

JAVA / LESSON 15 OF 40

Hotbar Arrays

Array indexing and traversal

This chapter uses JDK 21 and does not require the preview function. | 90-110 minutes

Create a nine-frame hotbar manager

Can create and initialize arrays; can read and replace elements with indexes from 0 to length-1; can choose ordinary for or enhanced for traversal; and finally realize the display, replacement, search and vacancy statistics of the nine-grid hotbar.
Prerequisites: variables, loops, methods and scope. The array is like a row of boxes of the same type, each grid has a house number starting from 0.

Before you start: It is recommended to complete Lessons 12-14 first

Read the full lesson and copy the code:

<https://teacher-ma.de/minecraft-java-lesson.php?lesson=15&lang=en>

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

Nine-frame hotbar, numbered from 0

I n d e x	0	1	2	3	4	5	6	7	8	9	1 0	1 1	1 2	1 3	1 4	1 5	1 6	1 7	1 8	1 9	2 0	2 1	2 2	2 3	2 4	2 5
C h a r a c t e r	T	h	e		s	w	o	r	d		c	o	n	t	a	i	n	s		f	i	r	e	.	.	.

Length 9, legal index 0 to 8. A dot represents a grid that currently has no items.

In this lesson

01. Before you begin: Put the examples in the right place
02. Nine-frame hotbar starts from scratch
03. Array is a row of boxes of the same type
04. Create array and default value
05. Read and replace with index
06. Normal for grid-by-frame inspection
07. Enhancement for only viewing items
08. Statistical total average maximum value
09. Search for items and calculate vacancies
10. Array Fault Clinic
11. Challenge Blueprint Nine-frame Quick Bar
12. Challenge Implementation Show replacement statistics
13. Expansion tasks and stage review
14. Own acceptance record

01 / 14

Before you begin: Put the examples in the right place

This chapter uses JDK 21 and does not require the preview function. A complete program is given for each common example, saved with annotated file names. Multiple examples are placed in their own folders to avoid auxiliary classes with the same name.

Terminal command: Replace the Chinese placeholder with the actual name

```
javac -encoding UTF-8 ExampleFile.java  
java ExampleClass
```

Class names are case-sensitive. If the complete challenge code is divided into multiple sections, please connect them in the same file in order. Do not create separate classes for the continued sections. Run verification only lists the program output; additional input sequences are provided where input is required.

Learning action: first write down your prediction, compare it after running, and then change only one condition. Experiments marked "Intentional Bug" are used to identify bugs; they are clearly distinguished from ready-to-run examples.

02 / 14

Nine-frame hotbar starts from scratch

The administrator shouted: "Put the torch in the first slot!" The programmer put it into slots[0]. The administrator was silent for three seconds and decided to learn arrays first.

An array places a group of elements of the same type in a grid arranged by index. The array length is fixed and each element is accessed by index.

Nine-frame hotbar: the index of the last frame is 8

I n d e x	0	1	2	3	4	5	6	7	8	9	1 0	1 1	1 2	1 3	1 4	1 5	1 6	1 7	1 8	1 9	2 0	2 1	2 2	2 3	2 4	2 5
C h a r a c t e r	T	h	e		s	w	o	r	d		c	o	n	t	a	i	n	s		f	i	r	e	.	.	.

A single symbol is used to represent the item here; there are 9 grids in total, the index is 0 to 8, and the length is 9.

Starting with four questions

- Why are the legal indexes of an array of length 9 from 0 to 8?
- After creating a new String array, what is the grid without items?
- What tasks are ordinary for and enhanced for suitable for?
- How to avoid errors caused by cross-border and null?

Challenge of this lesson: Complete the nine-grid hotbar: display all grids, replace designated positions, find items, and count empty spaces.

03 / 14

Array is a row of boxes of the same type

String[] slots represents a set of string grids, and new String[9] creates nine positions. The array variable holds a reference to the entire row of grids.

Nine-frame hotbar: the index of the last frame is 8

I n d e x	0	1	2	3	4	5	6	7	8	9	1 0	1 1	1 2	1 3	1 4	1 5	1 6	1 7	1 8	1 9	2 0	2 1	2 2	2 3	2 4	2 5
C h a r a c t e r	T	h	e		s	w	o	r	d		c	o	n	t	a	i	n	s		f	i	r	e	.	.	.

A single symbol is used to represent the item here; there are 9 grids in total, the index is 0 to 8, and the length is 9.

