

JAVA / LESSON 08 OF 40

The Biome Navigator

A choice, against a clear route list

This lesson uses JDK 21, and there is no need to turn on the preview function. | 100-120 minutes



Create one of eight biome transportation stations

Understand the brakes of traditional switches, understand arrow-type switches and return values; clean up input, receive unknown options, and finally create a console traffic station with one of eight choices.

Before you start: It is recommended to complete lessons 1-7 first. This lesson uses JDK 21

Read the full lesson and copy the code:

<https://teacher-ma.de/minecraft-java-lesson.php?lesson=8&lang=en>

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

In this lesson

01. There are so many street signs that they have grown into a jungle
02. Four parts of traditional switch
03. If break is missed, the mine car will drive past the station.
04. Arrow type: Select a grid to execute it
05. switch expression: return a result
06. Multi-sentence rules, use yield to return the value
07. Clear the ticket first, then find the platform
08. Several labels share one rule
09. Stop sign emergency room
10. Predict first, then run, then explain
11. Project Blueprint: Choose one of eight transportation stations
12. Traffic Station Code: Menu and Destination
13. Traffic Station Code: Verification and Materials
14. Normal, boundary, and error acceptance
15. Add a diamond to your menu

01 / 15

There are so many street signs that they have grown into a jungle



Steve nailed a road sign at the entrance of the village: Go to the desert to the left, go to the jungle to the right, go to the snowfield and go around again... On the eighth block, the vines have tied him into a "human bookmark." Alex arranges destinations into menus: first looking at the same selection value, then looking for the corresponding label.

switch is good at organizing a fixed list: a number or word, compared with several clear options. Whether the life is lower than 8 requires range judgment, which is still suitable for if; whichever item the destination number equals, it is suitable for switch.

- Why do traditional colon switches often require break?
- How to ensure that the arrow switch will not fall into the next route?
- How to get clear prompts for blank, capital and unknown numbers?

The Java 11 environment used in Lessons 1 to 7 is capable of running traditional switches. The modern writing methods of this lesson are all run according to JDK 21, and the editor's project JDK must also be set to 21. Destination and supplies are story practice menus, and printing the text will not initiate game transfer.

02 / 15

Four parts of traditional switch

switch takes a value to select, case writes a clear label, and default catches unlisted values. The colon type enters from the hit label, and break is responsible for leaving the switch and then continuing to execute the outside code.

Snippet: place inside main and run separately

```
String biome = "desert";
switch (biome) {
    case "desert":
        System.out.println("Desert Route");
        break;
    case "jungle":
        System.out.println("Jungle Route");
        break;
    default:
        System.out.println("Routes not listed");
}
System.out.println("End of route check");
```

Output

```
Desert Route
End of route check
```

Complete value matches label

biome

Label	Path
case "desert"	Desert route, then break
case "jungle"	Jungle route, then break
default	Routes not listed

String case compares text content, but is case-sensitive. Change biome to DESERT and desert will not be hit. The following short examples are put into main to run.

03 / 15

If break is missed, the mine car will drive past the station.



Intentional leakage of brakes: observation throughout execution

```
int station = 2;
switch (station) {
    case 1: System.out.println("Pingyuan Station");
    case 2: System.out.println("Desert Station");
    case 3: System.out.println("Snowfield Station");
    default: System.out.println("Repair station");
}
```

Output

```
Desert Station
Snowfield Station
Repair station
```

It only matches case 2, but executes the following statements all the way from there. This is called penetration. The subsequent case labels will not be reused as condition checks; even the default statement may be executed.

Snippet: place inside main and run separately

```
int station = 2;
switch (station) {
    case 1: System.out.println("Pingyuan Station"); break;
    case 2: System.out.println("Desert Station"); break;
    case 3: System.out.println("Snowfield Station"); break;
    default: System.out.println("Repair station");
}
```

Output

Desert Station

Test 1, 2, 9, compare the output of the two versions. Breakthrough is legal syntax and can be used for shared actions; when a "one vote, one stop" menu is required, missing a break is usually a logic error.

