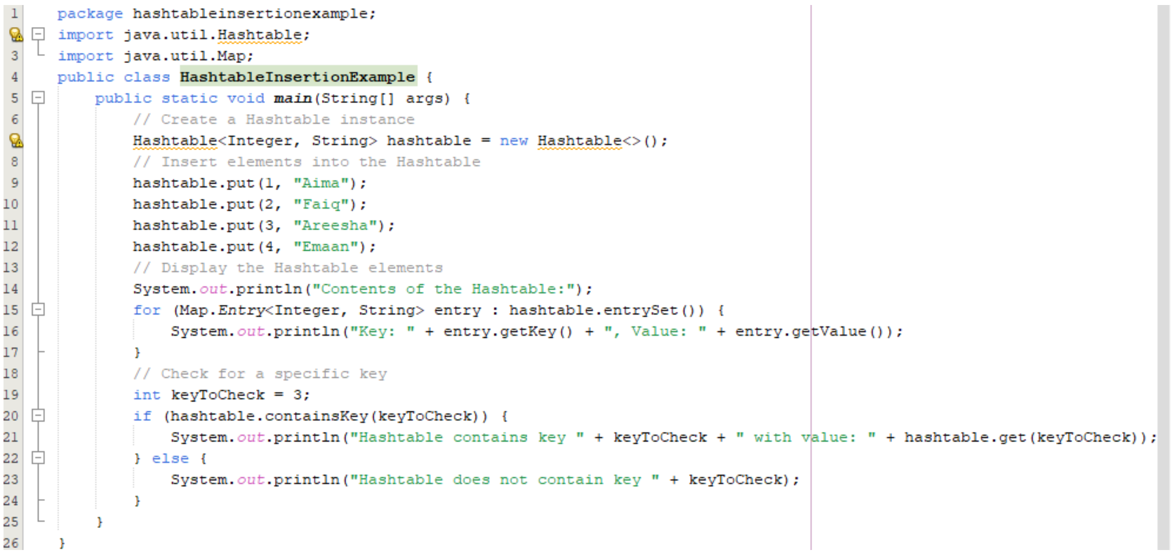
# LAB # 13

## Implementing Hashing techniques and using HashTable ADT and Collision Resolution Techniques

**OBJECTIVE:** Implementing Hashing techniques with collision resolution.

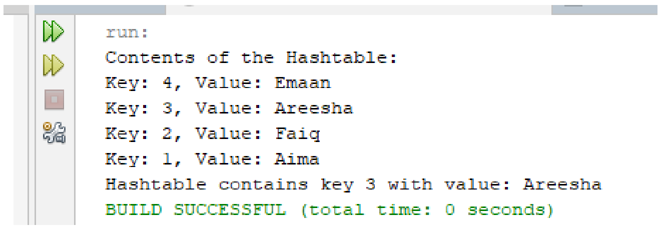
### Lab Task

1. Write a program which shows the insertion of the elements in a hashtable.
CODE



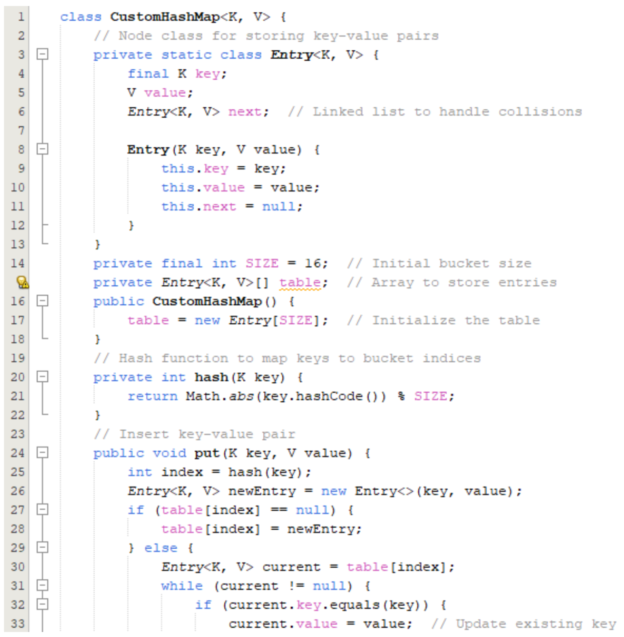
```java
package hashtableinsertionexample;
import java.util.Hashtable;
import java.util.Map;
public class HashtableInsertionExample {
    public static void main(String[] args) {
        // Create a Hashtable instance
        Hashtable<Integer, String> hashtable = new Hashtable<>();
        // Insert elements into the Hashtable
        hashtable.put(1, "Aima");
        hashtable.put(2, "Faiq");
        hashtable.put(3, "Areesha");
        hashtable.put(4, "Emaan");
        // Display the Hashtable elements
        System.out.println("Contents of the Hashtable:");
        for (Map.Entry<Integer, String> entry : hashtable.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());
        }
        // Check for a specific key
        int keyToCheck = 3;
        if (hashtable.containsKey(keyToCheck)) {
            System.out.println("Hashtable contains key " + keyToCheck + " with value: " + hashtable.get(keyToCheck));
        } else {
            System.out.println("Hashtable does not contain key " + keyToCheck);
        }
    }
}
```

OUTPUT

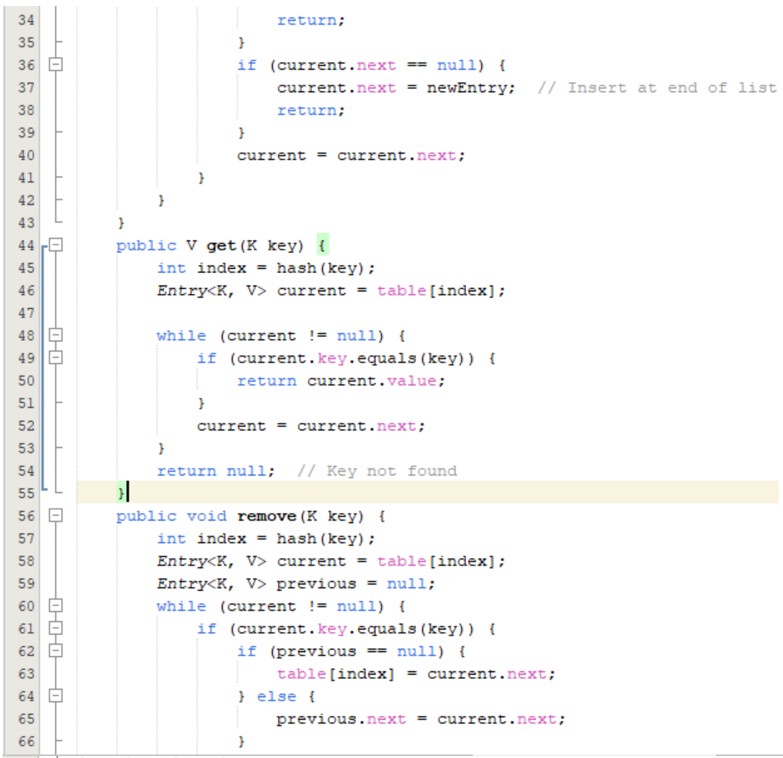


```
run:
Contents of the Hashtable:
Key: 4, Value: Emaan
Key: 3, Value: Areesha
Key: 2, Value: Faiq
Key: 1, Value: Aima
Hashtable contains key 3 with value: Areesha
BUILD SUCCESSFUL (total time: 0 seconds)
```

