1)

```java
import java.util.HashMap;

import java.util.Map;

public class DuplicateFinder {

    public static void main(String[] args) {

        int[] array = {1, 2, 3, 4, 5, 2, 4, 6, 7, 1};

        Map<Integer, Integer> countMap = new HashMap<>();

        for (int number : array) {

            if (countMap.containsKey(number)) {

                int count = countMap.get(number);

                countMap.put(number, count + 1);

            } else {

                countMap.put(number, 1);

            }

        }


        System.out.println("Duplicates in the array:");


        for (Map.Entry<Integer, Integer> entry : countMap.entrySet()) {

            if (entry.getValue() > 1) {

                System.out.println(entry.getKey());

            }

        }

    }

}
```

2)

```java
public class QuickSort {

    public static void main(String[] args) {

        int[] array = {9, 5, 1, 8, 3, 7, 4, 6, 2};
    System.out.println("Original Array:");
    printArray(array);
    quickSort(array, 0, array.length - 1);
    System.out.println("Sorted Array:");
    printArray(array);
}
public static void quickSort(int[] array, int low, int high) {
    if (low < high) {
        int pivotIndex = partition(array, low, high);
        quickSort(array, low, pivotIndex - 1);
        quickSort(array, pivotIndex + 1, high);
    }
}
public static int partition(int[] array, int low, int high) {
    int pivot = array[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (array[j] <= pivot) {
            i++;
            swap(array, i, j);
        }
    }
    swap(array, i + 1, high);
    return i + 1;
}
public static void swap(int[] array, int i, int j) {
    int temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}
public static void printArray(int[] array) {
        for (int num : array) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

3)

```
public class BubbleSort {
    public static void main(String[] args) {
        int[] array = {9, 5, 1, 8, 3, 7, 4, 6, 2};
        System.out.println("Original Array:");
        printArray(array);
        bubbleSort(array);
        System.out.println("Sorted Array:");
        printArray(array);
    }
    public static void bubbleSort(int[] array) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    swap(array, j, j + 1);
                }
            }
        }
    }
    public static void swap(int[] array, int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
    public static void printArray(int[] array) {
        for (int num : array) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

4)

```
public class MergeSort {
    public static void main(String[] args) {
        int[] array = {9, 5, 1, 8, 3, 7, 4, 6, 2};
        System.out.println("Original Array:");
        printArray(array);
        mergeSort(array, 0, array.length - 1);
        System.out.println("Sorted Array:");
        printArray(array);
    }
    public static void mergeSort(int[] array, int low, int high) {
        if (low < high) {
            int mid = (low + high) / 2;
            mergeSort(array, low, mid);
            mergeSort(array, mid + 1, high);
            merge(array, low, mid, high);
        }
    }
    public static void merge(int[] array, int low, int mid, int high) {
        int n1 = mid - low + 1;
        int n2 = high - mid;
        int[] leftArray = new int[n1];
        int[] rightArray = new int[n2];
        for (int i = 0; i < n1; ++i) {
            leftArray[i] = array[low + i];
        }
        for (int j = 0; j < n2; ++j) {
            rightArray[j] = array[mid + 1 + j];
        }
        int i = 0, j = 0;
        int k = low;
        while (i < n1 && j < n2) {
            if (leftArray[i] <= rightArray[j]) {
                array[k] = leftArray[i];
                i++;
            } else {
                array[k] = rightArray[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            array[k] = leftArray[i];
            i++;
            k++;
        }
        while (j < n2) {
            array[k] = rightArray[j];
            j++;
            k++;
        }
    }
    public static void printArray(int[] array) {
        for (int num : array) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

5)

```
public class SelectionSort {
    public static void main(String[] args) {
        int[] array = {9, 5, 1, 8, 3, 7, 4, 6, 2};
        System.out.println("Original Array:");
        printArray(array);
        selectionSort(array);
        System.out.println("Sorted Array:");
        printArray(array);
    }
    public static void selectionSort(int[] array) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (array[j] < array[minIndex]) {
                    minIndex = j;
                }
            }
            swap(array, i, minIndex);
        }
    }
    public static void swap(int[] array, int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
    public static void printArray(int[] array) {
        for (int num : array) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

6)

```
import java.util.HashSet;
import java.util.Set;
public class SubsetChecker {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3, 4, 5, 6};
        int[] array2 = {3, 5, 1};
        boolean isSubset = isSubset(array1, array2);
        System.out.println("Array 2 is a subset of Array 1: " + isSubset);
    }
    public static boolean isSubset(int[] array1, int[] array2) {
        Set<Integer> set = new HashSet<>();
        for (int num : array1) {
            set.add(num);
        }
        for (int num : array2) {
            if (!set.contains(num)) {
                return false;
            }
        }
        return true;
    }
}
```