L15Example02.java · Complete independent example

```
public class L15Example02 {  
    public static void main(String[] args) {  
        String[] slots = {"sword", "Bread.", "Torch"};  
        System.out.println(slots.length);  
        System.out.println(slots[0]);  
        System.out.println(slots[2]);  
    }  
}
```

Output

```
3  
sword  
Torch
```

Fixed length: After the array is created, 9 grids cannot be changed into 10 grids; elements in the grid can be replaced, but the length cannot be modified.

04 / 14

Create array and default value

Java will fill in the default value for the new array. The space in `String[]` is null, it is not an empty string, nor is it a string with "air" written on it.

The new box is first filled with default values

Element	Explanation
int element	Default 0
double element	Default 0.0
boolean element	Default false
Reference element	Default null

L15Example03.java · Complete independent example

```
public class L15Example03 {  
    public static void main(String[] args) {  
        int[] counts = new int[2];  
        boolean[] locks = new boolean[2];  
        String[] slots = new String[9];  
        System.out.println(counts[0]);  
        System.out.println(locks[0]);  
        System.out.println(slots[0]);  
    }  
}
```

Output

```
0  
false  
null
```

Select a space representation: null can be used, or "air" can be used uniformly. Both are fine, but the entire program must be consistent and checked before use.

05 / 14

Read and replace with index

Indexes are placed in square brackets. Reading does not delete the item, and assignment replaces the original element at that position.

Nine-frame hotbar: the index of the last frame is 8

I n d e x	0	1	2	3	4	5	6	7	8	9	1 0	1 1	1 2	1 3	1 4	1 5	1 6	1 7	1 8	1 9	2 0	2 1	2 2	2 3	2 4	2 5
C h a r a c t e r	T	h	e		s	w	o	r	d		c	o	n	t	a	i	n	s		f	i	r	e	.	.	.

A single symbol is used to represent the item here; there are 9 grids in total, the index is 0 to 8, and the length is 9.

L15Example04.java · Complete independent example

```
public class L15Example04 {
    public static void main(String[] args) {
        String[] slots = new String[9];
        slots[1] = "Bread.";
        String before = slots[1];
        slots[1] = "Carrot.";
        System.out.println(before + " -> " + slots[1]);
    }
}
```

Output

```
Bread. -> Carrot.
```

Index boundary: Legal index satisfies $\text{index} \geq 0$ && $\text{index} < \text{slots.length}$. `slots[9]` has gone out of the nine-slot hotbar.

06 / 14

Normal for grid-by-frame inspection

Use ordinary for when indexing is required. The loop condition is written $< \text{length}$ because the last legal position is $\text{length} - 1$.

Index loop gets location and items at the same time

Element	Explanation
Start	i equals 0
Conditions	i is less than length
Read	Access slots[i]
Update	i plus 1, continue inspection

L15Example05.java · Complete independent example

```
public class L15Example05 {  
    public static void main(String[] args) {  
        String[] slots = {"sword", null, "Torch"};  
        for (int i = 0; i < slots.length; i++) {  
            String text = slots[i] == null ? "Vacancy" : slots[i];  
            System.out.println(i + ": " + text);  
        }  
    }  
}
```

Output

```
0: sword  
1: Vacancy  
2: Torch
```

Loop contract: initialization starts from 0, the condition is $i < \text{length}$, the loop body only uses legal slots[i], and i increases by 1 in each round.

07 / 14

Enhancement for only viewing items

When you only care about elements and do not need position, enhance for to be more concise. The loop variable gets the element value, but it cannot be used to directly replace the array grid.

Enhance for to take out each element in turn

Element	Explanation
item	Copy of current round element value
Read	Suitable for piece-by-piece statistics
Reassignment	Array grid will not be replaced
Replacement elements	Need slots[i] assignment

L15Example06.java · Complete independent example

```
public class L15Example06 {  
    public static void main(String[] args) {  
        String[] slots = {"Bread.", "Torch"};  
        for (String item : slots) { item = "diamond"; }  
        for (String item : slots) { System.out.println(item); }  
    }  
}
```

Output

```
Bread.  
Torch
```

item is a local variable that receives the element value this round. Reassigning it will not replace the array grid; to replace the grid, use slots[i] = ... Lesson 18 will study the situation of modifying the object pointed to by an element.

How to choose

- Need index or replace grid: ordinary for.
- Read only each element: enhanced for.
- It is necessary to access the front and rear elements at the same time: ordinary for is more suitable.

