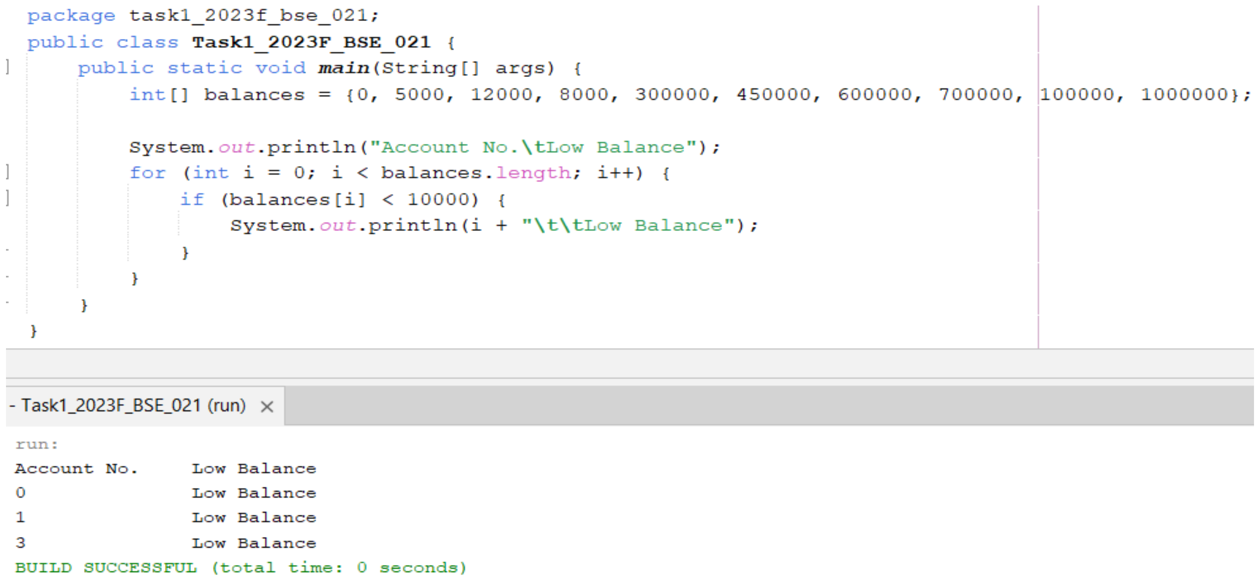
LAB # 06

Searching in a Linear Array

OBJECTIVE: To find an element in linear array using Linear Search and Binary Search.

Lab Task

1. Declare an array of size 10 to store account balances. Initialize with values 0 to 1000000. Check all array if any value is less than 10000. Show message:
   Account No. Low Balance
   Account No. Low Balance



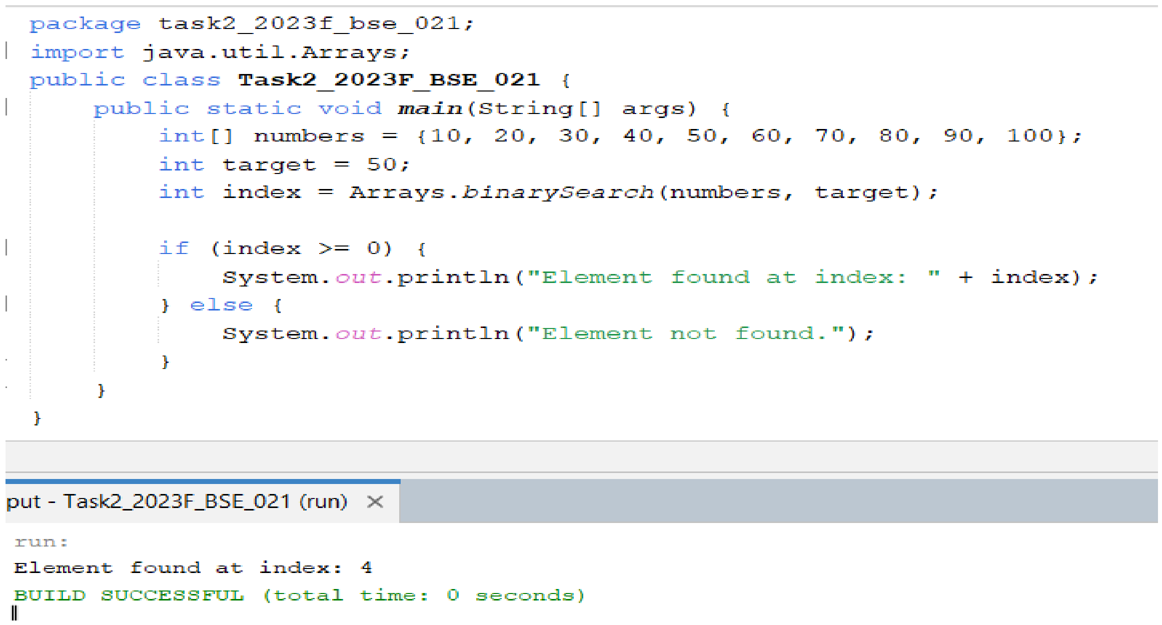
```java
package task1_2023f_bse_021;
public class Task1_2023F_BSE_021 {
    public static void main(String[] args) {
        int[] balances = {0, 5000, 12000, 8000, 300000, 450000, 600000, 700000, 100000, 1000000};

        System.out.println("Account No.\tLow Balance");
        for (int i = 0; i < balances.length; i++) {
            if (balances[i] < 10000) {
                System.out.println(i + "\t\tLow Balance");
            }
        }
    }
}
```