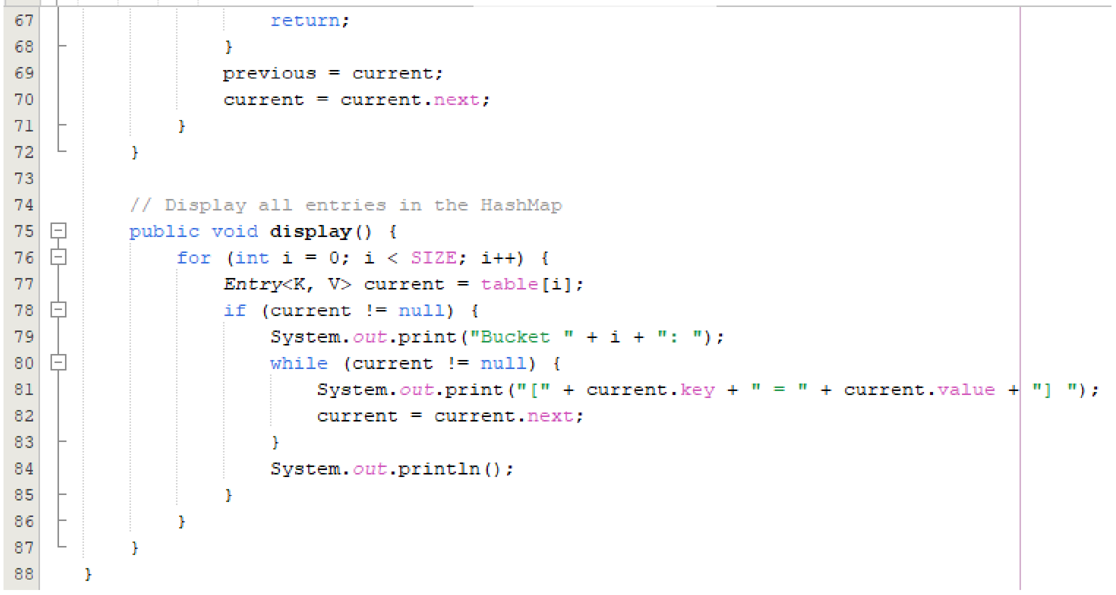
2. Design a HashMap without using any built-in hash table libraries.
CODE



```java
class CustomHashMap<K, V> {
    // Node class for storing key-value pairs
    private static class Entry<K, V> {
        final K key;
        V value;
        Entry<K, V> next;  // Linked list to handle collisions

        Entry(K key, V value) {
            this.key = key;
            this.value = value;
            this.next = null;
        }
    }
    private final int SIZE = 16;  // Initial bucket size
    private Entry<K, V>[] table;  // Array to store entries
    public CustomHashMap() {
        table = new Entry[SIZE];  // Initialize the table
    }
    // Hash function to map keys to bucket indices
    private int hash(K key) {
        return Math.abs(key.hashCode()) % SIZE;
    }
    // Insert key-value pair
    public void put(K key, V value) {
        int index = hash(key);
        Entry<K, V> newEntry = new Entry<>(key, value);
        if (table[index] == null) {
            table[index] = newEntry;
        } else {
            Entry<K, V> current = table[index];
            while (current != null) {
                if (current.key.equals(key)) {
                    current.value = value;  // Update existing key
```

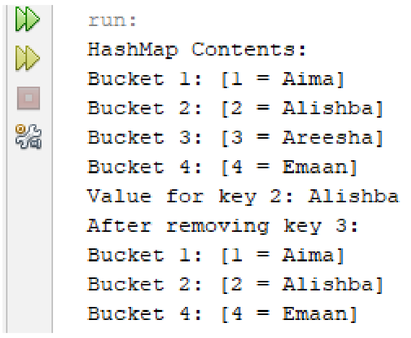


```
                    return;
                }
                if (current.next == null) {
                    current.next = newEntry;  // Insert at end of list
                    return;
                }
                current = current.next;
            }
        }
    }
    public V get(K key) {
        int index = hash(key);
        Entry<K, V> current = table[index];

        while (current != null) {
            if (current.key.equals(key)) {
                return current.value;
            }
            current = current.next;
        }
        return null;  // Key not found
    }
    public void remove(K key) {
        int index = hash(key);
        Entry<K, V> current = table[index];
        Entry<K, V> previous = null;
        while (current != null) {
            if (current.key.equals(key)) {
                if (previous == null) {
                    table[index] = current.next;
                } else {
                    previous.next = current.next;
                }
```



```
                return;
            }
            previous = current;
            current = current.next;
        }
    }

    // Display all entries in the HashMap
    public void display() {
        for (int i = 0; i < SIZE; i++) {
            Entry<K, V> current = table[i];
            if (current != null) {
                System.out.print("Bucket " + i + ": ");
                while (current != null) {
                    System.out.print("[" + current.key + " = " + current.value + "] ");
                    current = current.next;
                }
                System.out.println();
            }
        }
    }
}
```

