Name: _____ Session: AM / PM
Circle one

Chapter 3 - Day 2

Combining and Comparing Conditions

WHAT THE BRACES MEAN

Braces decide what is controlled: Without braces, an if controls _____. Indentation does not change that; Java ignores it. An else pairs with the nearest if above it that _____, unless braces say otherwise.

E2-A Reading code without braces

MissionBraces.java

What changed: The same readiness checks from Day 1, written without braces. The indentation suggests one thing; Java does another.

```
1  public class MissionBraces {
2      public static void main(String[] args) {
3          double fuel = 20;
4          boolean batteryReady = false;
5
6          // Missing braces: which lines does this if control?
7          if (fuel >= 25)
8              System.out.println("Fuel OK.");
9              System.out.println("Cleared for launch.");
10
11         // No braces: which if does this else belong to?
12         if (batteryReady)
13             if (fuel >= 25)
14                 System.out.println("Mission ready.");
15         else
16             System.out.println("Battery not ready.");
17     }
18 }
```

1. Lines the if on line 7 controls:

2. Printed by lines 7-9 when fuel is 20:

3. The else on line 15 pairs with the if on line

4. Printed by lines 12-16 when batteryReady is false:

Fix it: The stray semicolon

Broken on purpose

```
7         if (fuel >= 25);
8             System.out.println("Fuel OK.");
```

Cause:

Repair:

When does it fail?

LOGICAL OPERATORS

Term	Definition / memory cue
logical operator	
&& (AND)	
|| (OR)	
! (NOT)	
range check	
short-circuit evaluation	

Complete the truth table.

A	B	A && B	A || B	!A
true	true			
true	false			
false	true			
false	false			

E2-B Inside and outside a range

TelemetryCheck.java

What changed: Validates a telemetry reading against the classroom rule 0 to 100. The same range is checked three ways.

```
        // Inside the range: BOTH sides must be true
        boolean inRange = (reading >= 0 && reading <= 100);
        if (inRange) {
            System.out.println("Reading accepted.");
        }

        // Outside the range: EITHER side is enough
        if (reading < 0 || reading > 100) {
            System.out.println("Sensor fault.");
        }

        // NOT flips true and false
        if (!inRange) {
            System.out.println("Request a new reading.");
        }
```

1. An inside-the-range check uses

2. An outside-the-range check uses

3. `!inRange` is true when the reading is

4. Why can `reading < 0 && reading > 100` never be true?

Predict every line printed after the prompt.

Reading	Output	Ran it?
`-5`		☐
`0`		☐
`100`		☐
`100.5`		☐

Write the condition for each requirement.

Requirement	Condition
crew count from 2 to 6, inclusive	
altitude below 400 or above 420	
pressure from 95 to 105, inclusive	
hatch is NOT sealed (boolean hatchSealed)	

PRECEDENCE AND SHORT-CIRCUIT

Order of evaluation: `&&` is evaluated before `||`, the same way `*` comes before `+`. Use parentheses to make the grouping you want obvious. With `&&`, a _____ left side skips the right side. With `||`, a _____ left side skips the right side.

E2-C Grouping with and without parentheses

`LaunchDecision.java`

What changed: Two launch checks that use the same three flags. Only the parentheses are different.

```
3         boolean batteryReady = false;
4         boolean fuelReady = true;
5         boolean override = true;
6
7         // && is done before ||
8         if (batteryReady && fuelReady || override) {
9             System.out.println("Check 1: launch approved.");
10        }
11
12        // Parentheses change the grouping
13        if (batteryReady && (fuelReady || override)) {
14            System.out.println("Check 2: launch approved.");
15        }
```

1. Does Check 1 print?

2. Does Check 2 print?

3. Line 8 is read as `(batteryReady && fuelReady) || override`, so the result is

E2-D A guard that prevents a crash `SupplyShare.java`

What changed: Checks supplies per crew member, but only after making sure the crew count is not 0.

```
11          // The left side guards the division on the right
12          if (crew != 0 && supplies / crew >= 10) {
13              System.out.println("Enough supplies per crew member.");
14          } else {
15              System.out.println("Check the crew count or supplies.");
16          }
```

1. Crew is 0. Does Java evaluate `supplies / crew`?

2. Printed for 120 supplies and crew 0:

3. If the two sides were swapped, crew 0 would cause

Is the right side evaluated? Assume int crew and int supplies.