- Task1_2023F_BSE_021 (run)

```
run:
Account No.       Low Balance
0                 Low Balance
1                 Low Balance
3                 Low Balance
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Write a program to search in array using Array built-in class.



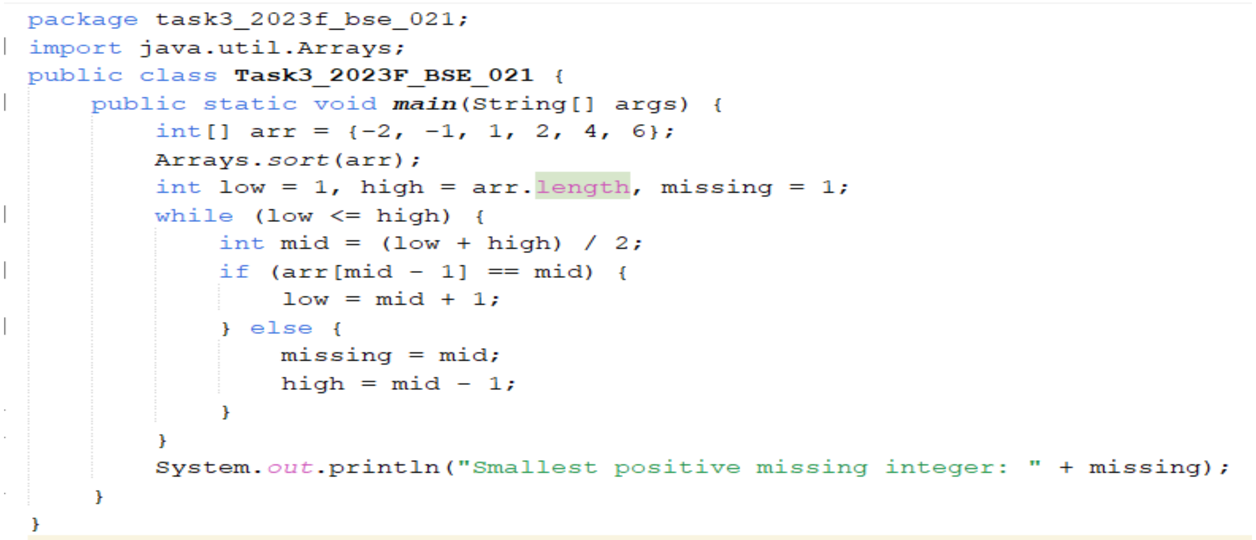
```java
package task2_2023f_bse_021;
import java.util.Arrays;
public class Task2_2023F_BSE_021 {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
        int target = 50;
        int index = Arrays.binarySearch(numbers, target);

        if (index >= 0) {
            System.out.println("Element found at index: " + index);
        } else {
            System.out.println("Element not found.");
        }
    }
}
```

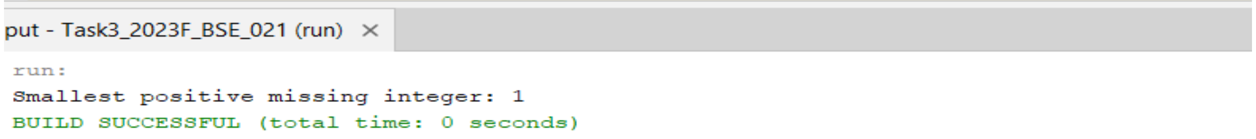
put - Task2_2023F_BSE_021 (run)

```
run:
Element found at index: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Given an unsorted array `arr` of integers, find the smallest positive integer that is **missing** from the array. You need to implement this using **binary search**. The array can contain both negative numbers and positive numbers, and you can assume that the array does not have duplicates.