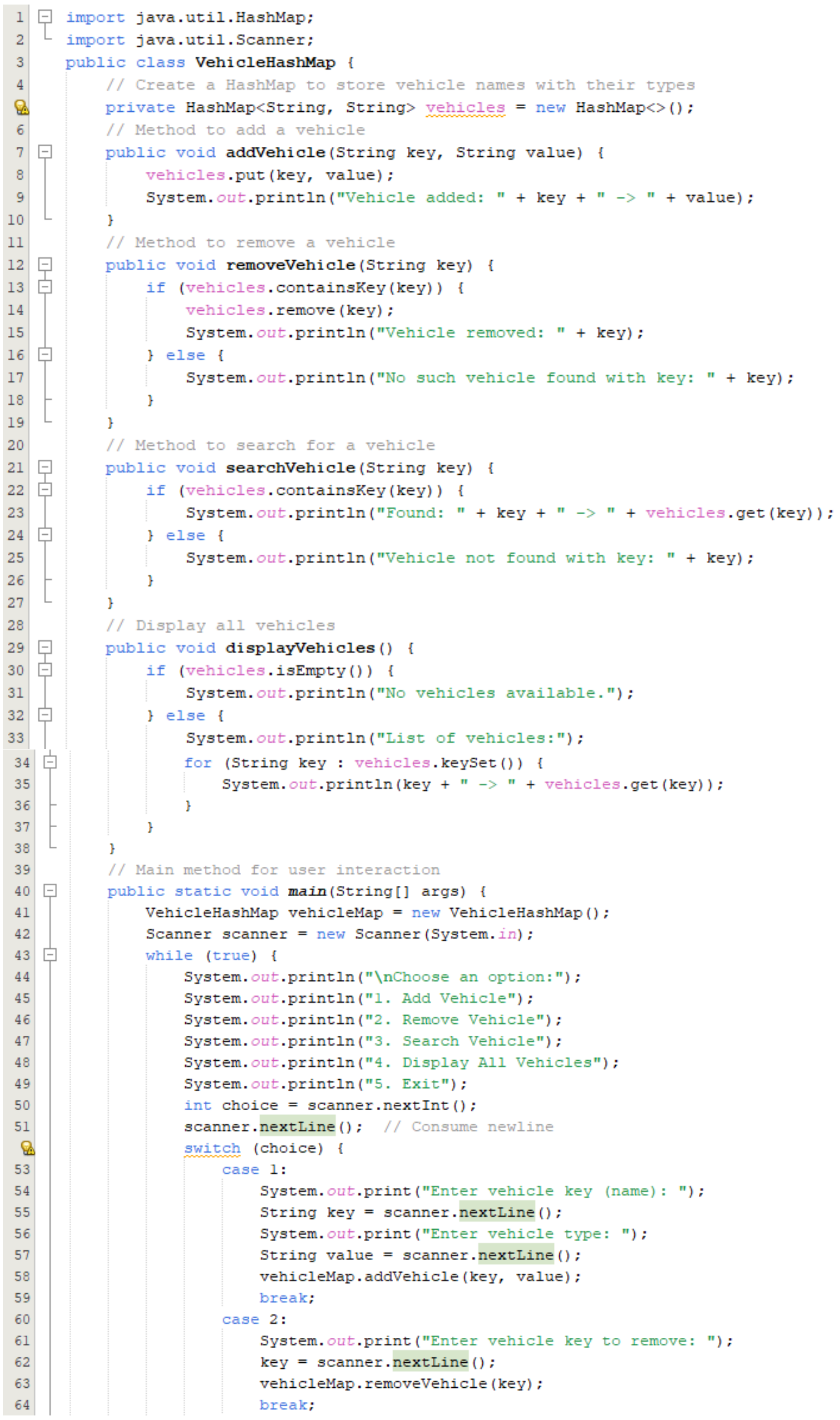
OUTPUT



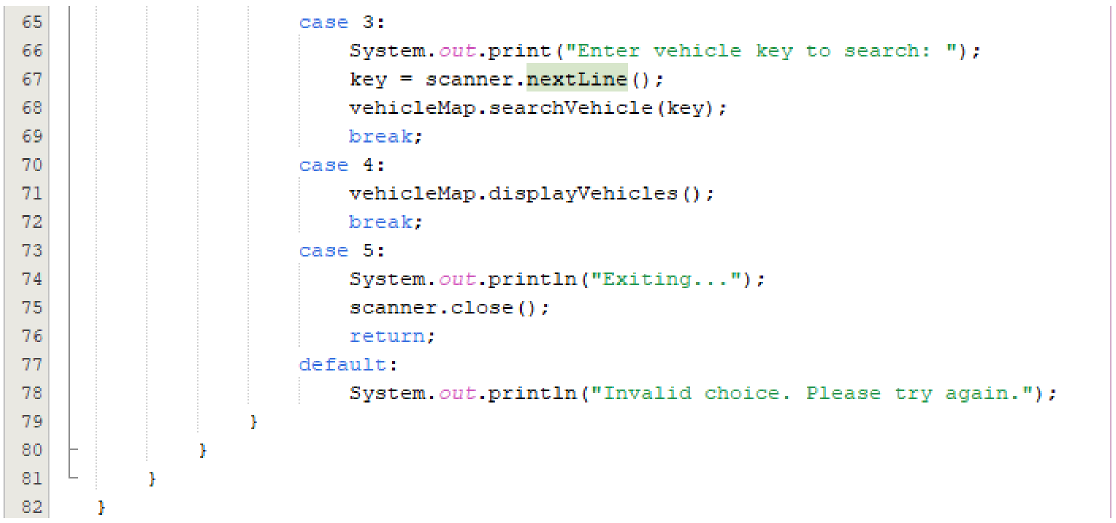
```
run:
HashMap Contents:
Bucket 1: [1 = Aima]
Bucket 2: [2 = Alishba]
Bucket 3: [3 = Areesha]
Bucket 4: [4 = Emaan]
Value for key 2: Alishba
After removing key 3:
Bucket 1: [1 = Aima]
Bucket 2: [2 = Alishba]
Bucket 4: [4 = Emaan]
```

3. Create a HashMap object called Vehicles which store String keys and String values and perform add( ), remove( ) and search( ) methods on it.

CODE

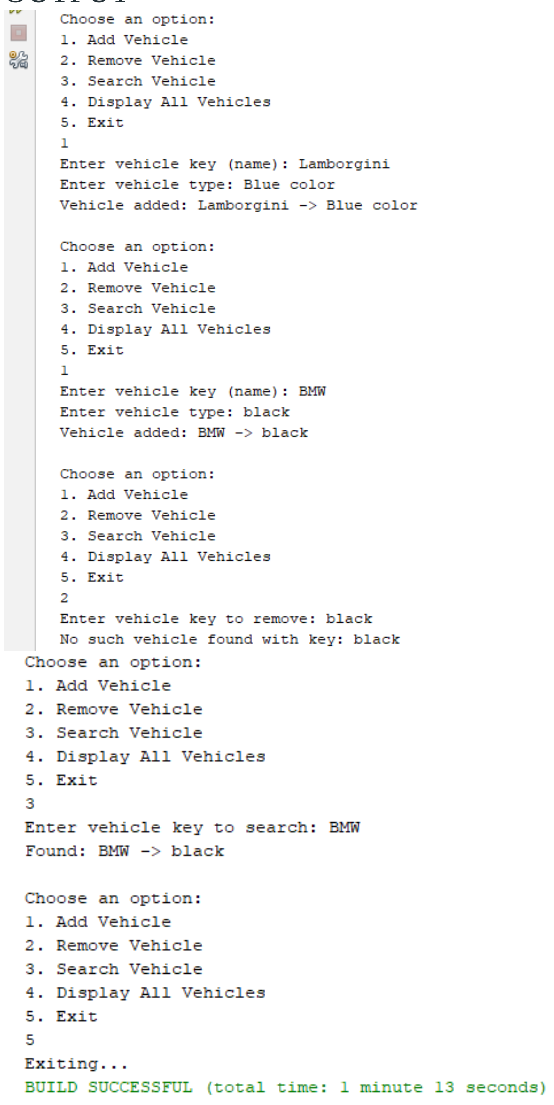


```java
import java.util.HashMap;
import java.util.Scanner;
public class VehicleHashMap {
    // Create a HashMap to store vehicle names with their types
    private HashMap<String, String> vehicles = new HashMap<>();
    // Method to add a vehicle
    public void addVehicle(String key, String value) {
        vehicles.put(key, value);
        System.out.println("Vehicle added: " + key + " -> " + value);
    }
    // Method to remove a vehicle
    public void removeVehicle(String key) {
        if (vehicles.containsKey(key)) {
            vehicles.remove(key);
            System.out.println("Vehicle removed: " + key);
        } else {
            System.out.println("No such vehicle found with key: " + key);
        }
    }
    // Method to search for a vehicle
    public void searchVehicle(String key) {
        if (vehicles.containsKey(key)) {
            System.out.println("Found: " + key + " -> " + vehicles.get(key));
        } else {
            System.out.println("Vehicle not found with key: " + key);
        }
    }
    // Display all vehicles
    public void displayVehicles() {
        if (vehicles.isEmpty()) {
            System.out.println("No vehicles available.");
        } else {
            System.out.println("List of vehicles:");
            for (String key : vehicles.keySet()) {
                System.out.println(key + " -> " + vehicles.get(key));
            }
        }
    }
    // Main method for user interaction
    public static void main(String[] args) {
        VehicleHashMap vehicleMap = new VehicleHashMap();
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nChoose an option:");
            System.out.println("1. Add Vehicle");
            System.out.println("2. Remove Vehicle");
            System.out.println("3. Search Vehicle");
            System.out.println("4. Display All Vehicles");
            System.out.println("5. Exit");
            int choice = scanner.nextInt();
            scanner.nextLine();  // Consume newline
            switch (choice) {
                case 1:
                    System.out.print("Enter vehicle key (name): ");
                    String key = scanner.nextLine();
                    System.out.print("Enter vehicle type: ");
                    String value = scanner.nextLine();
                    vehicleMap.addVehicle(key, value);
                    break;
                case 2:
                    System.out.print("Enter vehicle key to remove: ");
                    key = scanner.nextLine();
                    vehicleMap.removeVehicle(key);
                    break;
```



```java
                    case 3:
                        System.out.print("Enter vehicle key to search: ");
                        key = scanner.nextLine();
                        vehicleMap.searchVehicle(key);
                        break;
                    case 4:
                        vehicleMap.displayVehicles();
                        break;
                    case 5:
                        System.out.println("Exiting...");
                        scanner.close();
                        return;
                    default:
                        System.out.println("Invalid choice. Please try again.");
                }
            }
        }
    }
```

