

JAVA / LESSON 07 OF 40

Night Falls: if or else?

Let the program choose to act according to conditions

Java 11+ | 90-110 minutes



Making a Nighttime Survival Advisor

From a conditional door to a complete decision-making chain: distinguish multiple reminders and one main suggestion, check the rules with boundary values, and finally complete a night survival consultant that can understand the answers.

Before you start: It is recommended to complete Lessons 1-6 first

Read the full lesson and copy the code:

<https://teacher-ma.de/minecraft-java-lesson.php?lesson=7&lang=en>

Long code lines may wrap at the > margin marker. The marker is not part of the code.

In this lesson

- 01. A door without walls
- 02. if: Act only when the condition is met
- 03. else: Otherwise go another way
- 04. else if: The order is also a rule
- 05. Turn small conditions into big problems
- 06. Multiple reminders, or one main suggestion?
- 07. Bug loves to sit on the threshold
- 08. Nesting: Find a room after entering the village
- 09. It can run, but it may be wrong.
- 10. Guess first, then run, and finally explain
- 11. Project Blueprint: Night Survival Consultant
- 12. Consultant Code: Reading and Verification
- 13. Consultant Code: Selection and Finishing
- 14. Acceptance: Only change one input at a time
- 15. Put judgment skills into your backpack

01 / 15

A door without walls



As soon as the sun hid behind the mountain, Steve stuck the door in the grass: "Safe house, completed!" Alex pointed to both sides of the door: "Where are the walls?" A zombie walked around the door, as if praising the ventilation design of the house.

Procedures also require clear thresholds. What do you do when it's dark, what do you do when your life is low, and what do you do first when both things happen at the same time? In this lesson, we write the vague "it depends" into a rule that can be checked.

- When the condition of if is true, which statements will be executed?
- Is the writing method of reminding all missing equipment the same as selecting only one priority action?
- When life happens to be 0, 8, or 9, why are the results different?

Life 0 to 20, supply threshold and suggested order are all practice rules for this lesson. The output "Return to the house" is just a line of console text and does not move the game character.

02 / 15

if: Act only when the condition is met

if can be read as "if". The boolean value must be obtained within the parentheses, and the curly braces enclose what is to be done when the condition is met. The program is only judged once when it is executed here; if the variables change later, this code will not be automatically repeated.

Complete program: Save as HealthAlert.java

```
public class HealthAlert {  
    public static void main(String[] args) {  
        int health = 7;  
        if (health <= 8) {  
            System.out.println("Supply first");  
        }  
        System.out.println("End of inspection");  
    }  
}
```

Output

```
Supply first  
End of inspection
```

Conditional door: two roads finally meet

health <= 8 ?

Condition	Path
true	Output "supply first"
false	Skip supply statement

↓ Both paths output "Check End"

Change 7 to 12, leaving only "Check End". Subsequent code marked as "fragment" should be put into main separately each time. Do not stick examples of variables with the same name into one large section.

else: Otherwise go another way



If you want to do something even when the condition is not true, add else. The same if / else judgment only goes one way, like a mine cart crossing a fork in the road, and it will not copy another one of itself.

Snippet: place inside main and run separately

```
boolean isNight = true;
if (isNight) {
    System.out.println("Return to the house to rest");
} else {
    System.out.println("Collection during the day");
}
```

Output

```
Return to the house to rest
```

Snippet: place inside main and run separately

```
int diamonds = 2;
if (diamonds >= 3) {
    System.out.println("The number of diamonds is enough");
} else {
    System.out.println("Not much." + (3 - diamonds) + "diamonds");
}
```

Output

```
Not much.1diamonds
```

No more conditions will be written after else. Changing diamonds to 3 and 4 will report that the quantity is sufficient; only the number of diamonds is checked here, not the sticks or workbench.

04 / 15

else if: The order is also a rule

A branch chain is checked from top to bottom. If the first branch whose condition is true is selected, the subsequent branches will no longer be checked. Deal with more urgent and specific situations first, and don't let one broad condition swallow up the narrow conditions that follow.