Condition	crew	Left side	Right side evaluated?
`crew != 0 && supplies / crew >= 10`	0		
`crew != 0 && supplies / crew >= 10`	6		
`crew == 0 || supplies / crew < 10`	0		
`crew == 0 || supplies / crew < 10`	6		

COMPARING STRINGS

Why == is the wrong tool: A String variable holds a _____ to an object, not the text itself. == compares references, so two Strings with the same text can still fail. Use a String method to compare the _____.

String Comparison Methods

Method	Example	What it returns
`equals`	`command.equals("LAUNCH")`	
`equalsIgnoreCase`	`destination.equalsIgnoreCase("mars")`	
`compareTo`	`a.compareTo(b)`	
`compareToIgnoreCase`	`a.compareToIgnoreCase(b)`	

E2-E Comparing mission text

MissionCommand.java

What changed: Reads a command and a destination typed by the user, then compares them three different ways.

```
11          if (command == "LAUNCH") {
12              System.out.println("== found a match.");
13          }
14          if (command.equals("LAUNCH")) {
15              System.out.println("Launch command received.");
16          }
17          if (destination.equalsIgnoreCase("mars")) {
18              System.out.println("Course set for Mars.");
19          }
20
21          String callSign1 = "Orion";
22          String callSign2 = "orion";
23          System.out.println(callSign1.compareTo(callSign2));
24          System.out.println(callSign1.compareToIgnoreCase(callSign2));
```

1. User types LAUNCH. Does line 12 run?

2. Method that matches mars, MARS, and Mars:

3. Line 23 prints a _____ number

4. Why?

Predict every line printed after the prompts.

Command	Destination	Output	Ran it?
`LAUNCH`	`MARS`		☐
`launch`	`Venus`		☐

Negative, zero, or positive?

Expression	Result	Meaning
`"Apollo".compareTo("Gemini")`		
`"Mars".compareTo("Mars")`		
`"Venus".compareTo("Mars")`		

SCOPE

Where a variable exists: A local variable exists from its declaration to the _____ of the block it was declared in. Before Java lets you use it, it must also have a value on _____ that reaches that line.

E2-F A value from a decision

MissionStatus.java

What changed: Chooses GO or NO-GO and prints it after the if-else is finished.

```
 9         // Declared before the if, so it still exists after the if
10         String status;
11         if (fuel >= 25) {
12             status = "GO";
13         } else {
14             status = "NO-GO";
15         }
16         System.out.println("Status: " + status);
```

1. Line where status is declared: line

2. Why is status declared before the if?

3. Why does line 16 compile?

Fix it: Declared inside the block

Broken on purpose

```
11         if (fuel >= 25) {
12             String status = "GO";
13         } else {
14             String status = "NO-GO";
15         }
16         System.out.println("Status: " + status);
```

Cause:

Repair:

When does it fail?

Fix it: Missing a path

Broken on purpose

```
10          String status;
11          if (fuel >= 25) {
12              status = "GO";
13          }
14          System.out.println("Status: " + status);
```

Cause:

Repair:

When does it fail?

THE CONDITIONAL OPERATOR

condition ? valueIfTrue : valueIfFalse: The conditional (ternary) operator is an _____: it produces a value that must be stored, printed, or otherwise used. It fits a decision with exactly _____ outcomes.

E2-G Two-outcome decisions in one line

`StatusTernary.java`

What changed: The GO / NO-GO choice from MissionStatus, rewritten as one line, plus an even/odd bay choice.

```
11          // condition ? valueIfTrue : valueIfFalse
12          String status = (fuel >= 25) ? "GO" : "NO-GO";
13          String bay = (containerId % 2 == 0) ? "port" : "starboard";
14
15          System.out.println("Status: " + status);
16          System.out.println("Container " + containerId + ": " + bay);
```

1. Value stored in status when fuel is 30:

2. Container ID 7 goes to the _____ bay

3. `containerId % 2 == 0` is true when the ID is

Rewrite each if-else as one assignment using the conditional operator.

if-else	Conditional operator
`if (crew >= 3) team = "full"; else team = "short";`	
`if (oxygen < 20) alarm = true; else alarm = false;`	