LAB # 02

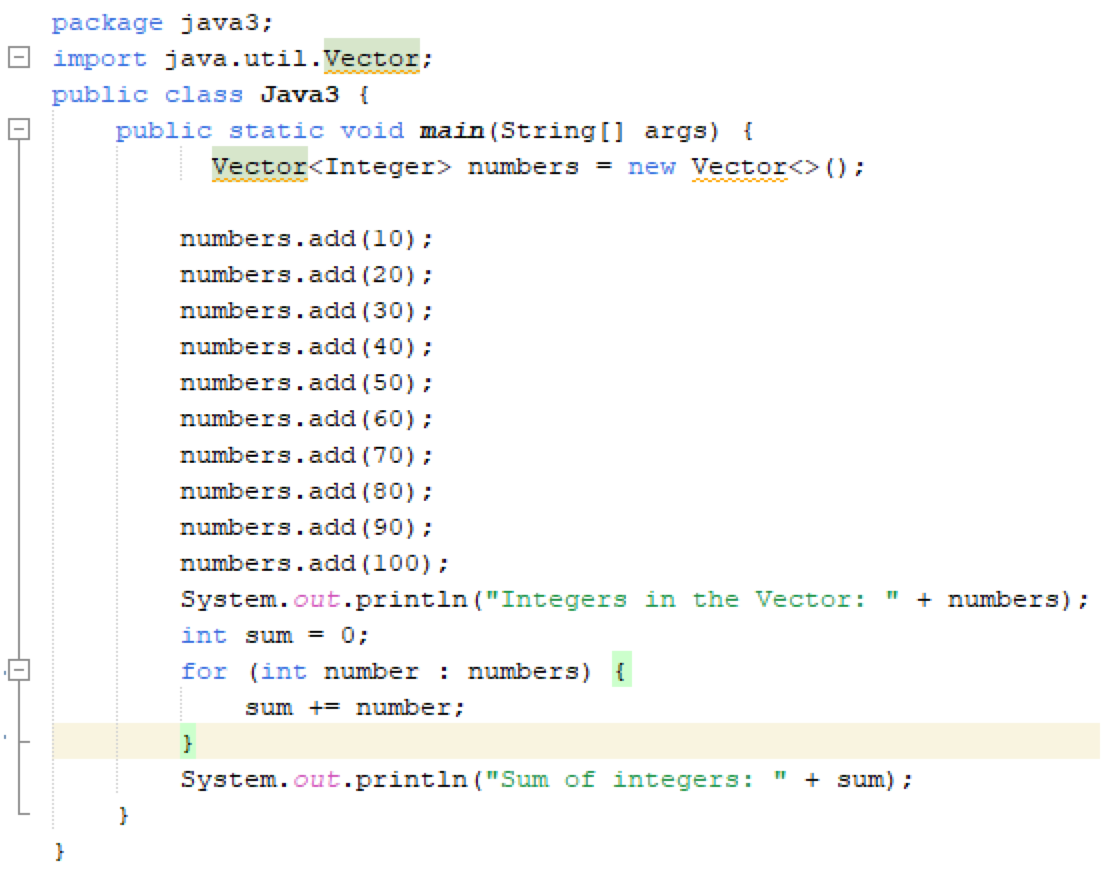
ArrayList and Vector in JAVA

OBJECTIVE: To implement ArrayList and Vector.

Lab Tasks

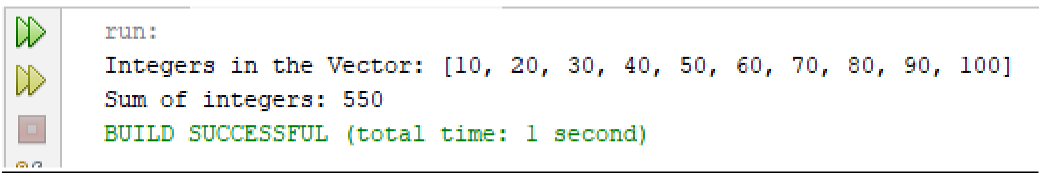
1. Write a program that initializes Vector with 10 integers in it. Display all the integers and sum of these integers.

CODE:



```java
package java3;
import java.util.Vector;
public class Java3 {
    public static void main(String[] args) {
        Vector<Integer> numbers = new Vector<>();

        numbers.add(10);
        numbers.add(20);
        numbers.add(30);
        numbers.add(40);
        numbers.add(50);
        numbers.add(60);
        numbers.add(70);
        numbers.add(80);
        numbers.add(90);
        numbers.add(100);
        System.out.println("Integers in the Vector: " + numbers);
        int sum = 0;
        for (int number : numbers) {
            sum += number;
        }
        System.out.println("Sum of integers: " + sum);
    }
}
```