Snippet: place inside main and run separately

```
int health = 7;
if (health <= 0) {
    System.out.println("Reborn.");
} else if (health <= 8) {
    System.out.println("Eat something quickly");
} else {
    System.out.println("Continue the adventure");
}
```

Output

```
Eat something quickly
```

Select only one: Find the entrance from top to bottom

Check in order	On a match
health <= 0 ?	Reborn.
health <= 8 ?	Eat something quickly
else	Continue the adventure

Check from top to bottom; execute only the first matching branch.

If you write `health <= 8` first, and then `health <= 0`, `health 0` will also be blocked by the first item. The program compiles, but gives incorrect suggestions. We will study the sequence here for the time being and will add the legal range check later.

05 / 15

Turn small conditions into big problems

First explain the conditions clearly in one sentence

Element	Explanation
&& and	Both conditions are met: it is dark and there is no shelter.
or	At least one item is satisfied: having a key or being the owner.
! Negation	Turn true and false over: !hasShelter means no shelter.

Snippet: place inside main and run separately

```
boolean isNight = true, hasShelter = false;
if (isNight && !hasShelter) {
    System.out.println("Build temporary shelter first");
}
```

Output

```
Build temporary shelter first
```

Snippet: place inside main and run separately

```
boolean hasKey = false, isOwner = true;
if (hasKey || isOwner) {
    System.out.println("The door is open");
}
```

Output

```
The door is open
```

Snippet: place inside main and run separately

```
String weather = " RAIN ".strip();
if ("rain".equalsIgnoreCase(weather)) {
    System.out.println("Move the construction plan indoors");
}
```

Output

Move the construction plan indoors

When the left side of `&&` is false and `||` is true, the calculation on the right side will be skipped. Use parentheses to write clear combination rules, such as `thunder || (isNight && !hasShelter)`. To compare literal content use `equals` or `equalsIgnoreCase`.

06 / 15

Multiple reminders, or one main suggestion?

Lack of bread and shield can remind you of two things; deciding "what to do now" may only require one main suggestion. Determine the task first, then choose an independent if or branch chain.

Snippet: place inside main and run separately

```
int health = 6;
if (health <= 8) {
    System.out.println("Bring bread");
}
if (health <= 16) {
    System.out.println("With shield");
}
```

Output

Bring bread
With shield

Snippet: place inside main and run separately

```
int health = 6;
if (health <= 8) {
    System.out.println("Bring bread");
} else if (health <= 16) {
    System.out.println("With shield");
}
```

Output

Bring bread

Same conditions, different organizational methods

Element	Explanation
Two independent if	Each of the two conditions is checked once, and two reminders may be output.
An if / else if chain	After selecting the previous item, the following conditions will no longer be checked.

Change the life to 20, and there will be no output in both sections. A branch chain without a trailing else allows a branch to not be executed at all.

07 / 15

Bug loves to sit on the threshold

Just because 7 can run correctly does not mean that 8 is also correct. First define the legal range of life, and then divide it into several levels without omissions and repeated hits. Each threshold tests itself and its neighbors on both sides.

Snippet: place inside main and run separately

```
int health = 8;
if (health < 0 || health > 20) {
    System.out.println("Life range error");
} else if (health == 0) {
    System.out.println("Reborn.");
} else if (health <= 8) {
    System.out.println("Low life");
} else if (health <= 16) {
    System.out.println("Proceed with caution");
} else {
    System.out.println("Sufficient life");
}
```

Output

Low life

Life classification: write the test points next to the rules

Input range	The results of this lesson	represents the test value
Out of range	Life range error	-1, 21
0	Reborn.	0
1 to 8	Low life	1, 8
9 to 16	Proceed with caution	9, 16
17 to 20	Sufficient life	17, 20

When health ≤ 8 is reached, out-of-range and 0 have been excluded; therefore this branch actually represents 1 to 8. The final else only leaves 17 to 20, which does not represent all integers.

08 / 15

Nesting: Find a room after entering the village

First ask if it is night, and then ask if there is a residence in the night branch. This is nesting. Indentation helps us see the hierarchy, and curly braces really determine which statements belong to which layer.

