Arrays


FROM tbl_room WHERE room_checked='0'"
"room_name_full"];
tbl_disposition.disposition_user = u.user_ID
MacBook

Lecture Overview

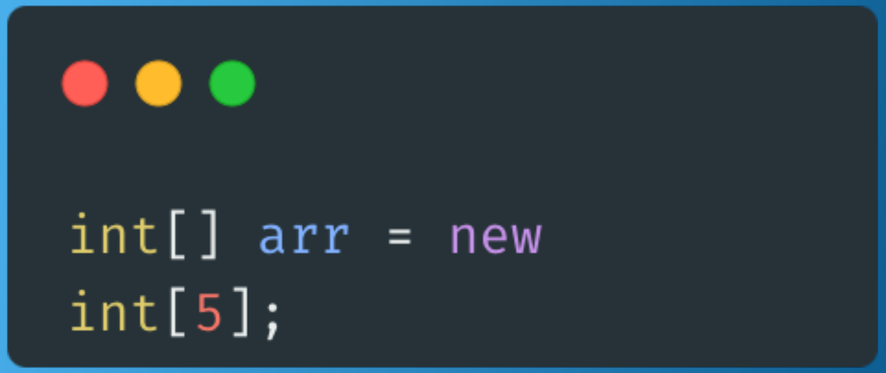
- Introduction to Arrays
- One-Dimensional Arrays
- Multi-Dimensional Arrays
- Common Array Operations
- Enhanced For Loop and Arrays Class
- Memory Considerations
- Practical Examples

Introduction to Array

What is an Array?

- A collection of variables of the same type.
- Fixed size.
- Indexed from 0 to n-1.

Syntax: dataType[] arrayName;



Declaring and Initializing Arrays

Declaration:

- `dataType[] arrayName;` or `dataType arrayName[];`

Initialization:

- `arrayName = new dataType[size];`

```java
int[] arr = new int[5];  // Declaration and initialization
```

```java
int[] arr = {10, 20, 30, 40, 50};
```

One-Dimensional Arrays

Accessing elements:

- Use the index to access elements.
- Example: `myArray[2]` gives `3`.

Modifying elements:

```
int[] myArray = {1, 2, 3, 4, 5};
```

```
myArray[1] = 10;
```

