

NAME _____

DATE _____

PERIOD _____

UNIT 3 · TOPIC 3.3 · WORKSHEET

Inside the Class

Label the parts, trace two objects, play compiler at the private door, and build a class that works.

About 35 minutes

9 questions

Cumulative through 3.3

TOPIC

3.3

**DIRECTIONS**

Work each puzzle in pencil. For Java code, write exactly what prints, or say why it doesn't compile. In a one-file program, only the class with `main` is marked `public`; read past `static` in `main`'s header.

**THINK FIRST**

A class can mark some of its parts *private*. Why would the person who wrote a class want to stop other code from touching its data? Guess in the margin; question 6 puts it to the test.

PART A · 3 QUESTIONS

A Take it apart

1 PENCIL CHECK

Here is a complete class in its own file, `Aquarium.java`. Name the part of the class on each set of lines.

`Aquarium.java`

```

1  public class Aquarium {
2      private int fishCount;
3      private boolean lightOn;
4      public Aquarium() {
5          fishCount = 3;
6          lightOn = false;
7      }
8      public void addFish() {
9          fishCount++;
10     }
11     public void printTank() {
12         System.out.println(fishCount + " fish");
13     }
14 }
```

Lines	Part of the class
1	
2-3	
4-7	
8-10 and 11-13	

2 YOU'RE THE JVM

This main is in another class, TankDemo, and uses the Aquarium class from question 1. What prints?

```
Aquarium a = new Aquarium();
Aquarium b = new Aquarium();
a.addFish();
a.addFish();
b.addFish();
a.printTank();
b.printTank();
```

- | | |
|------------------------------|------------------------------|
| A 5 fish, then 4 fish | B 6 fish, then 6 fish |
| C 2 fish, then 1 fish | D 3 fish, then 3 fish |

3 WRITE IT

A specification says: "A Plant class has a String name and an int height, and a boolean called watered that is a public instance variable." Write the class header and the three instance variable declarations, following good encapsulation. (No constructor or methods yet.)

YOUR CLASS

PART B · 2 PUZZLES

B Who gets in?

4 YOU'RE THE COMPILER

This class is in the same file as `main`, but `main` belongs to a different class, `TicketBooth`. Each snippet in `main` comes right after `Ticket t = new Ticket();`. Decide whether it compiles. If it does, write the output. If not, say why.

```
class Ticket {
    public String event;
    private int seat;
    public Ticket() {
        event = "Play";
        seat = 12;
    }
    public void printTicket() {
        System.out.println(event + " seat " + seat);
    }
}
```

Snippet in main	Compiles?	Output, or why not
1 <code>t.printTicket();</code>		
2 <code>t.seat = 5;</code>		
3 <code>t.event = "Concert";</code> <code>t.printTicket();</code>		
4 <code>System.out.println(t.seat);</code>		



FAIR QUESTION

They're in the same file. Why can't main see seat?

Because `private` is about the *class*, not the file. Only code between `Ticket`'s own braces is inside. A `main` in another class is outside, even one line away.

5 SNIPPET POOL

Fill each hole from the pool so the program prints 70 degrees. Each snippet is used at most once, and some aren't needed.

TeaTime.java

```
public class TeaTime {
    public static void main(String[] args) {
        Kettle k = new Kettle();
        k.heat();
        k.heat();
        k.printTemp();
    }
}

class Kettle {
    private int temp;
    (1) {
        temp = 20;
    }
    public void heat() {
        temp = temp + (2);
    }
    public void printTemp() {
        System.out.println((3) + " degrees");
    }
}
```

THE POOL

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

public Kettle()	public void Kettle()	private Kettle()
public kettle()	25	50
10	temp	

PART C · 4 QUESTIONS

C

Keep it private

6

TWO TRUTHS AND A LIE

Here is a GiftCard class. Two claims about it are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
class GiftCard {  
    private int balance;  
    public GiftCard() {  
        balance = 25;  
    }  
    public void spendFive() {  
        if (balance >= 5) {  
            balance = balance - 5;  
        }  
    }  
    public void printBalance() {  
        System.out.println("$" + balance);  
    }  
}
```

- **1** Code in another class can't make balance negative: the only way to change it is spendFive, which checks first.
- **2** Because balance is private, its value can never change after the constructor runs.
- **3** If a specification said balance is a public instance variable, you would declare it public.

THE LIE, REWRITTEN

7 ONE-TOKEN FIX

main should ring the bell by using only the parts of Doorbell that are meant for outside code. It doesn't compile. Change exactly **one token in `main`** (lines 1-6) so it prints Ding dong!.

```

1 public class FrontDoor {
2     public static void main(String[] args) {
3         Doorbell bell = new Doorbell();
4         bell.ring();
5     }
6 }
7
8 class Doorbell {
9     public Doorbell() {
10    }
11    public void press() {
12        ring();
13    }
14    private void ring() {
15        System.out.println("Ding dong!");
16    }
17 }
```

YOUR FIX, AND THE COMPILER'S COMPLAINT

8 FACE-OFF

Tonight: Public and Private argue about who's more useful. Fill in the blanks.

PUBLIC	Anyone can use me, from inside my class or from _____ class. That's why constructors always wear me.
PRIVATE	I'm for code _____ my own class only. Another class's main can't even read me.
PUBLIC	So what are you good for?
PRIVATE	Protecting data. Mark the instance variables with me and no outside code can put an object into a _____ state. That's called _____.
PUBLIC	Fine, but I'm still the one outside code calls. Your _____ methods only run because one of my methods calls them.

9 WHO AM I?

Some parts of a class are playing a party game. Each gives a clue about itself. Name the part or idea.

Clue	I am...
I share the class's name, I have no return type, and new runs me.	
I'm declared in the class but outside every method, and each object gets its own copy of me.	
I'm the first line: the keyword <code>class</code> and the class's name.	
I'm a keyword that lets only code inside the class use what I'm attached to.	
I keep a class's data hidden inside it, so other classes use only its public parts.	