Snippet: place inside main and run separately

```
boolean isNight = true, hasShelter = false;
if (isNight) {
    if (hasShelter) {
        System.out.println("Return to the house to rest");
    } else {
        System.out.println("Build temporary shelter first");
    }
} else {
    System.out.println("Collection during the day");
}
```

Output

```
Build temporary shelter first
```

Snippet: place inside main and run separately

```
boolean isNight = false;
String advice;
if (isNight) {
    advice = "Go home";
} else {
    advice = "Collection";
}
System.out.println(advice);
```

Output

```
Collection
```

Advice is declared outside first, and it can be read outside the block by assigning values to it in both ways. Local variables declared in a certain branch cannot be used outside that code block.

It can run, but it may be wrong.



Logical counterexample: predict first, then find redundant symbols

```
boolean isNight = false;
if (isNight); { // Error demonstration: extra semicolons
    System.out.println("Close the door");
}
```

Output

```
Close the door
```

The semicolon is an empty statement. Here if only controls this empty statement, and the subsequent code blocks are executed independently; deleting the semicolon after the condition can be fixed.

Logical counterexample: the key was secretly changed by the program

```
boolean hasKey = false;
if (hasKey = true) { // Error demonstration: assignment, not comparison
    System.out.println("Open the door");
}
```

Output

```
Open the door
```

boolean assignment also produces a boolean result, so this error actually compiles. It is enough to change it to `if (hasKey);` if `(health = 8)` is written, the result is int, which cannot be used as a condition.

- Check the curly braces to avoid mismatching else if.

- Check = and ==, do not treat assignments as comparisons.
- Check < and <= and test with a value exactly equal to the threshold.

10 / 15

Guess first, then run, and finally explain

Let each run leave you with thoughts

Step	Explanation
1. Guess first	Write down the output and hit branches.
2. Run again	Run only the current fragment, checking each line.
3. Explanation	Indicate which judgment changed the route.

Snippet: place inside main and run separately

```
int diamonds = 3, sticks = 1;
if (diamonds >= 3 && sticks >= 2) {
    System.out.println("The materials are ready");
} else {
    System.out.println("The materials are not yet complete");
}
```

Snippet: place inside main and run separately

```
int health = 8;
if (health <= 8) {
    health += 2;
}
if (health > 8) {
    health += 1;
}
System.out.println(health);
```

Check answers

The first question outputs "the materials are not complete" because there is only one wooden stick. The second question outputs 11: the first time it turns 8 into 10, the second independent if reads the new value 10 and adds 1.

Only change the second if of the second question to else if, and the result becomes 10. Once the previous branch has been selected, subsequent conditions will no longer be checked.

11 / 15

Project Blueprint: Night Survival Consultant

"Don't recommend decoration when my life is 0." Steve's needs are very reasonable. The advisor reads three lines of input each time and gives only one main suggestion; if you want to ask again, rerun the program.

Enter a priori and then make a decision

Step	Explanation
1. Read entire line	Life, time period, residence, strip first.
2. Check options	Time period day/night, residence yes/no.
3. Parse numbers	parseInt and catch, then check the range.

Main suggestion priority: stop checking downwards if selected previously

Check in order	On a match
Legal life == 0	Prepare for rebirth
Legal life <= 8	Go to a safe place for supplies first
Life 9 to 20	Let's look at time period and residence again

Check from top to bottom; execute only the first matching branch.

There are unknown characters in the time period and residence, please report the option error first. If the life cannot be parsed, an integer error will be reported, and if it is out of bounds, a range error will be reported. nite cannot be directly used as daytime: false will only have a clear meaning after the second choice verification is passed.

12 / 15

Consultant Code: Reading and Verification

Create new NightAdvisor.java. The complete program is printed on this page and the next page: keep them sequentially in the same file, without adding extra braces between the two pages.