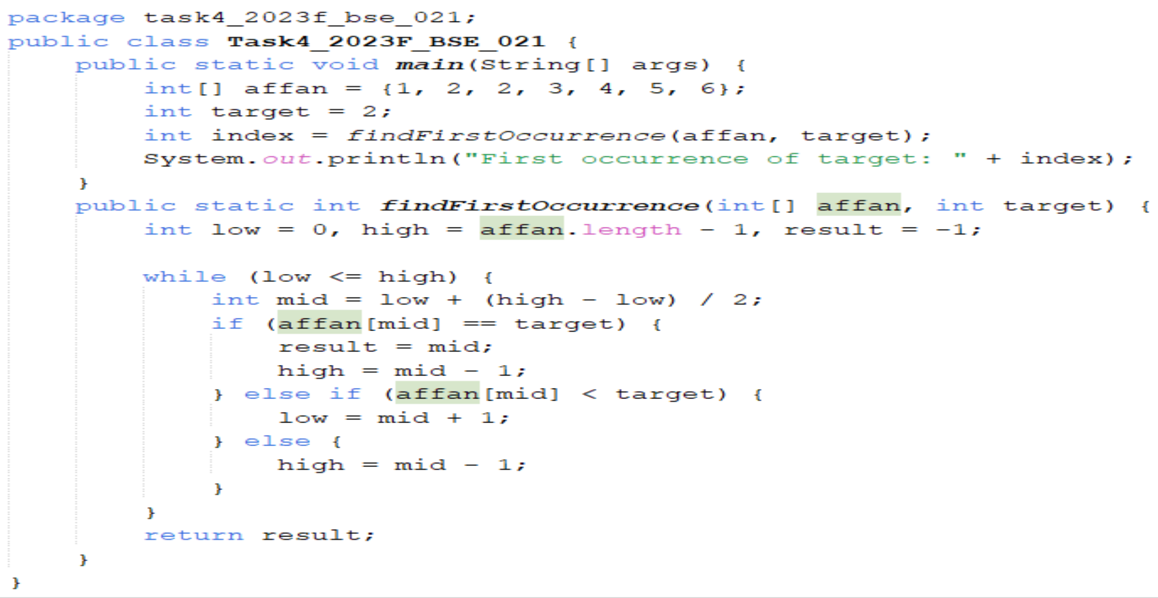
```
package task3_2023f_bse_021;
import java.util.Arrays;
public class Task3_2023F_BSE_021 {
    public static void main(String[] args) {
        int[] arr = {-2, -1, 1, 2, 4, 6};
        Arrays.sort(arr);
        int low = 1, high = arr.length, missing = 1;
        while (low <= high) {
            int mid = (low + high) / 2;
            if (arr[mid - 1] == mid) {
                low = mid + 1;
            } else {
                missing = mid;
                high = mid - 1;
            }
        }
        System.out.println("Smallest positive missing integer: " + missing);
    }
}
```



put - Task3_2023F_BSE_021 (run)

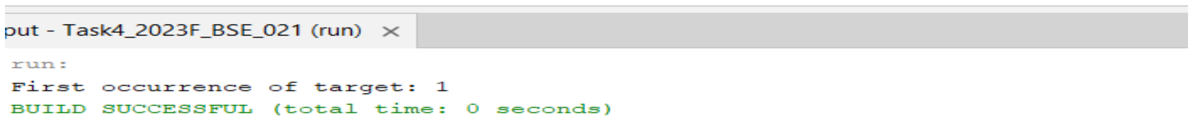
```
run:
Smallest positive missing integer: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. You are given a sorted array arr[] and a target element target. Your task is to find the **first occurrence** of the target in the array using binary search. If the target is not found, return -1. You are given a sorted array arr[] and a target element target. Your task is to find the **first occurrence** of the target in the array using binary search. If the target is not found, return -1.