04 / 15

Arrow type: Select a grid to execute it

Use -> after case. After this rule is completed, it will not run through to the next rule, and there is no need to add a break. To do multiple things, put them in a pair of curly braces.

Snippet: place inside main and run separately

```
String biome = "jungle";
switch (biome) {
    case "desert" -> System.out.println("Desert Route");
    case "jungle" -> System.out.println("Jungle Route");
    case "snow" -> System.out.println("Snowfield route");
    default -> System.out.println("Unknown route");
}
```

Output

Jungle Route

Arrow rule: leave this switch after execution

Select value

Label	Path
desert ->	Only perform desert actions

Label	Path
jungle ->	Only perform jungle actions
default ->	Prompt unknown route

Snippet: place inside main and run separately

```
int station = 4;
switch (station) {
    case 4 -> {
        System.out.println("Seolwon Station opens");
        System.out.println("View the practice material list");
    }
    default -> System.out.println("Other sites");
}
```

Output

```
Seolwon Station opens
View the practice material list
```

05 / 15

switch expression: return a result

switch can also generate a value and load it into a variable. Each route must produce an appropriate result, without allowing one input to lead to an empty treasure chest. Here default ensures that the menu always has a String result.

Snippet: place inside main and run separately

```
String biome = "desert";
String advice = switch (biome) {
    case "desert" -> "Check Desert Route";
    case "jungle" -> "Check Jungle Route";
    case "snow" -> "Check Snowfield Route";
    default -> "Unknown route";
};
System.out.println(advice);
```

Output

```
Check Desert Route
```

Load the selection result into the variable

Step	Explanation
1. biome	Provide this selection value.
2. switch expression	A hit rule returns a value.
3. String advice	Receive the results and output them uniformly.

Snippet: place inside main and run separately

```
String zone = "far";
int ticket = switch (zone) {
    case "near" -> 2;
    case "far" -> 5;
    default -> 0;
};
System.out.println("Practice Fare: " + ticket);
```

Output

```
Practice Fare: 5
```

The end of the expression is `};` because the entire sentence is declaring and assigning values. The previous page is just a switch statement, and there is no need for this semicolon after the closing curly brace at the end. Ticket prices are practice points and do not represent real game transactions.

06 / 15

Multi-sentence rules, use yield to return the value

If there are multiple sentences after the arrow of the expression, wrap them in curly braces, and then use `yield` to return the result. `break` means to leave, and `yield` means to hand the result to the outer switch expression.

Snippet: place inside main and run separately

```
String zone = "far";
int ticket = switch (zone) {
    case "near" -> 2;
    case "far" -> {
        System.out.println("Preparing long distance route");
        yield 5;
    }
    default -> 0;
};
System.out.println("Fare=" + ticket);
```

Output

```
Preparing long distance route  
Fare=5
```

Statements and expressions have different tasks

Element	Explanation
switch statement	To execute an action, multiple sentences can be printed. After the mission, go down.
switch expression	Produce a value. Each input must have a result, and the result types must be compatible.

This lesson does not use switch pattern matching or preview functions. First, familiarize yourself with the fixed number and string menus: look at the selection value, look at the label, look at the action or return value, look at the unknown option.

07 / 15

Clear the ticket first, then find the platform



Players may enter JuNgLe, or they may enter more spaces. First strip, then press Locale.ROOT to convert to lowercase, so that uppercase and lowercase and blank spaces at both ends do not affect the internal commands of the program.

Snippet: place inside main and run separately

```
String raw = " JuNgLe ";
String biome = raw.strip()
    .toLowerCase(java.util.Locale.ROOT);
String advice = switch (biome) {
    case "desert" -> "Desert Route";
    case "jungle" -> "Jungle Route";
    case "snow" -> "Snowfield route";
    case "" -> "No input yet";
    default -> "Unknown route: " + biome;
};
System.out.println(advice);
```

Output

```
Jungle Route
```

default To make the problem clear, don't secretly interpret the misspelled jungel as plain. Input errors and ordinary default suggestions are two different tasks; first explain the default behavior when designing.

There is no case null String switch in this lesson. An exception will be thrown when the selected value is null, and default will not replace the null check. NextLine normally reads a string; no readable line is another input exception.