Common misunderstanding: Reassigning the loop variable of enhanced for only changes the local variable of this loop and does not replace the original array element.

08 / 14

Statistical total average maximum value

After combining arrays and loops, the entire set of data can be processed at once. The maximum algorithm is initialized with the first element to avoid guessing a starting point.

Numeric array enters statistical workbench

Element	Explanation
total	Add the number of each grid
average	Convert to decimal and divide by length
maximum	Starting from the first element
Empty array	Check first, then access and divide

L15Example07.java · Complete independent example

```
public class L15Example07 {  
    public static void main(String[] args) {  
        int[] counts = {3, 12, 7, 18, 5};  
        int total = 0, maximum = counts[0];  
        for (int count : counts) {  
            total += count;  
            maximum = Math.max(maximum, count);  
        }  
        double average = (double) total / counts.length;  
        System.out.println(total + "," + average + "," + maximum);  
    }  
}
```

Output

```
45,9.0,18
```

The array of this example is known to be non-empty. The general method should handle `length == 0` before reading `counts[0]` or calculating the average; an empty array has no maximum element and the average is not defined.

Empty array alert: Make sure `length > 0` before accessing `ores[0]`. An empty array does not have a first element, and the average cannot be calculated directly.

09 / 14

Search for items and calculate vacancies

When searching an array, null is processed first, and then the string method is called. Whether to continue after finding it depends on whether you want to find the first place or count all.

Inspect each grid and record the findings

Element	Explanation
find	Return the first index or -1
Vacancy	This course uses null uniformly
Compare	Non-null target calls equals
Number of spaces	Cell by cell counting

L15Example08.java · Complete independent example

```
public class L15Example08 {
    static int find(String[] slots, String target) {
        if (target == null) return -1;
        for (int i = 0; i < slots.length; i++)
            if (target.equals(slots[i])) return i;
        return -1;
    }
    static int emptyCount(String[] slots) {
        int result = 0;
        for (String item : slots) if (item == null) result++;
        return result;
    }
    public static void main(String[] args) {
        String[] slots = {"sword", null, "Torch", null};
        System.out.println(find(slots, "Torch"));
        System.out.println(find(slots, "diamond"));
        System.out.println(emptyCount(slots));
    }
}
```

Output

```
2
-1
2
```

Safety comparison: writing "Diamond Sword".equals(slots[i]) will not throw a NullPointerException even if the slot is null.

10 / 14

Array Fault Clinic

`ArrayIndexOutOfBoundsException` usually indicates that the index is out of bounds; `NullPointerException` often indicates that a method was called on null. The two practices are different.

Out of bounds and null are two common creepers

Element	Explanation
Crossing the line	Index length is illegal
Error Boundary	Less than or equal to will move one more square
null array	Unable to read length
Empty array	Length 0, but not null

Fault experiment: Index 3 exceeds the legal range of 0 to 2, and `ArrayIndexOutOfBoundsException` will occur. Errors are intentionally left below, please anticipate them first and then try to fix them.

Fault experiment: `slots` does not point to the array yet, `NullPointerException` will occur. Errors are intentionally left below, please anticipate them first and then try to fix them.

Repair one piece at a time: do not change the loop condition and array length at the same time. First prove which round made the error, and then make minimal modifications.

BadIndex.java · Intentional error example

```
public class BadIndex {
    public static void main(String[] args) {
        int[] counts = new int[3];
        System.out.println(counts[counts.length]);
    }
}
```

BadArrayNull.java · Intentional error example

```
public class BadArrayNull {
    public static void main(String[] args) {
        String[] slots = null;
        System.out.println(slots.length);
    }
}
```

Emergency sequence

- Read the exception name and the first line of your own code.
- Print index, length and current element.
- Use `< length` for index, and check for null before using the reference.

11 / 14

Challenge Blueprint Nine-frame Quick Bar

The challenge class is called `HotbarManager`. The hotbar is fixed to nine spaces, and the initial items are placed at positions 0, 1, 2 and 8.

Display, replace, search and count four functions

Element	Explanation
show	Display frame by frame
replace	Check index and items first
find	Find first match
emptyCount	Statistics null grid

Acceptance sequence

- First display the initial hotbar.
- Replace position 1 from bread to golden carrot.
- Searching for torches should return 8.
- To count the gaps, 5 should be returned.
- Try index 9, the program must not crash.