```
package task4_2023f_bse_021;
public class Task4_2023F_BSE_021 {
    public static void main(String[] args) {
        int[] affan = {1, 2, 2, 3, 4, 5, 6};
        int target = 2;
        int index = findFirstOccurrence(affan, target);
        System.out.println("First occurrence of target: " + index);
    }
    public static int findFirstOccurrence(int[] affan, int target) {
        int low = 0, high = affan.length - 1, result = -1;

        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (affan[mid] == target) {
                result = mid;
                high = mid - 1;
            } else if (affan[mid] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return result;
    }
}
```

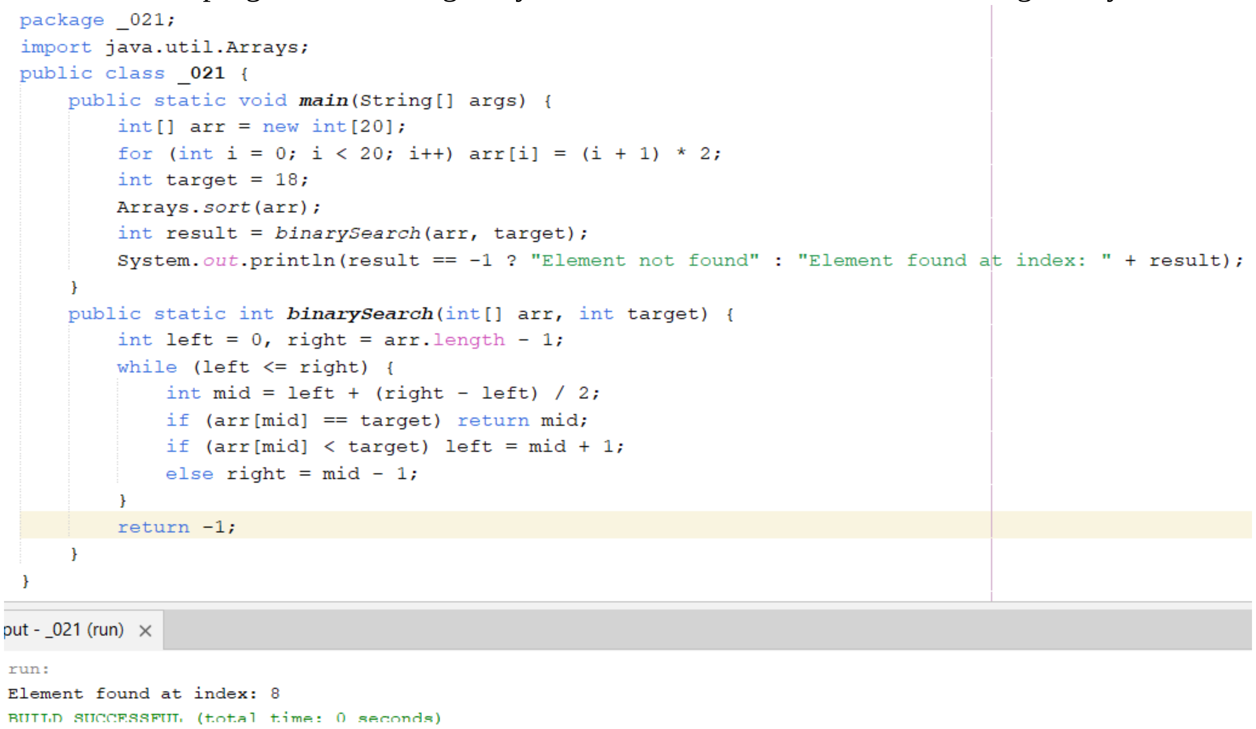


put - Task4_2023F_BSE_021 (run)

```
run:
First occurrence of target: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Home Task