08 / 15

Several labels share one rule

The arrow type can use commas to juxtapose multiple fixed tags. In the traditional colon style, you can write several case labels in succession so that they can share the following statements. Don't just move commas into old grammar.

Snippet: place inside main and run separately

```
String dimension = "nether";
String level = switch (dimension) {
    case "nether", "end" -> "Practice high-risk routes";
    case "overworld" -> "Practice common routes";
    default -> "Unknown dimensions";
};
System.out.println(level);
```

Output

```
Practice high-risk routes
```

Eight-choose one menu: the number is the fixed label

Number	Destination	Number	Destination
1	plain	5	Ocean
2	Desert	6	Swamp
3	Jungle	7	Badlands
4	Yukihara.	8	Mushroom Island.

Snippet: place inside main and run separately

```
int day = 7;
String plan = switch (day) {
    case 1, 2, 3, 4, 5 -> "After-school programming";
    case 6, 7 -> "Weekend construction";
    default -> "Wrong date";
};
System.out.println(plan);
```

Output

```
Weekend construction
```

09 / 15

Stop sign emergency room



The label must be clear and no competition for position can be made

- If case 2 occurs repeatedly, compilation will fail. Each same label appears only once.
- In the fixed value menu of this lesson, the case label must be a constant; ordinary variable local variables are not constant labels.
- To judge the range of health ≤ 8 , use if; do not stuff the comparison condition into a fixed value case.
- If the String switch expression only lists two words and does not have default, there is no guarantee that every input will have a result.

Snippet: place inside main and run separately

```
final int DESERT = 2;
int station = 2;
switch (station) {
    case DESERT -> System.out.println("Desert Station");
    default -> System.out.println("Other stations");
}
```

Output

Desert Station

Here final int DESERT is initialized with constant expression 2, which can be used as a label. All final variables cannot be treated as compile-time constants. When debugging, first read the first relevant error report, and then check the label, break and result coverage.

10 / 15

Predict first, then run, then explain

Leave evidence for choice

Step	Explanation
1. Predict first	Write the cleaned values and hit labels.
2. Run again	Compare the actual output of each line.
3. I can explain.	Explain why other rules are not executed.

Snippet: place inside main and run separately

```
String biome = " SNOW ".strip()
    .toLowerCase(java.util.Locale.ROOT);
int boots = switch (biome) {
    case "snow" -> 1;
    default -> 0;
};
System.out.println(boots);
```

Snippet: place inside main and run separately

```
int day = 6;
String plan = switch (day) {
    case 6, 7 -> "Weekend construction";
    case 1, 2, 3, 4, 5 -> "Collection after class";
    default -> "Wrong date";
};
System.out.println(plan);
```

Check: the first question outputs 1, and after cleaning it is snow; the second question outputs "weekend construction", and 6 hits the first group. Change day to 0, 1, 5, 7, 8 and predict the results before running.

If the result is different from expectation, first confirm what the selected value is. If the input cleaning is wrong, no matter how beautiful the switch is later, it can only choose the wrong route.

11 / 15

Project Blueprint: Choose one of eight transportation stations

Only apply for one ticket at a time: display the complete menu, read in the number, analyze, select the route, and output the practice material list. There is no loop, and you need to buy a second ticket and run it again.

Input, selection, and output each take care of one thing

Step	Explanation
1. Read entire line	Clean up the blanks at both ends, and then parseInt.
2. switch	Numbers 1 to 8 are mapped to destinations.
3. Print	A list of practice supplies is attached to the legal route.

- A legal number is output to a destination, and other destinations are not allowed.
- 0, 9 and other out-of-bounds integers: report number range error.

- abc, decimal, or large number that does not fit into int: Report an integer input error.
- Entering "3" still selects jungle; 0002 and +5 can be parsed into 2 and 5.

The first switch's default returns the empty string as an internal invalid token, followed by an error with isEmpty; it does not appear as a destination. The default of the second switch provides general practice supplies for legal routes, this time the default value is not a typo.

12 / 15

Traffic Station Code: Menu and Destination

Create a new BiomeTransit.java, using JDK 21. Put the codes of this page and the next page into the same file in order; the last switch of this page has not ended yet.

