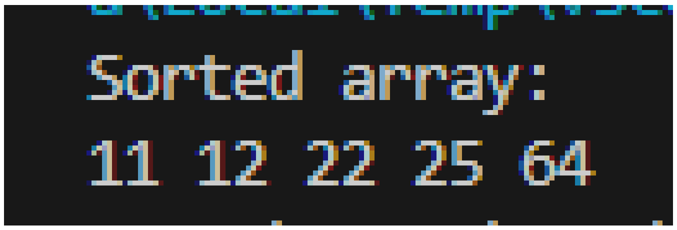
SelectionSort

```
public class selectionsort
{
  public static void selectionSort(int[] arr) {
    int n = arr.length;

    for (int i = 0; i < n - 1; i++) {

      int minIndex = i;
      for (int j = i + 1; j < n; j++) {
        if (arr[j] < arr[minIndex]) {
          minIndex = j;
        }
      }

      int temp = arr[minIndex];
      arr[minIndex] = arr[i];
      arr[i] = temp;
    }
  }

  public static void main(String[] args) {
    int[] arr = {64, 25, 12, 22, 11};
    selectionSort(arr);
```

```
        System.out.println("Sorted array:");
        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
}
```

Output:



```
Sorted array:
11 12 22 25 64
```

Analysis Of Selection Sort

The code implements **selection sort** to sort an array of integers. It works by iterating over the array, finding the smallest element in the unsorted portion, and swapping it with the first unsorted element. This process repeats until the array is sorted.

- **selection() function**: It loops through the array, identifies the smallest element in the unsorted part, and swaps it with the current element.
- **printArr() function**: Prints the array elements.
- **Main function**: Initializes an array, prints it before sorting, sorts it using selection sort, and then prints it after sorting.

Issue:

- There's a type (ss) in the printArr() function which needs to be removed.

Complexity:

- **Time Complexity**: **O(n^2)** in all cases (since it always scans the entire unsorted part).
- **Space Complexity**: **O(1)** because it's an in-place sort.

Output:

The program prints the array before and after sorting.