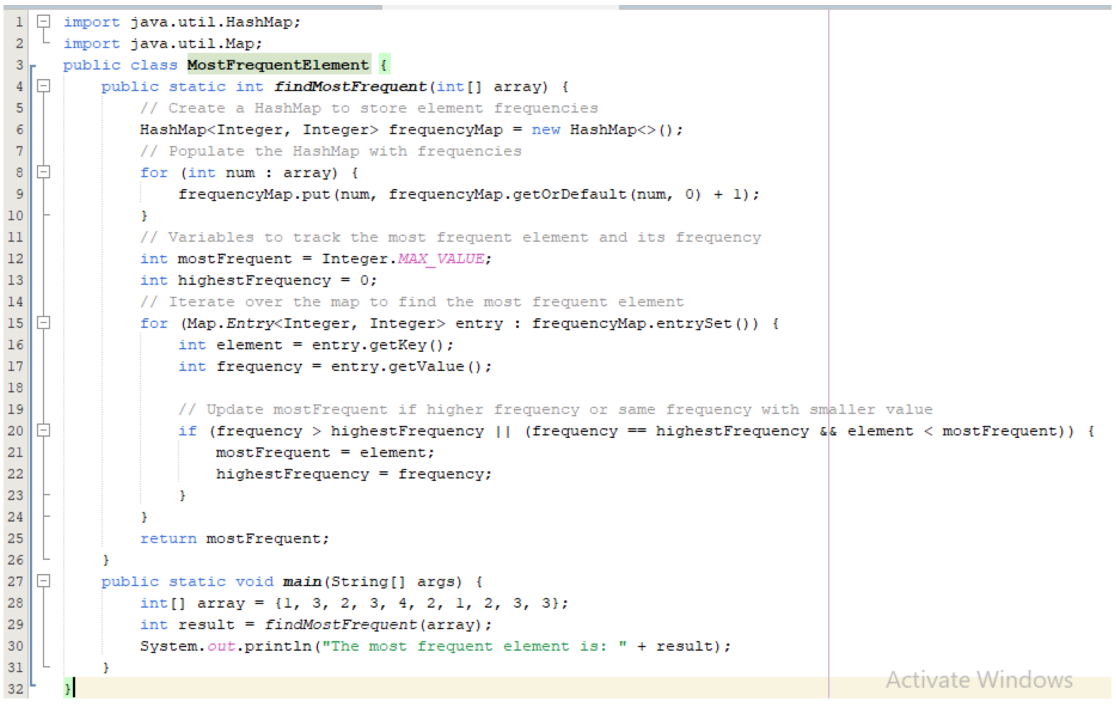
OUTPUT



```
Choose an option:
1. Add Vehicle
2. Remove Vehicle
3. Search Vehicle
4. Display All Vehicles
5. Exit
1
Enter vehicle key (name): Lamborgini
Enter vehicle type: Blue color
Vehicle added: Lamborgini -> Blue color

Choose an option:
1. Add Vehicle
2. Remove Vehicle
3. Search Vehicle
4. Display All Vehicles
5. Exit
1
Enter vehicle key (name): BMW
Enter vehicle type: black
Vehicle added: BMW -> black

Choose an option:
1. Add Vehicle
2. Remove Vehicle
3. Search Vehicle
4. Display All Vehicles
5. Exit
2
Enter vehicle key to remove: black
No such vehicle found with key: black
Choose an option:
1. Add Vehicle
2. Remove Vehicle
3. Search Vehicle
4. Display All Vehicles
5. Exit
3
Enter vehicle key to search: BMW
Found: BMW -> black

Choose an option:
1. Add Vehicle
2. Remove Vehicle
3. Search Vehicle
4. Display All Vehicles
5. Exit
5
Exiting...
BUILD SUCCESSFUL (total time: 1 minute 13 seconds)
```

4.Given an array of integers, write a Java program to find the most frequent element in the array. If there are multiple elements with the same frequency, return the one with the smallest value. Use a HashMap to count the frequency of each element and identify the most frequent one.

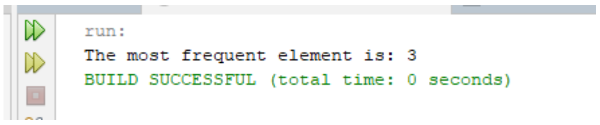
CODE



```java
import java.util.HashMap;
import java.util.Map;
public class MostFrequentElement {
    public static int findMostFrequent(int[] array) {
        // Create a HashMap to store element frequencies
        HashMap<Integer, Integer> frequencyMap = new HashMap<>();
        // Populate the HashMap with frequencies
        for (int num : array) {
            frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);
        }
        // Variables to track the most frequent element and its frequency
        int mostFrequent = Integer.MAX_VALUE;
        int highestFrequency = 0;
        // Iterate over the map to find the most frequent element
        for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) {
            int element = entry.getKey();
            int frequency = entry.getValue();

            // Update mostFrequent if higher frequency or same frequency with smaller value
            if (frequency > highestFrequency || (frequency == highestFrequency && element < mostFrequent)) {
                mostFrequent = element;
                highestFrequency = frequency;
            }
        }
        return mostFrequent;
    }
    public static void main(String[] args) {
        int[] array = {1, 3, 2, 3, 4, 2, 1, 2, 3, 3};
        int result = findMostFrequent(array);
        System.out.println("The most frequent element is: " + result);
    }
}
```