BiomeTransit.java · First half, continue on next page

```
import java.util.Scanner;

public class BiomeTransit {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("=== Biome Transportation Station ===");
        System.out.println("1 plain  2 Desert  3 Jungle  4 YukiHara.");
        System.out.println("5 Ocean  6 Swamp  7 Badlands  8 Mushroom Island.");
        System.out.println("Destination number 1 to 8: ");
        String text = in.nextLine().strip();
        try {
            int choice = Integer.parseInt(text);
            String destination = switch (choice) {
                case 1 -> "plain";
                case 2 -> "Desert";
                case 3 -> "Jungle";
                case 4 -> "YukiHara.";
                case 5 -> "Ocean";
                case 6 -> "Swamp";
                case 7 -> "Badlands";
                case 8 -> "Mushroom Island.";
                default -> "";
            };
        }
```

Push three tickets by hand: "3" clears to 3 and hits the jungle; 0 goes to default and gets an empty string; abc fails at parseInt and jumps to catch.

13 / 15

Traffic Station Code: Verification and Materials

BiomeTransit.java · Immediately following the previous page

```
        if (destination.isEmpty()) {
            System.out.println("Number must be within 1 to 8 between");
        } else {
            String supply = switch (choice) {
                case 2 -> "Water and food";
                case 3 -> "Scissors and torches";
                case 4 -> "Leather boots";
                case 5 -> "Boat.";
                default -> "Bed and food";
            };
            System.out.println("Destination: " + destination);
            System.out.println("Practice material list: " + supply);
        }
    } catch (NumberFormatException e) {
        System.out.println("Please enter an integer number, for example 3");
    }
}
```



The second switch only processes numbers that are already legal. Entering 3 should output "Destination: Jungle" and "Practice Supply List: Scissors and Torches" after the full menu.

The supply list is a story mission, not a complete survival strategy. "The route is prepared" is just program text; to connect the code into the game, you need to learn the MOD development interface later.

14 / 15

Normal, boundary, and error acceptance

Traffic station acceptance form

Enter	Intended destination or error	Practice material list
1	plain	Bed and food
2	Desert	Water and food
3	Jungle	Scissors and torches
4 / 5	Snowfield/Ocean	Leather boots / boat
6 / 7 / 8	Swamp/Badlands/Mushroom Island	Bed and food
0 / 9 / -1	Number must be between 1 and 8	Do not output material list
abc / 4.5 / too large integer	Please enter an integer number, such as 3	Do not output material list

Use input to prove "one vote, one stop"

- Each of the eight legal numbers is tested once to check the destination and materials.
- Then test the empty string, blanks at both ends, 0002, +5.
- If you run the same number repeatedly, the output should be consistent.
- Deliberately repeat a case label, observe the compilation error, and then recover.

Don't just record "success." Write down the input, prediction, and actual output; if there is an extra next stop, go back and check the break in traditional writing. If all use default, first check whether the value and label are consistent.

15 / 15

Add a diamond to your menu



Wooden Pickaxe Task: Four Tool Instructions

Accept pickaxe, axe, shovel, and hoe, and output the tool usage respectively; unknown words will report an error. First clean up whitespace and case, and test four legal words, the empty string and sword.

Iron Pickaxe Task: Weekly Task List

Read 1 to 7, 1 to 5 share the after-school plan, and 6 and 7 share the weekend plan. 0, 8 and abc check for range or parsing errors respectively. Explain why multiple tags can share the same result.

Diamond Pickaxe Task: Three-dimensional menu plus confirmation

Select overworld, nether, end and then read yes/no. Unknown dimensions or confirmation words will report an error, and only yes will show that the preparation is completed. Test at least 8 groups, covering whitespace, upper and lower case, each dimension and unknown words.

Backpack review: Use switch for the fixed list and if for the range condition. Pay attention to break in colon expressions, and do not break through arrow expressions; the expression must return the result. The next lesson uses for to hand off duplication to a counter.

Technical Check: Oracle Java 21 Language Guide, switch expressions and statements.

<https://docs.oracle.com/en/java/javase/21/language/switch-expressions-and-statements.html>