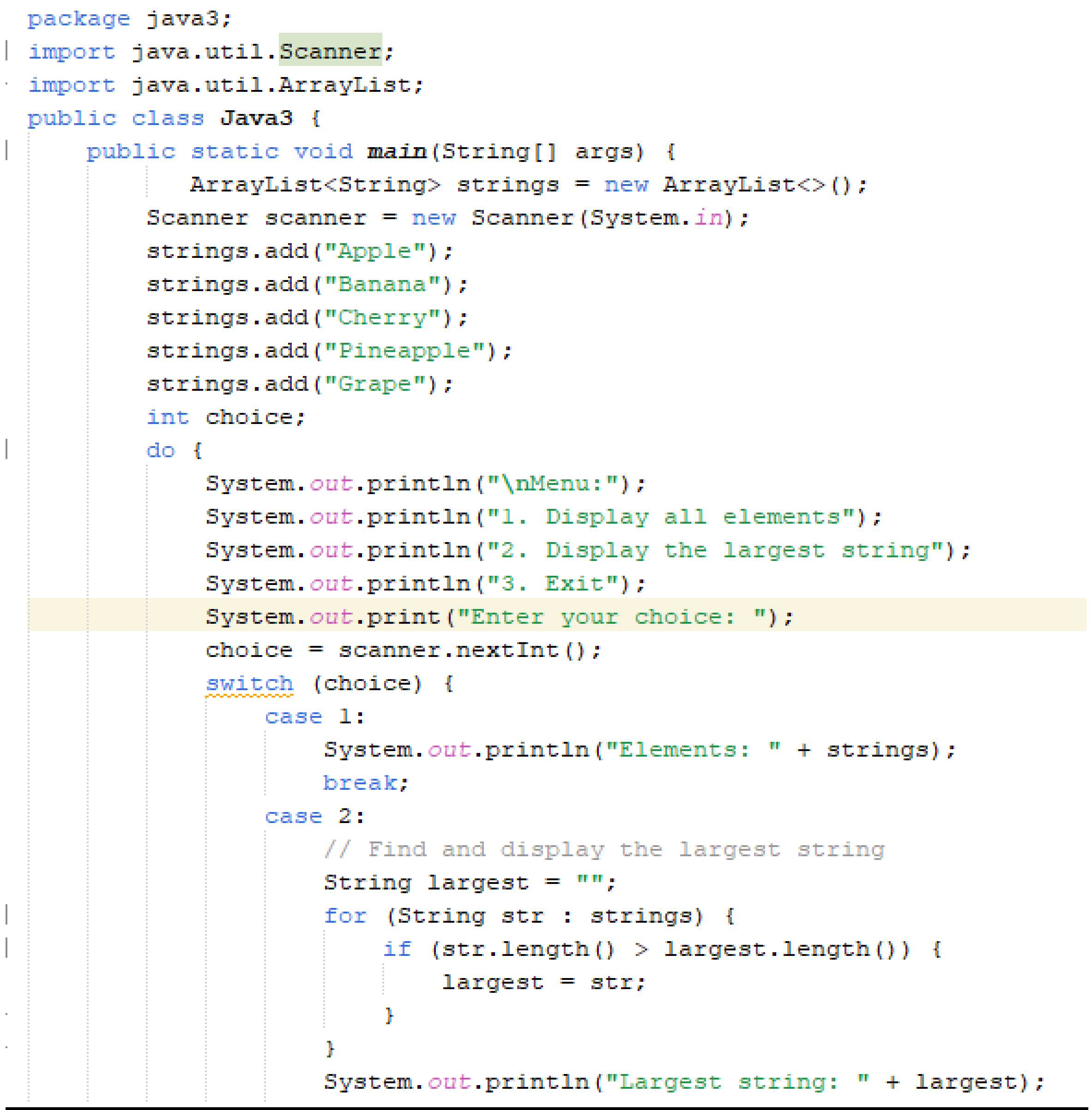
OUTPUT



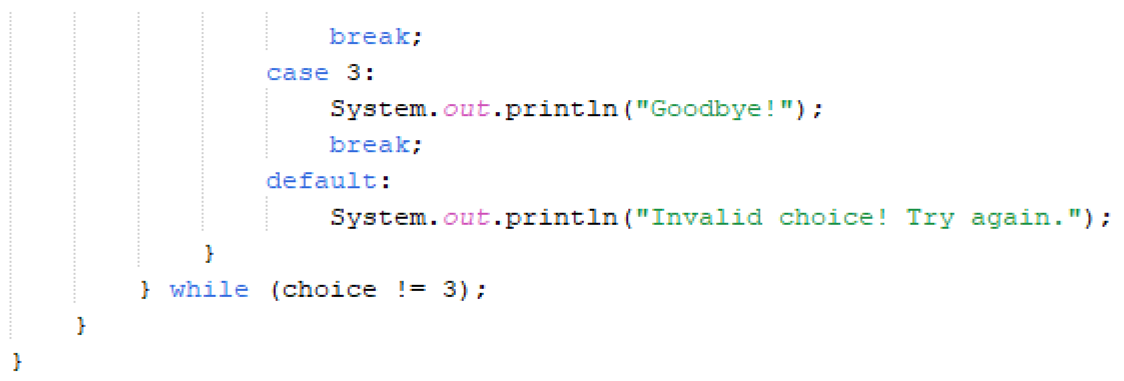
```
run:
Integers in the Vector: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
Sum of integers: 550
BUILD SUCCESSFUL (total time: 1 second)
```