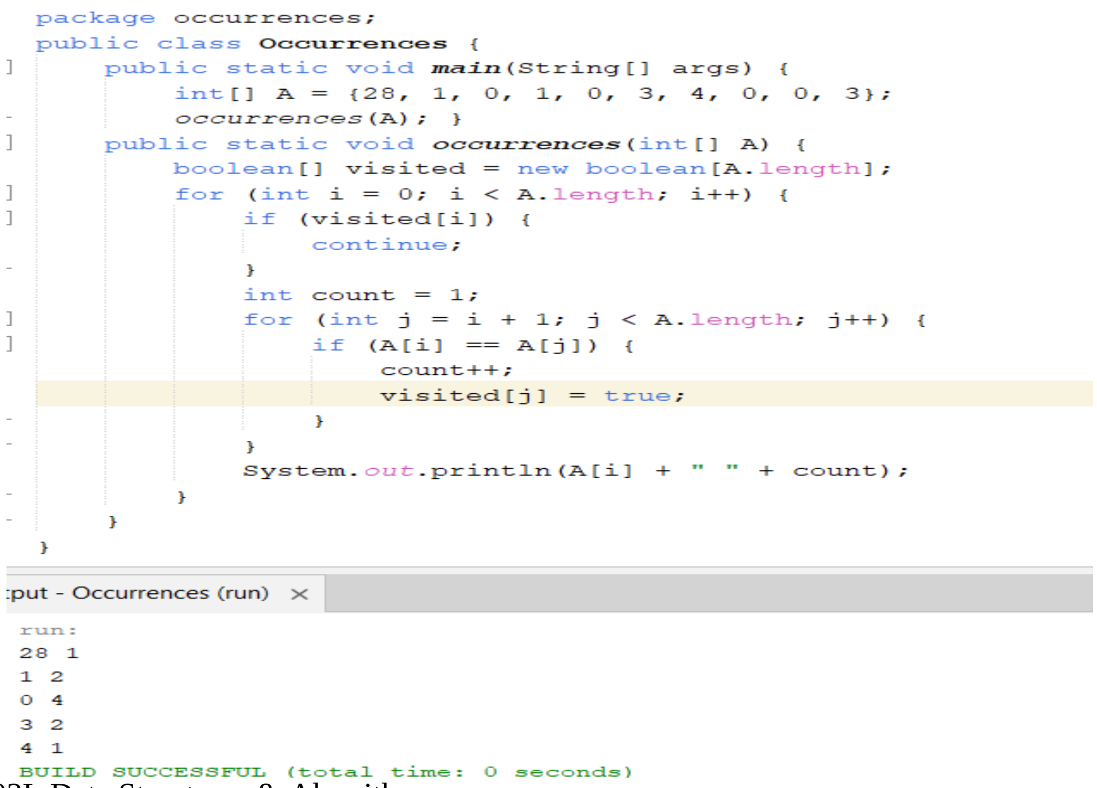
1. Write a program initializing array of size 20 and search an element using binary search.



```java
package _021;
import java.util.Arrays;
public class _021 {
    public static void main(String[] args) {
        int[] arr = new int[20];
        for (int i = 0; i < 20; i++) arr[i] = (i + 1) * 2;
        int target = 18;
        Arrays.sort(arr);
        int result = binarySearch(arr, target);
        System.out.println(result == -1 ? "Element not found" : "Element found at index: " + result);
    }
    public static int binarySearch(int[] arr, int target) {
        int left = 0, right = arr.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] == target) return mid;
            if (arr[mid] < target) left = mid + 1;
            else right = mid - 1;
        }
        return -1;
    }
}
```

```
run:
Element found at index: 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

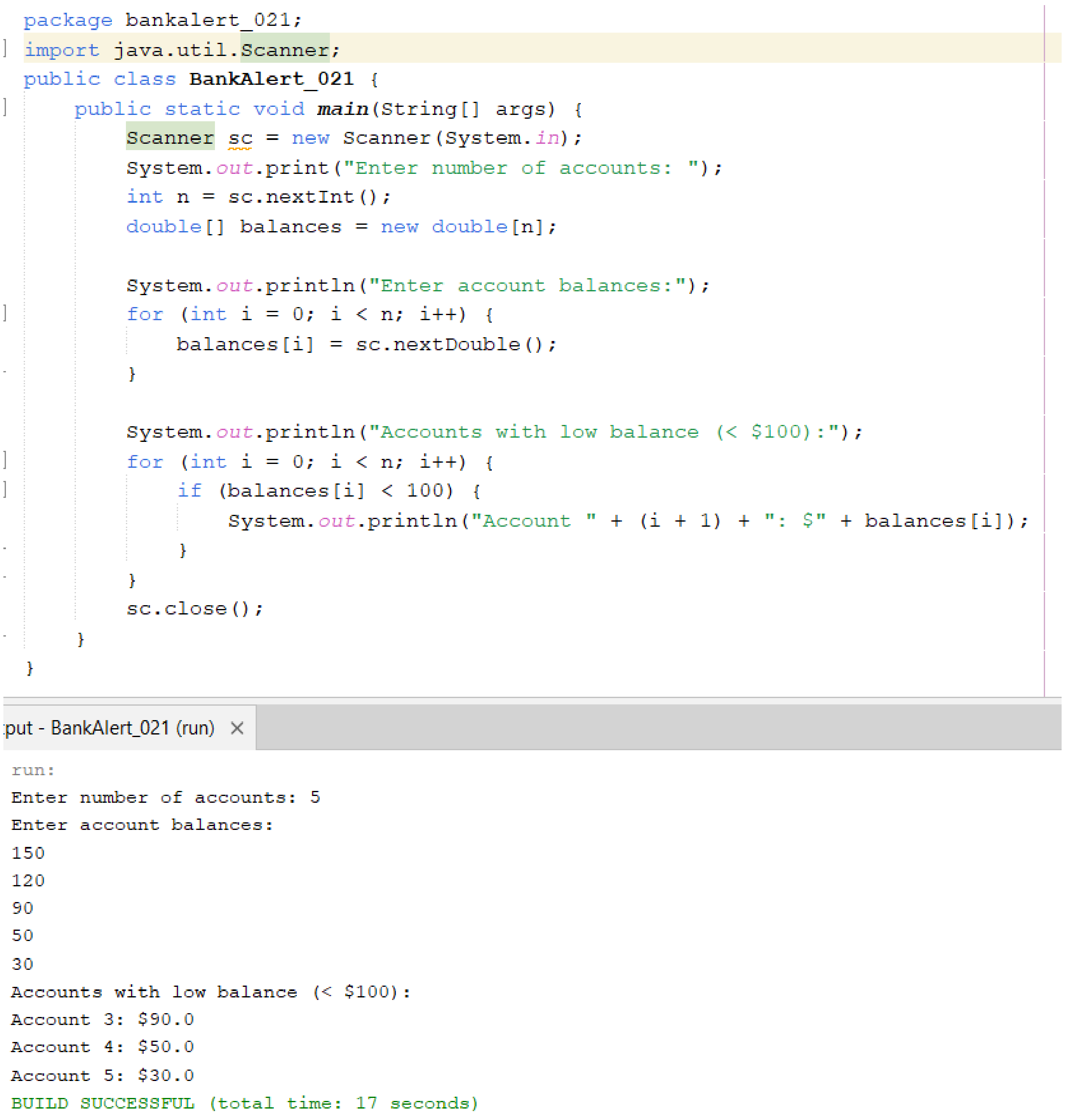
2. Write a function called occurrences that, given an array of numbers A, prints all the distinct values in A each followed by its number of occurrences. For example, if A = (28, 1, 0, 1, 0, 3, 4, 0, 0, 3), the function should output the following five lines (here separated by a semicolon) “28 1; 1 2; 0 4; 3 2; 4 1”.