Acceptance and inspection forms

Function	Enter	Output or effect
show	Array	Display 0 to 8 one by one
replace	Array index new item	Replace and return true when legal
find	Array item name	Return the first index or -1
emptyCount	Array	Return null grid number

Minimal version: Complete show and replace first, then add find and emptyCount. Run each method added.

12 / 14

Challenge Implementation Show replacement statistics

Create new HotbarManager.java. The complete program uses a normal for to display the index and an enhanced for to count the gaps.

Four array methods collaboratively manage the hotbar

Element	Explanation
Initial state	Display all nine grids
Replacement	Replace grid No. 1 with carrot
Search	Torch at index 2
Vacancy	There are 5 in total

Acceptance: After running, check a total of nine lines from 0 to 8; confirm that position 1 has been replaced; then test the four indexes -1, 0, 8, and 9.

HotbarManager.java · Complete standalone example

```
public class HotbarManager {
    static void show(String[] slots) {
        for (int i = 0; i < slots.length; i++) {
            String text = slots[i] == null ? "Vacancy" : slots[i];
            System.out.println(i + ": " + text);
        }
    }
    static boolean replace(String[] slots, int index, String item) {
        if (index < 0 || index >= slots.length
            || item == null || item.isBlank()) return false;
        slots[index] = item;
        return true;
    }
    static int find(String[] slots, String target) {
        if (target == null) return -1;
        for (int i = 0; i < slots.length; i++)
            if (target.equals(slots[i])) return i;
        return -1;
    }
    static int emptyCount(String[] slots) {
        int result = 0;
        for (String item : slots) if (item == null) result++;
        return result;
    }
    public static void main(String[] args) {
        String[] slots = new String[9];
        slots[0] = "Iron sword"; slots[1] = "Bread.";
        slots[2] = "Torch"; slots[8] = "Map";
        show(slots);
        System.out.println(replace(slots, 1, "Carrot."));
        System.out.println("Modified 1 Number grid: " + slots[1]);
        System.out.println("Torch Index: " + find(slots, "Torch"));
        System.out.println("Vacancy: " + emptyCount(slots));
    }
}
```

Output

```
0: Iron sword
1: Bread.
2: Torch
3: Vacancy
4: Vacancy
5: Vacancy
6: Vacancy
7: Vacancy
8: Map
true
Modified 1 Number grid: Carrot.
Torch Index: 2
Vacancy: 5
```

Expansion tasks and stage review

This is the last lesson in the basic grammar stage. Lesson 16 combines variables, conditionals, loops, methods, and arrays into a "Backpack Manager" project.

The third-level task uses arrays into real data

Element	Explanation
Wooden Pickaxe	Ore array is displayed grid by grid
Iron Pickaxe	Statistics of collection volume per week
Diamond Pickaxe	Implement circular shift
Review	Test length 0 and 1

Third gear operation

- Wooden pickaxe: Put six kinds of ores into the String array and print the index and name one by one.
- Iron Pickaxe: Use an int array to record the mining quantity for seven days a week, and output the total number, average value and maximum value.
- Diamond Pickaxe: Realize that the hotbar rotates one space to the right, and the last space returns to position 0.

Acceptance and inspection forms

Self-evaluation	Can	Still need to practice
I can tell the legal index range	☐	☐
I would choose two for	☐	☐
I can handle out of bounds and null	☐	☐

Next lesson: Lesson 16 is the stage project "Backpack Administrator". You'll combine inputs, tests, loops, methods, and arrays into the same runnable program.

14 / 14

Own acceptance record

Write expectations before running, and record actual results after running. Normal cases show that the program does things; edge cases show that it knows when to stop; error cases show that it rejects unreasonable requests.

Write the evidence into the record

Category	Input of your choice	Expectation and actual
Normal	A common value or a successful operation	Is it consistent line by line?
Border	Minimum value, maximum value and adjacent value	Is there one less square and one more step?
Error	Illegal quantity, missing object or incorrect syntax	Explicitly reject? Is the status maintained?

- Write a bug as: triggering conditions, phenomena, causes, and minimal repairs.
- Organize a rule worth remembering and explain it in your own words. Don't just copy the definition.
- Change an example, keep a record of the test, and let the results prove your understanding.

Technical verification basis: Java SE 21 language specification, scope and access control (Chapter 6), classes and constructors (Chapter 8), method invocation (Chapter 15).

<https://docs.oracle.com/javase/specs/jls/se21/html/index.html>