Activate Windows

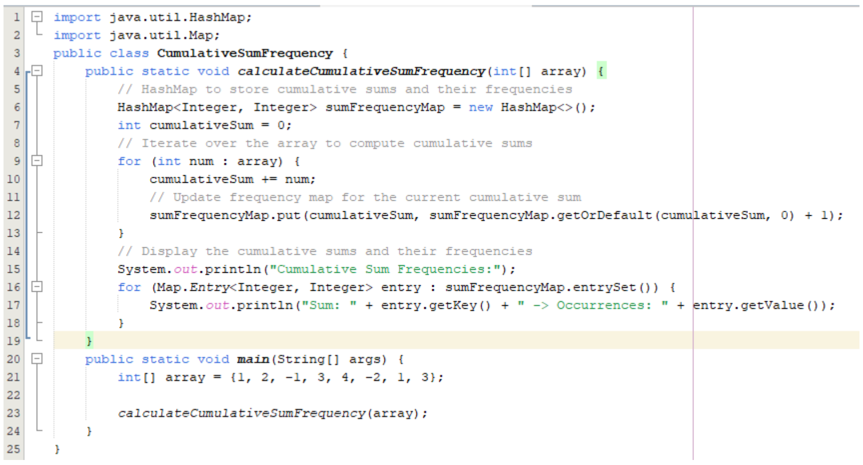
OUTPUT



```
run:
The most frequent element is: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

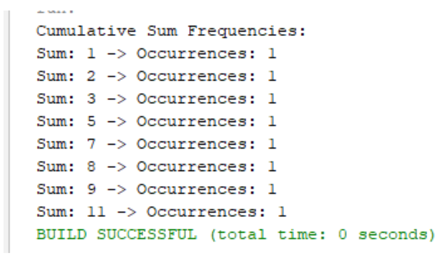
5.Use a HashMap to store the cumulative sum of elements and count how often a sum has occurred.

CODE



```java
import java.util.HashMap;
import java.util.Map;
public class CumulativeSumFrequency {
    public static void calculateCumulativeSumFrequency(int[] array) {
        // HashMap to store cumulative sums and their frequencies
        HashMap<Integer, Integer> sumFrequencyMap = new HashMap<>();
        int cumulativeSum = 0;
        // Iterate over the array to compute cumulative sums
        for (int num : array) {
            cumulativeSum += num;
            // Update frequency map for the current cumulative sum
            sumFrequencyMap.put(cumulativeSum, sumFrequencyMap.getOrDefault(cumulativeSum, 0) + 1);
        }
        // Display the cumulative sums and their frequencies
        System.out.println("Cumulative Sum Frequencies:");
        for (Map.Entry<Integer, Integer> entry : sumFrequencyMap.entrySet()) {
            System.out.println("Sum: " + entry.getKey() + " -> Occurrences: " + entry.getValue());
        }
    }
    public static void main(String[] args) {
        int[] array = {1, 2, -1, 3, 4, -2, 1, 3};

        calculateCumulativeSumFrequency(array);
    }
}
```

OUTPUT

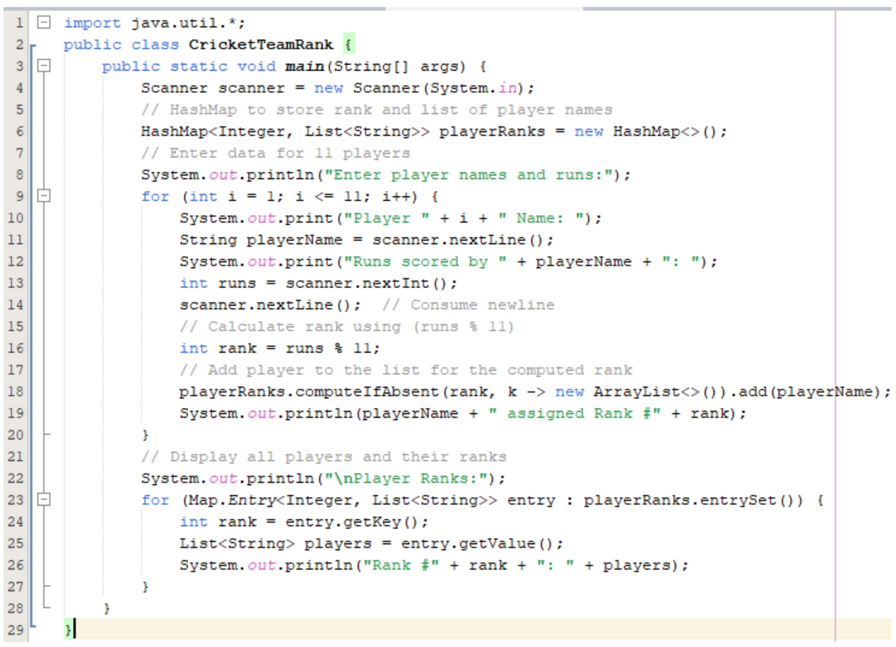


```
Cumulative Sum Frequencies:
Sum: 1 -> Occurrences: 1
Sum: 2 -> Occurrences: 1
Sum: 3 -> Occurrences: 1
Sum: 5 -> Occurrences: 1
Sum: 7 -> Occurrences: 1
Sum: 8 -> Occurrences: 1
Sum: 9 -> Occurrences: 1
Sum: 11 -> Occurrences: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Home Task

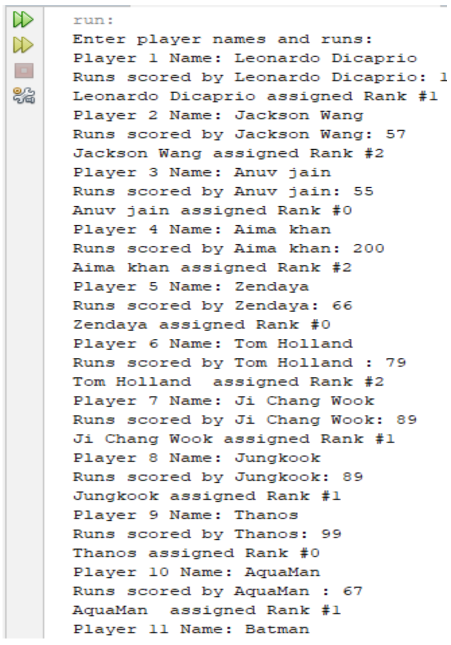
1. Enter data of a cricket team 11 players which is supposed to be a hash table value and insert runs of each player as a data, find out key treat's Rank# of a player. For example: Runs are 30 mod by 11 which is index no 8; rank#8 is a rank of a team member. (Use HashTable ADT class)