```java
package occurrences;
public class Occurrences {
    public static void main(String[] args) {
        int[] A = {28, 1, 0, 1, 0, 3, 4, 0, 0, 3};
        occurrences(A); }
    public static void occurrences(int[] A) {
        boolean[] visited = new boolean[A.length];
        for (int i = 0; i < A.length; i++) {
            if (visited[i]) {
                continue;
            }
            int count = 1;
            for (int j = i + 1; j < A.length; j++) {
                if (A[i] == A[j]) {
                    count++;
                    visited[j] = true;
                }
            }
            System.out.println(A[i] + " " + count);
        }
    }
}
```

```
run:
28 1
1 2
0 4
3 2
4 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Assume a bank's system needs to identify accounts with critically low balances and alert the user. Test the function with various balance values to ensure it correctly identifies all accounts below the threshold.



```java
package bankalert_021;
import java.util.Scanner;
public class BankAlert_021 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of accounts: ");
        int n = sc.nextInt();
        double[] balances = new double[n];

        System.out.println("Enter account balances:");
        for (int i = 0; i < n; i++) {
            balances[i] = sc.nextDouble();
        }

        System.out.println("Accounts with low balance (< $100):");
        for (int i = 0; i < n; i++) {
            if (balances[i] < 100) {
                System.out.println("Account " + (i + 1) + ": $" + balances[i]);
            }
        }
        sc.close();
    }
}
```

put - BankAlert_021 (run)

```
run:
Enter number of accounts: 5
Enter account balances:
150
120
90
50
30
Accounts with low balance (< $100):
Account 3: $90.0
Account 4: $50.0
Account 5: $30.0
BUILD SUCCESSFUL (total time: 17 seconds)
```