

NAME

DATE

PERIOD

UNIT 3 · TOPIC 3.9 · WORKSHEET

Who Is this?

Find the current object, rescue the hidden instance variable, and pass yourself along.

About 30 minutes

8 questions

Cumulative through 3.9

TOPIC

3.9



DIRECTIONS

Work in pencil. Lines marked *in main*, *in another class* run in a separate class's main, in order. Write output exactly as Java prints it. Some puzzles include pieces you won't need.



THINK FIRST

Two kites run the same `rise` method. When the code says `this.height`, how does Java know which kite's height you mean? Guess; question 2 settles it.

PART A · 2 QUESTIONS

A

Warm up your brain

1 FACE-OFF

Tonight, inside `public boolean isTwin(Kite other):` `this` and the parameter `other` argue. Fill in the blanks.

THIS I'm the object the method was called on: whatever is before the ____.

OTHER I'm a _____. I get my reference from the argument in the parentheses.

THIS Nobody declares me, and nobody assigns me. Java fills me in on every ____.

OTHER Yet in a _____ method you don't even exist.

THIS True. And `this == other` is true only when we're the _____ object.

2 YOU'RE THE JVM

You are the JVM. What prints?

```
class Kite {
    private int height;
    public Kite(int height) {
        this.height = height;
    }
    public void rise(int height) {
        this.height += height;
    }
    public int getHeight() { return height; }
}
// in main, in another class
Kite a = new Kite(10);
Kite b = new Kite(20);
b.rise(5);
a.rise(b.getHeight());
System.out.println(a.getHeight() + " " + b.getHeight());
```

A 35 25**B** 10 50**C** 15 25**D** 35 35**FAIR QUESTION**

If the constructor needs `this.height`, why doesn't `getHeight()`?

Inside `getHeight()` nothing else is named `height`, so the plain name already means the current object's height. You need `this.` only when a parameter or local variable hides the instance variable.

PART B · 2 PUZZLES

B Puzzles

3 MATCH-UP

Four versions of the Song constructor differ only in where `this.` appears. Match each version to what main prints. Each output is used once.

```
class Song {
    private String title;
    private int plays;
    public Song(String title, int plays) {
        // version 1, 2, 3, or 4 goes here
    }
    public String info() { return title + " " + plays; }
}
// in main, in another class
Song s = new Song("Sunrise", 12);
System.out.println(s.info());
```

Version	Constructor body	Output
1	<code>this.title = title;</code> <code>this.plays = plays;</code>	
2	<code>title = title;</code> <code>this.plays = plays;</code>	
3	<code>this.title = title;</code> <code>plays = plays;</code>	
4	<code>title = this.title;</code> <code>plays = this.plays;</code>	

P null 0 **Q** Sunrise 0 **R** null 12 **S** Sunrise 12

4 SNIPPET POOL

Fill each hole so the program prints Owls lost to Hawks and then Hawks lost to Owls. Each snippet is used at most once, and some aren't needed.

```
class Team {
    private String name;
    public Team(String name) {
        (1) = name;
    }
    public void beat(Team loser) {
        loser.lostTo((2));
    }
    public void lostTo(Team winner) {
        System.out.println(name + " lost to " + (3));
    }
}

// in main, in another class
Team a = new Team("Hawks");
Team b = new Team("Owls");
a.beat(b);
b.beat(a);
```

THE POOL

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

name	loser	this.name
winner	this	winner.name

PART C · 4 QUESTIONS

C Catch the mistake

5 YOU'RE THE COMPILER

You are the compiler. Each row is one method added to `Clock`. Does it compile? If not, say why.

```
class Clock {
    private int hour;
    public Clock(int hour) {
        this.hour = hour;
    }
    public int getHour() { return hour; }
    // one method from the table goes here
}
```

Method added to <code>Clock</code>	Compiles?	If not, why not
1 <code>public static int peek() { return this.hour; }</code>		
2 <code>public boolean same(Clock other) { return this == other; }</code>		
3 <code>public static boolean late() { return getHour() > 12; }</code>		
4 <code>public static int gap(Clock a, Clock b) { return a.hour - b.hour; }</code>		
5 <code>public void setHour(int hour) { this.hour = hour; }</code>		
6 <code>public static void tick() { hour++; }</code>		

6 TWO TRUTHS AND A LIE

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

```
class Coach {
    public void cheer(Player p) {
        p.addPoints(10);
    }
}
class Player {
    private int points = 0;
    public void askCoach(Coach c) {
        c.cheer(this);
    }
    public void addPoints(int n) { points += n; }
    public int getPoints() { return points; }
}
// in main, in another class
Player ana = new Player();
Coach c = new Coach();
ana.askCoach(c);
ana.askCoach(c);
System.out.println(ana.getPoints());
```

- **1** It prints 20.
- **2** While cheer runs, p and ana refer to the same object.
- **3** c.cheer(this) hands the coach a copy of the player, so ana keeps 0 points.

THE LIE, REWRITTEN

7 ADD-THIS FIX

setVolume should ignore any volume over 10, so this should print 7. It prints 3. Add `this.` in exactly one place to fix it.

```

1 class Speaker {
2     private int volume;
3     public Speaker(int volume) {
4         this.volume = volume;
5     }
6     public void setVolume(int volume) {
7         if (volume <= 10) {
8             volume = volume;
9         }
10    }
11    public int getVolume() { return volume; }
12 }
13 // in main, in another class
14 Speaker s = new Speaker(3);
15 s.setVolume(7);
16 s.setVolume(15);
17 System.out.println(s.getVolume());

```

LINE, WHERE THIS. GOES, AND WHY

8 WHO AM I?

Each idea gives a clue about itself. Name the idea.

Clue	I am...
I hold a reference to the object whose method or constructor is running.	
I'm the object before the dot in a call, the one in charge while it runs.	
I run with no current object, so <code>this</code> doesn't exist in me.	
A parameter with my name hides me; write <code>this.</code> to reach me.	
Pass <code>this</code> to a method and I receive a copy of the reference.	