NightAdvisor.java · First half, continue on next page

```
import java.util.Scanner;

public class NightAdvisor {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Life 0-20: ");
        String healthText = in.nextLine().strip();
        System.out.println("Period day/night: ");
        String time = in.nextLine().strip();
        System.out.println("Have a place to live yes/no: ");
        String home = in.nextLine().strip();
        boolean timeOK = "day".equalsIgnoreCase(time)
            || "night".equalsIgnoreCase(time);
        boolean homeOK = "yes".equalsIgnoreCase(home)
            || "no".equalsIgnoreCase(home);
        if (!timeOK || !homeOK) {
            System.out.println("The option is wrong, please check the time period and residence");
        } else {
            try {
                int health = Integer.parseInt(healthText);
                boolean isNight = "night".equalsIgnoreCase(time);
                boolean hasShelter = "yes".equalsIgnoreCase(home);
```

All three inputs are read with `nextLine` to avoid leaving the end of the line when mixing `nextInt`. `timeOK` and `homeOK` first confirm that the words are legal; then convert `night` and `yes` to `true` after passing.

Try pushing by hand: `8/night/yes` will supply supplies first; `9/night/yes` will return to the house. With only 1 life left, priorities dictate different routes.

Consultant Code: Selection and Finishing

NightAdvisor.java · Next to the previous page

```
        if (health < 0 || health > 20) {
            System.out.println("Life must be 0 to 20 between");
        } else if (health == 0) {
            System.out.println("Life is exhausted, prepare for rebirth");
        } else if (health <= 8) {
            System.out.println("Go to a safe place for supplies first");
        } else if (isNight) {
            if (hasShelter) {
                System.out.println("Return to the house to rest");
            } else {
                System.out.println("Build temporary shelter first");
            }
        } else {
            System.out.println("Collect during the day, don't forget to bring supplies");
        }
    } catch (NumberFormatException e) {
        System.out.println("Please enter life 0 to 20 integer");
    }
}
}
```

try/catch follows the parsing method of Lesson 6. If you enter abc, a decimal, or a large number that cannot fit into an int, catch will give an integer prompt. After the input is legal, this branch chain will only output one main suggestion.

After running, enter 8, night, yes in sequence, and press Enter on each line. After three prompts, "Go to a safe place to resupply first" should be output. The output is not the game action, but the consultant's judgment on the practice data.

14 / 15

Acceptance: Only change one input at a time

Only move one variable and observe the route changes

Element	Explanation
8 / night / yes	Go to a safe place for supplies first
9 / night / yes	Return to the house to rest
9 / night / no	Build temporary shelter first

Consultant test form

Life / Period / Residence	Expected main results
0 / day / yes	Life is exhausted, prepare for rebirth
20 / day / no	Collect during the day, don't forget to bring supplies
-1 or 21/day/yes	Lives must be between 0 and 20
abc, 8.5 or too large integer / day / yes	Please enter an integer from 0 to 20 for life
9 / nite / yes	The option is wrong, please check the time period and residence
9 / night / maybe	The option is wrong, please check the time period and residence

Let testing be evidence

- Test 1, 8, 9, and 20 again to confirm that each main suggestion is accessed.
- Try blanks at both ends and uppercase NIGHT / YES to confirm that cleanup and ignore case take effect.
- When life and options are wrong at the same time, the option error is reported first according to the agreement.
- Each row records the input, prediction, actual result, and reason for the variance.

15 / 15

Put judgment skills into your backpack



Wooden Pickaxe Task: Diamond Quantity Check

Use variables first, then connect to keyboard input. If a negative number reports an error, at least 3 will be reported as enough, otherwise the difference will be calculated. Test - 1, 0, 2, 3, 4; 2 should be 1 less.

Iron Pickaxe Mission: Life Fourth Gear Broadcast

The range is limited to 0 to 20, divided into 0, 1 to 8, 9 to 16, 17 to 20, and only one gear is output. Check with nine boundary test values and explain why only the last one is left in the last else.

Diamond Pickaxe Mission: Weather and Equipment Consultant

Read clear, rain, thunder and yes/no. Verify first, then deal with thunderstorms and shelter, and finally give general weather advice. When adding armor conditions, write down the priorities first and test at least 6 groups.

Before leaving, can you clarify: What is the difference between independent if and branch chain? Why can't 0 be caught by ≤ 8 first? Which set of boundary data caught your bug? The next lesson will use switch to organize a set of fixed options.

Technical check: Oracle Java Language Specification Chapter 14, if, code blocks and local variables.
<https://docs.oracle.com/javase/specs/jls/se21/html/jls-14.html>