2. Create a ArrayList of string. Write a menu driven program which:
   a. Displays all the elements
   b. Displays the largest String

CODE:

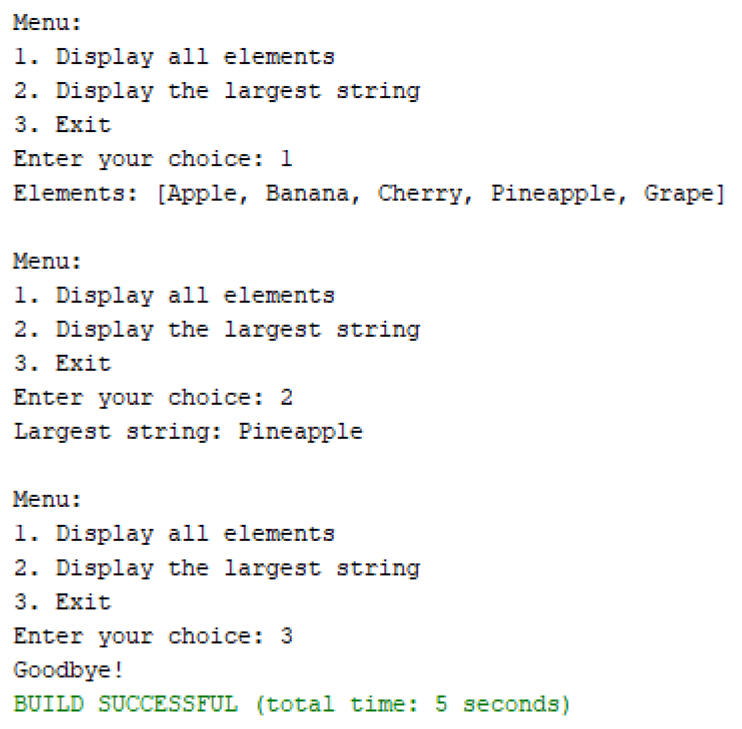


```java
package java3;
import java.util.Scanner;
import java.util.ArrayList;
public class Java3 {
    public static void main(String[] args) {
           ArrayList<String> strings = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        strings.add("Apple");
        strings.add("Banana");
        strings.add("Cherry");
        strings.add("Pineapple");
        strings.add("Grape");
        int choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("1. Display all elements");
            System.out.println("2. Display the largest string");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Elements: " + strings);
                    break;
                case 2:
                    // Find and display the largest string
                    String largest = "";
                    for (String str : strings) {
                        if (str.length() > largest.length()) {
                            largest = str;
                        }
                    }
                    System.out.println("Largest string: " + largest);
```



```
                    break;
                case 3:
                    System.out.println("Goodbye!");
                    break;
                default:
                    System.out.println("Invalid choice! Try again.");
            }
        } while (choice != 3);
    }
}
```

