each gives a clue about itself. Name the idea.

Clue	I am...
I have exactly the class's name and no return type, not even <code>void</code> .	
Nobody writes me. Java supplies me, with no parameters, only when a class has no constructor at all.	
I'm what a <code>String</code> instance variable holds before anyone assigns it.	
I'm the set of values an object's instance variables hold right now.	
"A frog has a name" describes me.	
My state can change after I'm built, so a constructor should store a copy of me.	