CODE

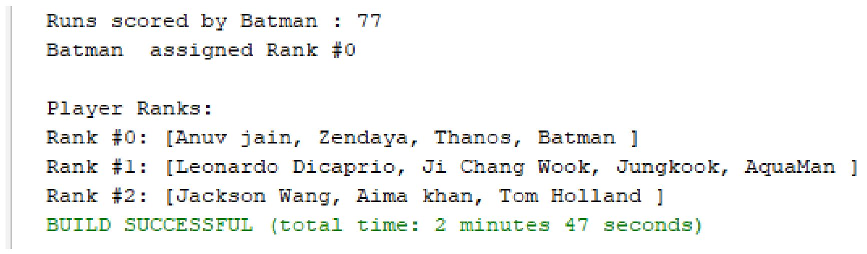


```java
import java.util.*;
public class CricketTeamRank {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // HashMap to store rank and list of player names
        HashMap<Integer, List<String>> playerRanks = new HashMap<>();
        // Enter data for 11 players
        System.out.println("Enter player names and runs:");
        for (int i = 1; i <= 11; i++) {
            System.out.print("Player " + i + " Name: ");
            String playerName = scanner.nextLine();
            System.out.print("Runs scored by " + playerName + ": ");
            int runs = scanner.nextInt();
            scanner.nextLine();  // Consume newline
            // Calculate rank using (runs % 11)
            int rank = runs % 11;
            // Add player to the list for the computed rank
            playerRanks.computeIfAbsent(rank, k -> new ArrayList<>()).add(playerName);
            System.out.println(playerName + " assigned Rank #" + rank);
        }
        // Display all players and their ranks
        System.out.println("\nPlayer Ranks:");
        for (Map.Entry<Integer, List<String>> entry : playerRanks.entrySet()) {
            int rank = entry.getKey();
            List<String> players = entry.getValue();
            System.out.println("Rank #" + rank + ": " + players);
        }
    }
}
```

OUTPUT



```
run:
Enter player names and runs:
Player 1 Name: Leonardo Dicaprio
Runs scored by Leonardo Dicaprio: 1
Leonardo Dicaprio assigned Rank #1
Player 2 Name: Jackson Wang
Runs scored by Jackson Wang: 57
Jackson Wang assigned Rank #2
Player 3 Name: Anuv jain
Runs scored by Anuv jain: 55
Anuv jain assigned Rank #0
Player 4 Name: Aima khan
Runs scored by Aima khan: 200
Aima khan assigned Rank #2
Player 5 Name: Zendaya
Runs scored by Zendaya: 66
Zendaya assigned Rank #0
Player 6 Name: Tom Holland
Runs scored by Tom Holland : 79
Tom Holland  assigned Rank #2
Player 7 Name: Ji Chang Wook
Runs scored by Ji Chang Wook: 89
Ji Chang Wook assigned Rank #1
Player 8 Name: Jungkook
Runs scored by Jungkook: 89
Jungkook assigned Rank #1
Player 9 Name: Thanos
Runs scored by Thanos: 99
Thanos assigned Rank #0
Player 10 Name: AquaMan
Runs scored by AquaMan : 67
AquaMan  assigned Rank #1
Player 11 Name: Batman
```



```
Runs scored by Batman : 77
Batman  assigned Rank #0

Player Ranks:
Rank #0: [Anuv jain, Zendaya, Thanos, Batman ]
Rank #1: [Leonardo Dicaprio, Ji Chang Wook, Jungkook, AquaMan ]
Rank #2: [Jackson Wang, Aima khan, Tom Holland ]
BUILD SUCCESSFUL (total time: 2 minutes 47 seconds)
```

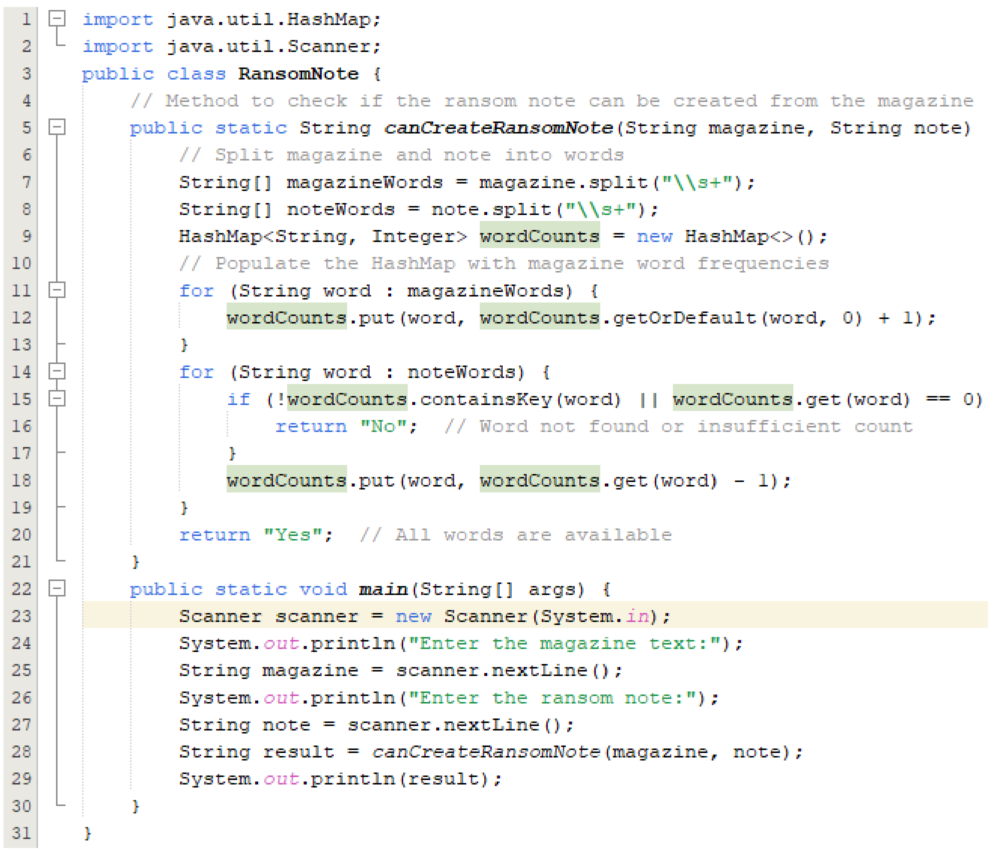
2. Harold is a kidnapper who wrote a ransom note, but now he is worried it will be traced back to him through his handwriting. He found a magazine and wants to know if he can cut out whole words from it and use them to create an untraceable replica of his ransom note. The words in his note are case-sensitive and he must use only whole words available in the magazine. He cannot use substrings or concatenation to create the words he needs.Given the words in the magazine and the words in the ransom note, print **Yes** if he can replicate his ransom note exactly using whole words from the magazine; otherwise, print **No**.

Example:

magazine = "attack at dawn" note = "Attack at dawn"

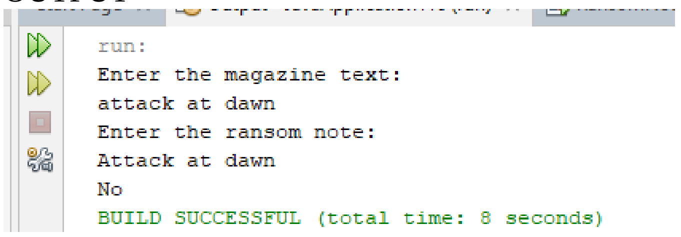
The magazine has all the right words, but there is a case mismatch. The answer is **No**.