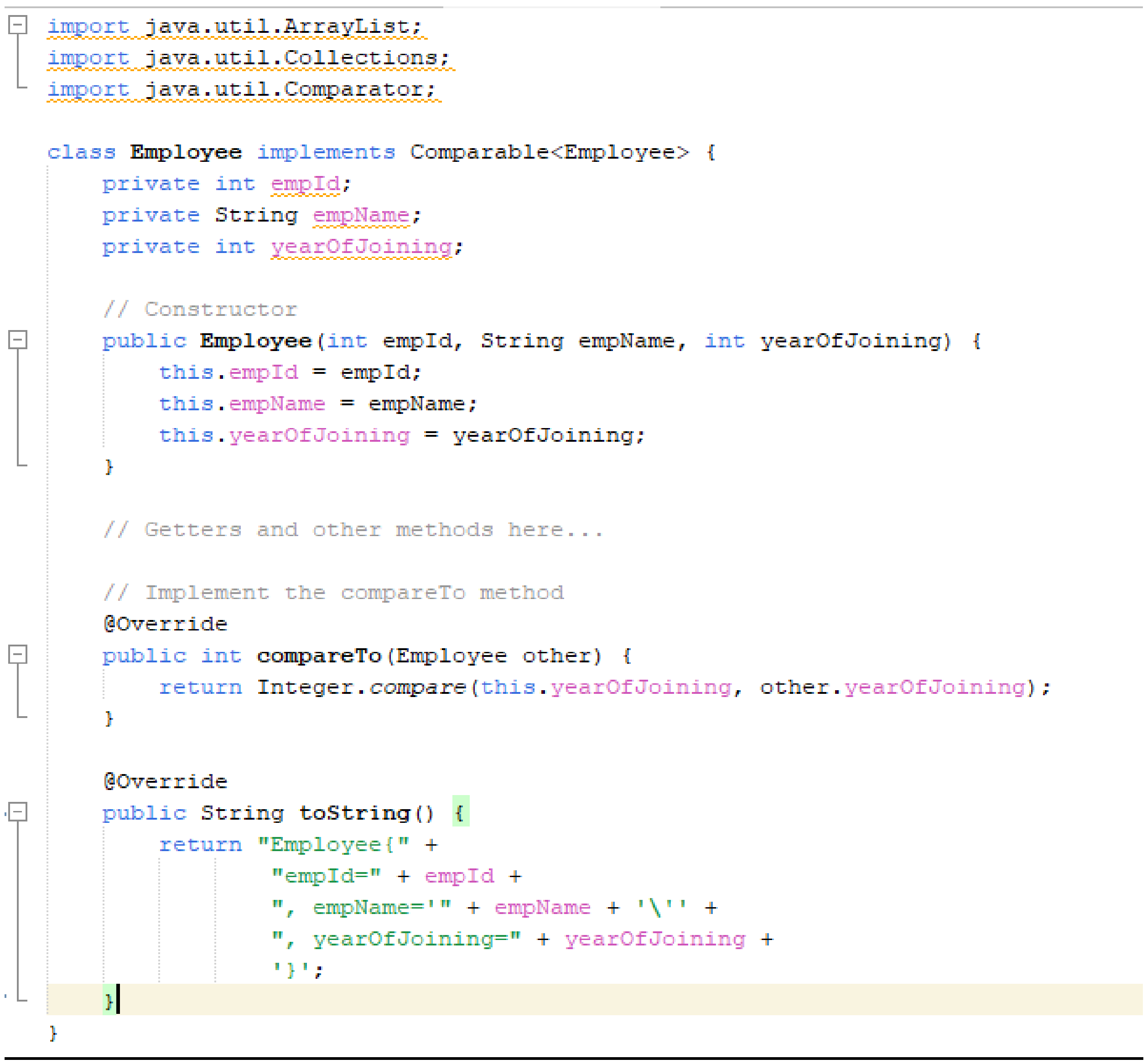
OUTPUT



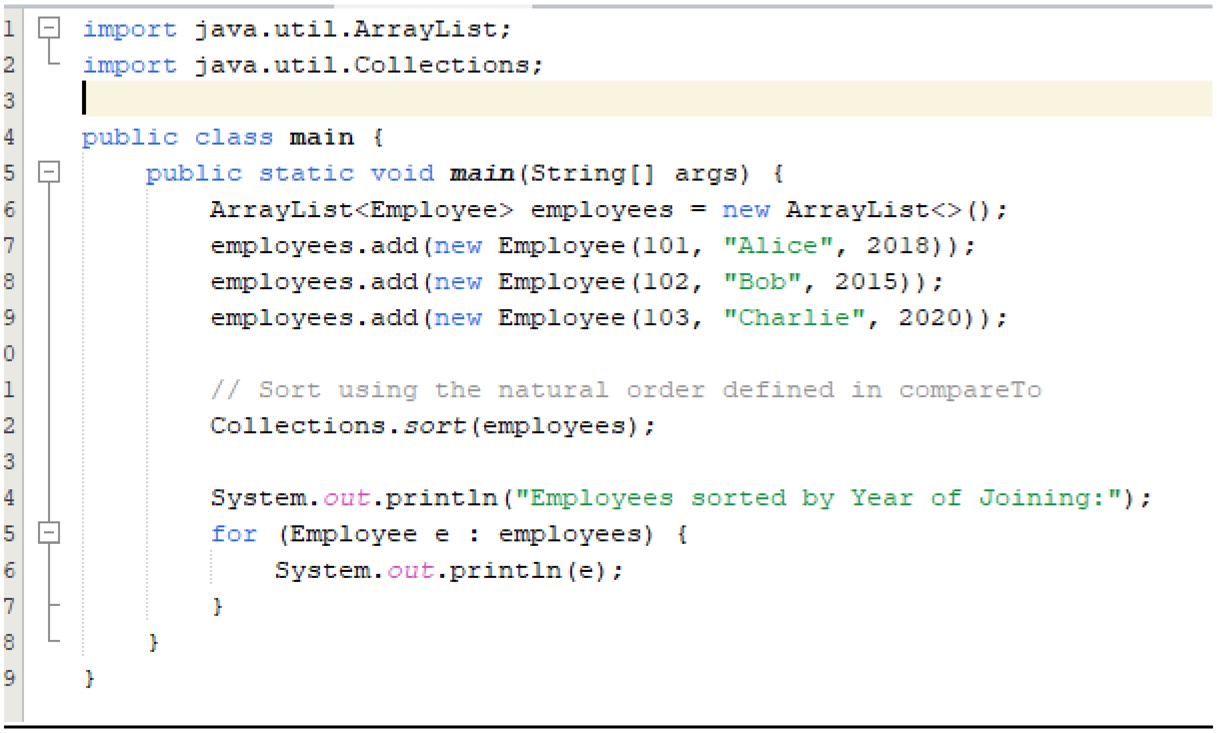
```
Menu:
1. Display all elements
2. Display the largest string
3. Exit
Enter your choice: 1
Elements: [Apple, Banana, Cherry, Pineapple, Grape]

Menu:
1. Display all elements
2. Display the largest string
3. Exit
Enter your choice: 2
Largest string: Pineapple

Menu:
1. Display all elements
2. Display the largest string
3. Exit
Enter your choice: 3
Goodbye!
BUILD SUCCESSFUL (total time: 5 seconds)
```