Multi-Dimensional Arrays

Syntax:

```
dataType[][] arrayName = new dataType[rows][cols];
```

Accessing Elements:

```
int element = matrix[1][2];  // Outputs 6
```

2D Array:

```
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Operation on Arrays

Traversing Arrays:

- Use a for loop or enhanced for loop.

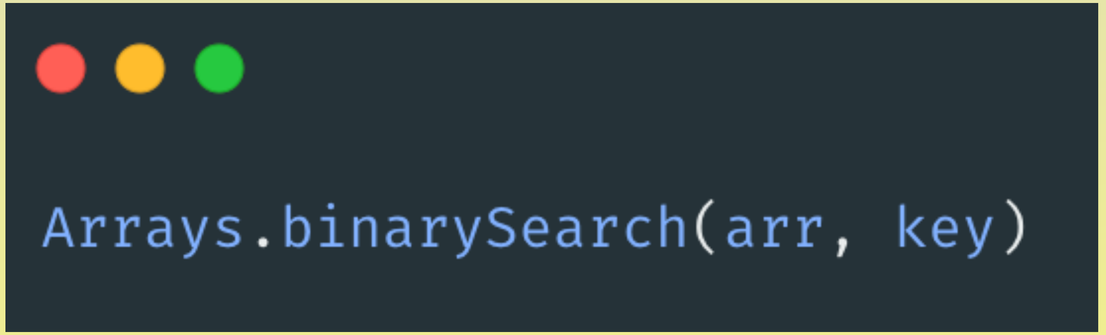
```java
for (int i = 0; i < myArray.length; i++) {
    System.out.println(myArray[i]);
}
```

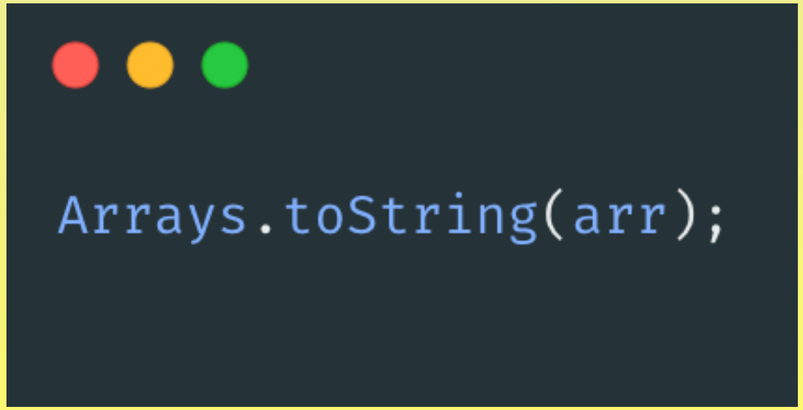
```java
for (int value : myArray) {
    System.out.println(value);
}
```

Arrays Class (java.util.Arrays)

Methods in java.util.Arrays:

- sort(): Sorts the array.
- binarySearch(): Searches for an element.
- toString(): Converts array to String.





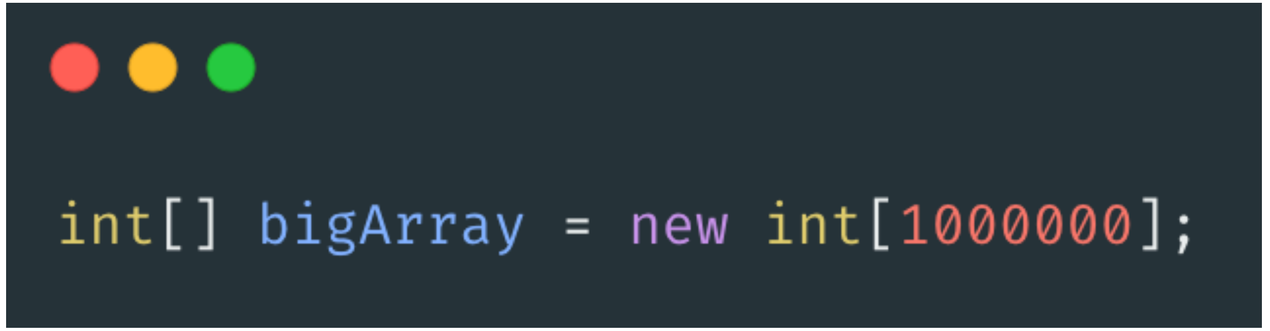
Memory Considerations

Fixed Size: Once an array is created, its size cannot change.

Heap Memory: Arrays are stored in heap memory.

Performance: Accessing array elements is fast, but large arrays can consume a lot of memory.

Example:



Practical Examples

Example 1: Finding the average of elements in an array.

```
int sum = 0;
for (int value : myArray) {
    sum += value;
}
double average = sum / (double) myArray.length;
```

Example 2: Matrix Multiplication (2D array).

```
int[][] product = new int[rows][cols];
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        product[i][j] = matrix1[i][j] * matrix2[i][j];
    }
}
```

Enhancing Array Usage

Enhanced For Loop: A simplified way to iterate over arrays.

Arrays Class: Simplifies common array operations like sorting and searching.

Memory Management Tips:

- Avoid creating excessively large arrays if possible.
- Consider alternative data structures (e.g., ArrayList) if the array size needs to be dynamic.

Summary

- Arrays are a foundational data structure in Java.
- One-dimensional and multi-dimensional arrays allow for efficient data storage and access.
- Java's Arrays class provides useful methods for common operations.
- Memory management and performance should always be considered when working with large arrays.

Thank You