CODE



```java
import java.util.HashMap;
import java.util.Scanner;
public class RansomNote {
    // Method to check if the ransom note can be created from the magazine
    public static String canCreateRansomNote(String magazine, String note)
        // Split magazine and note into words
        String[] magazineWords = magazine.split("\\s+");
        String[] noteWords = note.split("\\s+");
        HashMap<String, Integer> wordCounts = new HashMap<>();
        // Populate the HashMap with magazine word frequencies
        for (String word : magazineWords) {
            wordCounts.put(word, wordCounts.getOrDefault(word, 0) + 1);
        }
        for (String word : noteWords) {
            if (!wordCounts.containsKey(word) || wordCounts.get(word) == 0)
                return "No";  // Word not found or insufficient count
            }
            wordCounts.put(word, wordCounts.get(word) - 1);
        }
        return "Yes";  // All words are available
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the magazine text:");
        String magazine = scanner.nextLine();
        System.out.println("Enter the ransom note:");
        String note = scanner.nextLine();
        String result = canCreateRansomNote(magazine, note);
        System.out.println(result);
    }
}
```

OUTPUT



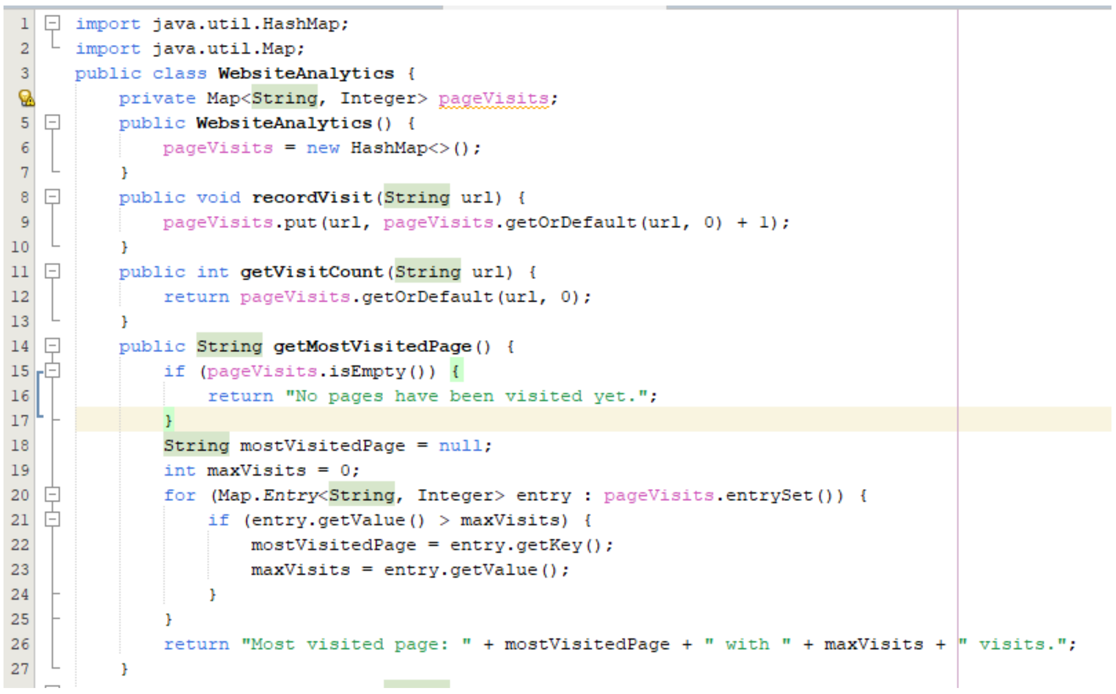
```
run:
Enter the magazine text:
attack at dawn
Enter the ransom note:
Attack at dawn
No
BUILD SUCCESSFUL (total time: 8 seconds)
```

**3. Scenario:**
You are building an analytics tool for a website. The tool needs to count the number of visitors to each page and store the data. Write a Java program that uses a HashMap to track the number of visits to different pages of a website. Each time a visitor accesses a page, the tool should increment the visit count for that page.

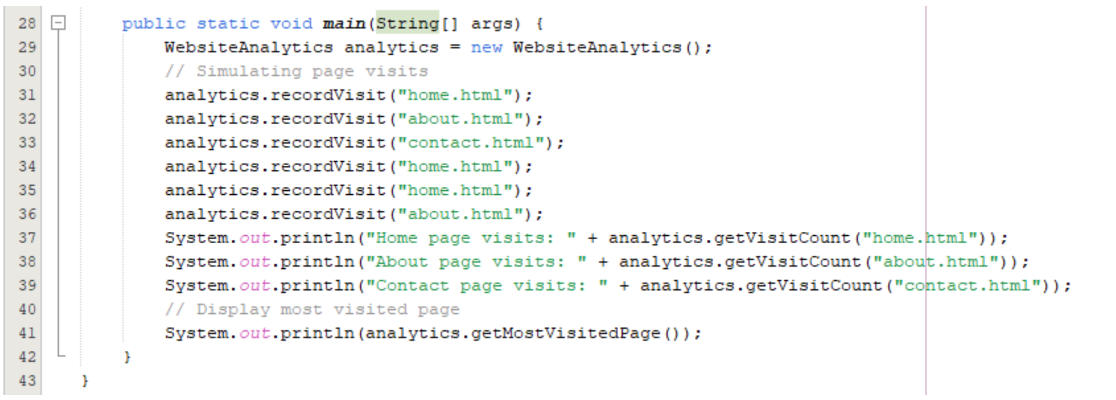
**Requirements:**

a) The key should be the page URL (as a string), and the value should be the number of visits (as an integer).
b) Provide a method to retrieve the visit count for a given page.
c) Provide a method to get the page with the highest visit count.

CODE

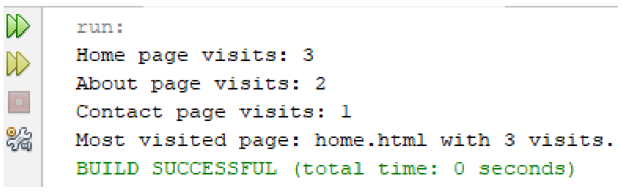


```java
import java.util.HashMap;
import java.util.Map;
public class WebsiteAnalytics {
    private Map<String, Integer> pageVisits;
    public WebsiteAnalytics() {
        pageVisits = new HashMap<>();
    }
    public void recordVisit(String url) {
        pageVisits.put(url, pageVisits.getOrDefault(url, 0) + 1);
    }
    public int getVisitCount(String url) {
        return pageVisits.getOrDefault(url, 0);
    }
    public String getMostVisitedPage() {
        if (pageVisits.isEmpty()) {
            return "No pages have been visited yet.";
        }
        String mostVisitedPage = null;
        int maxVisits = 0;
        for (Map.Entry<String, Integer> entry : pageVisits.entrySet()) {
            if (entry.getValue() > maxVisits) {
                mostVisitedPage = entry.getKey();
                maxVisits = entry.getValue();
            }
        }
        return "Most visited page: " + mostVisitedPage + " with " + maxVisits + " visits.";
    }
```



```java
    public static void main(String[] args) {
        WebsiteAnalytics analytics = new WebsiteAnalytics();
        // Simulating page visits
        analytics.recordVisit("home.html");
        analytics.recordVisit("about.html");
        analytics.recordVisit("contact.html");
        analytics.recordVisit("home.html");
        analytics.recordVisit("home.html");
        analytics.recordVisit("about.html");
        System.out.println("Home page visits: " + analytics.getVisitCount("home.html"));
        System.out.println("About page visits: " + analytics.getVisitCount("about.html"));
        System.out.println("Contact page visits: " + analytics.getVisitCount("contact.html"));
        // Display most visited page
        System.out.println(analytics.getMostVisitedPage());
    }
}
```