3. Create a Arraylist storing Employee details including Emp_id, Emp_Name, Emp_gender, Year_of_Joining (you can also add more attributes including these). Then sort the employees according to their joining year using Comparator and Comparable interfaces.

CODE:

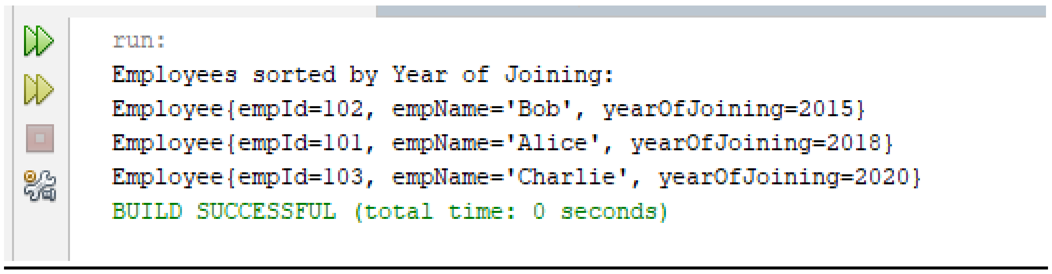


```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Employee implements Comparable<Employee> {
    private int empId;
    private String empName;
    private int yearOfJoining;

    // Constructor
    public Employee(int empId, String empName, int yearOfJoining) {
        this.empId = empId;
        this.empName = empName;
        this.yearOfJoining = yearOfJoining;
    }

    // Getters and other methods here...

    // Implement the compareTo method
    @Override
    public int compareTo(Employee other) {
        return Integer.compare(this.yearOfJoining, other.yearOfJoining);
    }

    @Override
    public String toString() {
        return "Employee{" +
                "empId=" + empId +
                ", empName='" + empName + '\'' +
                ", yearOfJoining=" + yearOfJoining +
                '}';
    }
}
```



```java
import java.util.ArrayList;
import java.util.Collections;

public class main {
    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();
        employees.add(new Employee(101, "Alice", 2018));
        employees.add(new Employee(102, "Bob", 2015));
        employees.add(new Employee(103, "Charlie", 2020));

        // Sort using the natural order defined in compareTo
        Collections.sort(employees);

        System.out.println("Employees sorted by Year of Joining:");
        for (Employee e : employees) {
            System.out.println(e);
        }
    }
}
```

OUTPUT:

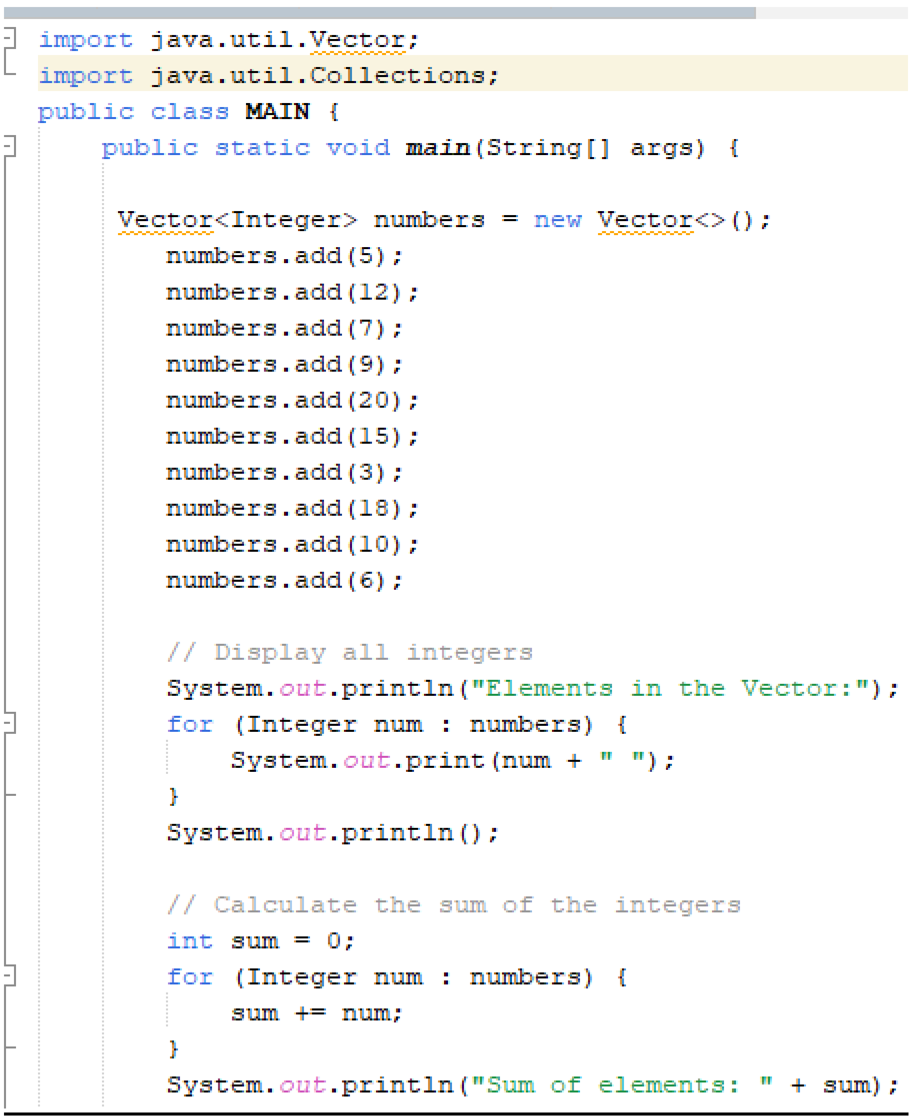


```
run:
Employees sorted by Year of Joining:
Employee{empId=102, empName='Bob', yearOfJoining=2015}
Employee{empId=101, empName='Alice', yearOfJoining=2018}
Employee{empId=103, empName='Charlie', yearOfJoining=2020}
BUILD SUCCESSFUL (total time: 0 seconds)
```

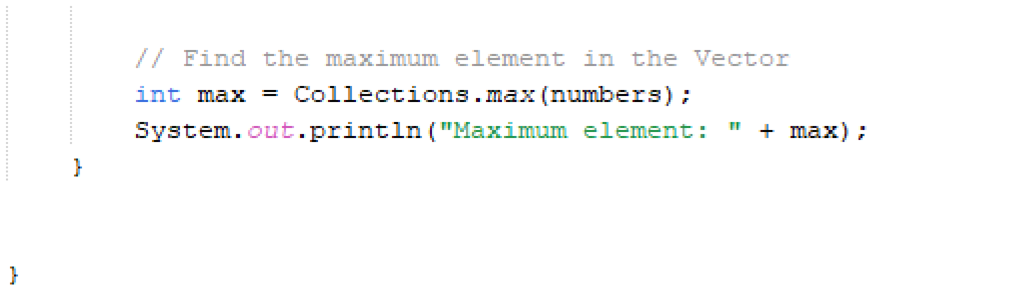
4. Write a program that initializes Vector with 10 integers in it.

- Display all the integers
- Sum of these integers.
- Find Maximum Element in Vector

CODE

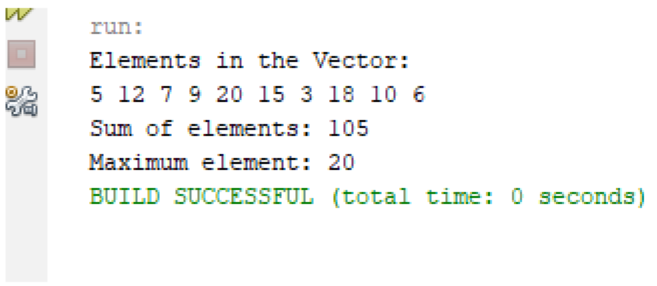


```
import java.util.Vector;
import java.util.Collections;
public class MAIN {
    public static void main(String[] args) {

     Vector<Integer> numbers = new Vector<>();
        numbers.add(5);
        numbers.add(12);
        numbers.add(7);
        numbers.add(9);
        numbers.add(20);
        numbers.add(15);
        numbers.add(3);
        numbers.add(18);
        numbers.add(10);
        numbers.add(6);

        // Display all integers
        System.out.println("Elements in the Vector:");
        for (Integer num : numbers) {
            System.out.print(num + " ");
        }
        System.out.println();

        // Calculate the sum of the integers
        int sum = 0;
        for (Integer num : numbers) {
            sum += num;
        }
        System.out.println("Sum of elements: " + sum);
```



```
        // Find the maximum element in the Vector
        int max = Collections.max(numbers);
        System.out.println("Maximum element: " + max);
    }


}
```

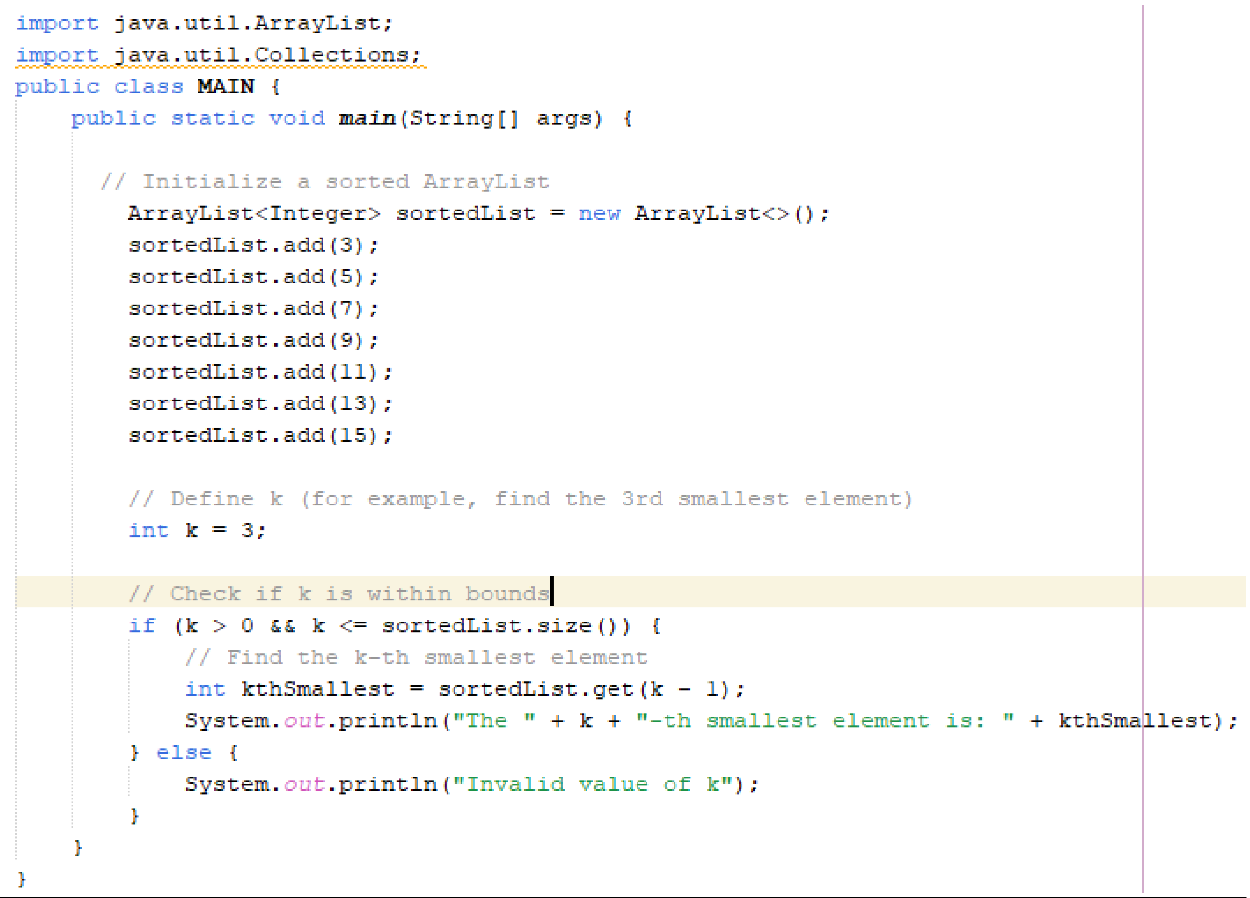
OUTPUT:



```
run:
Elements in the Vector:
5 12 7 9 20 15 3 18 10 6
Sum of elements: 105
Maximum element: 20
BUILD SUCCESSFUL (total time: 0 seconds)
```

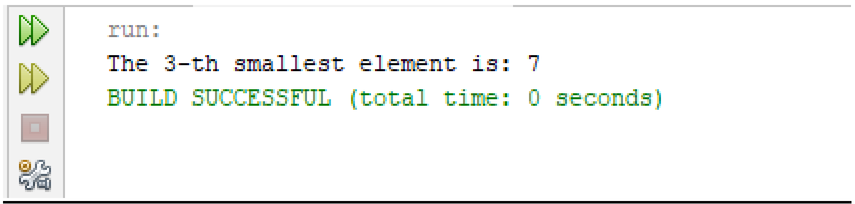
5. Find the k-th smallest element in a sorted ArrayList

CODE



```
import java.util.ArrayList;
import java.util.Collections;
public class MAIN {
    public static void main(String[] args) {

      // Initialize a sorted ArrayList
        ArrayList<Integer> sortedList = new ArrayList<>();
        sortedList.add(3);
        sortedList.add(5);
        sortedList.add(7);
        sortedList.add(9);
        sortedList.add(11);
        sortedList.add(13);
        sortedList.add(15);

        // Define k (for example, find the 3rd smallest element)
        int k = 3;

        // Check if k is within bounds
        if (k > 0 && k <= sortedList.size()) {
            // Find the k-th smallest element
            int kthSmallest = sortedList.get(k - 1);
            System.out.println("The " + k + "-th smallest element is: " + kthSmallest);
        } else {
            System.out.println("Invalid value of k");
        }
    }
}
```