OUTPUT



```
run:
Home page visits: 3
About page visits: 2
Contact page visits: 1
Most visited page: home.html with 3 visits.
BUILD SUCCESSFUL (total time: 0 seconds)
```

**4.Scenario:**

You are developing a system to track student grades. The system should store the grades for each student in a course and calculate the average grade. Use a HashMap where the key is the student name (string) and the value is their grade (integer). The system should be able to add new students, update existing students' grades, and calculate the class average.

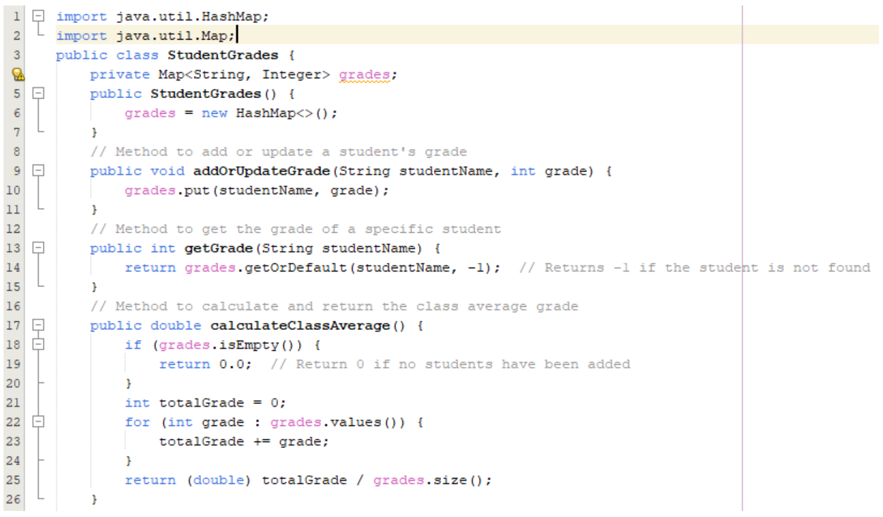
**Requirements:**

a) Implement methods to:

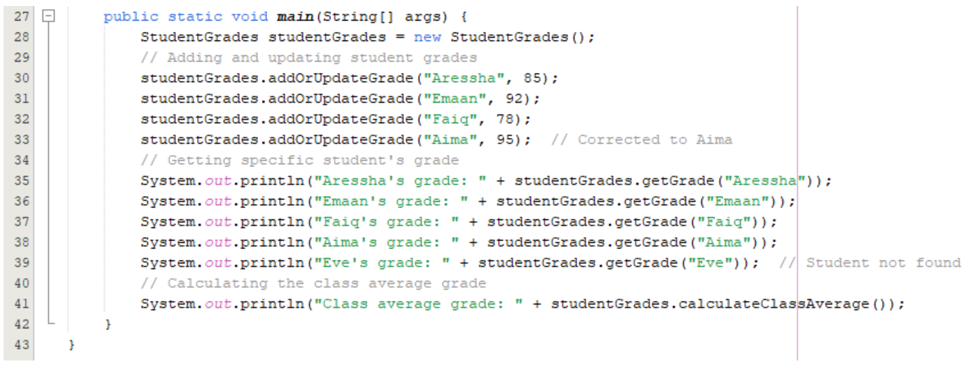
- Add or update student grades.
- Get the grade of a specific student.
- Calculate and return the class average grade.

b) Handle edge cases, such as when no students have been added yet.

CODE

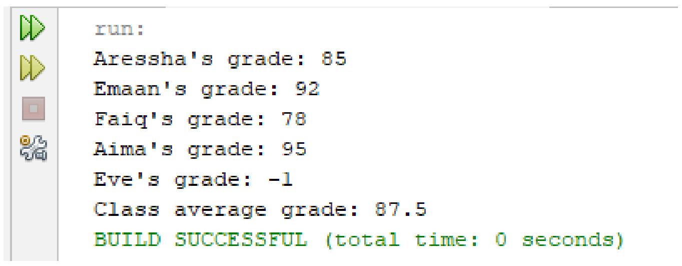


```java
import java.util.HashMap;
import java.util.Map;
public class StudentGrades {
    private Map<String, Integer> grades;
    public StudentGrades() {
        grades = new HashMap<>();
    }
    // Method to add or update a student's grade
    public void addOrUpdateGrade(String studentName, int grade) {
        grades.put(studentName, grade);
    }
    // Method to get the grade of a specific student
    public int getGrade(String studentName) {
        return grades.getOrDefault(studentName, -1);  // Returns -1 if the student is not found
    }
    // Method to calculate and return the class average grade
    public double calculateClassAverage() {
        if (grades.isEmpty()) {
            return 0.0;  // Return 0 if no students have been added
        }
        int totalGrade = 0;
        for (int grade : grades.values()) {
            totalGrade += grade;
        }
        return (double) totalGrade / grades.size();
    }
```



```java
    public static void main(String[] args) {
        StudentGrades studentGrades = new StudentGrades();
        // Adding and updating student grades
        studentGrades.addOrUpdateGrade("Aressha", 85);
        studentGrades.addOrUpdateGrade("Emaan", 92);
        studentGrades.addOrUpdateGrade("Faiq", 78);
        studentGrades.addOrUpdateGrade("Aima", 95);  // Corrected to Aima
        // Getting specific student's grade
        System.out.println("Aressha's grade: " + studentGrades.getGrade("Aressha"));
        System.out.println("Emaan's grade: " + studentGrades.getGrade("Emaan"));
        System.out.println("Faiq's grade: " + studentGrades.getGrade("Faiq"));
        System.out.println("Aima's grade: " + studentGrades.getGrade("Aima"));
        System.out.println("Eve's grade: " + studentGrades.getGrade("Eve"));  // Student not found
        // Calculating the class average grade
        System.out.println("Class average grade: " + studentGrades.calculateClassAverage());
    }
}
```

OUTPUT



```
run:
Aressha's grade: 85
Emaan's grade: 92
Faiq's grade: 78
Aima's grade: 95
Eve's grade: -1
Class average grade: 87.5
BUILD SUCCESSFUL (total time: 0 seconds)
```