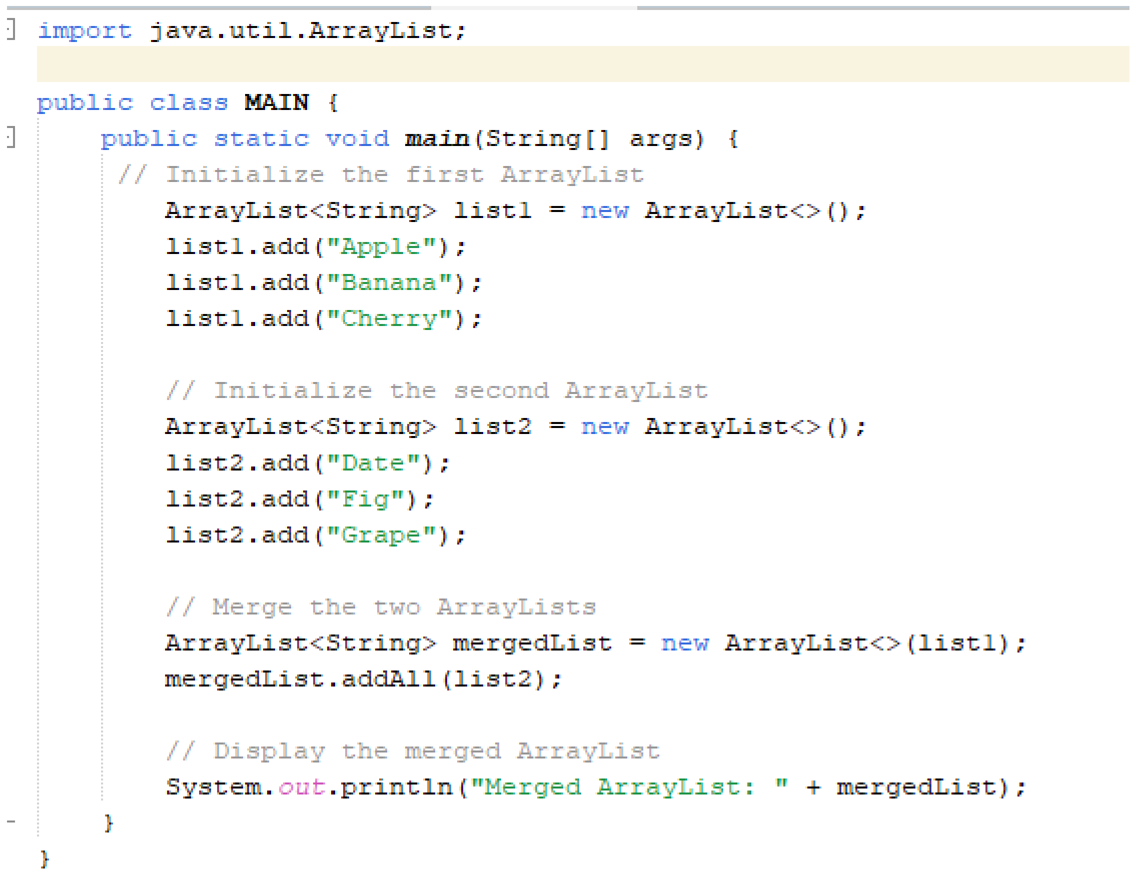
OUTPUT:



```
run:
The 3-th smallest element is: 7
BUILD SUCCESSFUL (total time: 0 seconds)
```

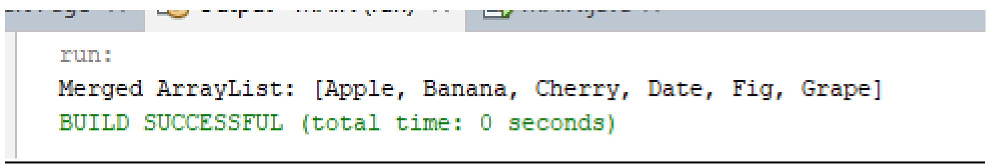
6. Write a program to merge two ArrayLists into one.

CODE:



```java
import java.util.ArrayList;

public class MAIN {
    public static void main(String[] args) {
     // Initialize the first ArrayList
        ArrayList<String> list1 = new ArrayList<>();
        list1.add("Apple");
        list1.add("Banana");
        list1.add("Cherry");

        // Initialize the second ArrayList
        ArrayList<String> list2 = new ArrayList<>();
        list2.add("Date");
        list2.add("Fig");
        list2.add("Grape");

        // Merge the two ArrayLists
        ArrayList<String> mergedList = new ArrayList<>(list1);
        mergedList.addAll(list2);

        // Display the merged ArrayList
        System.out.println("Merged ArrayList: " + mergedList);
    }
}
```

OUTPUT:

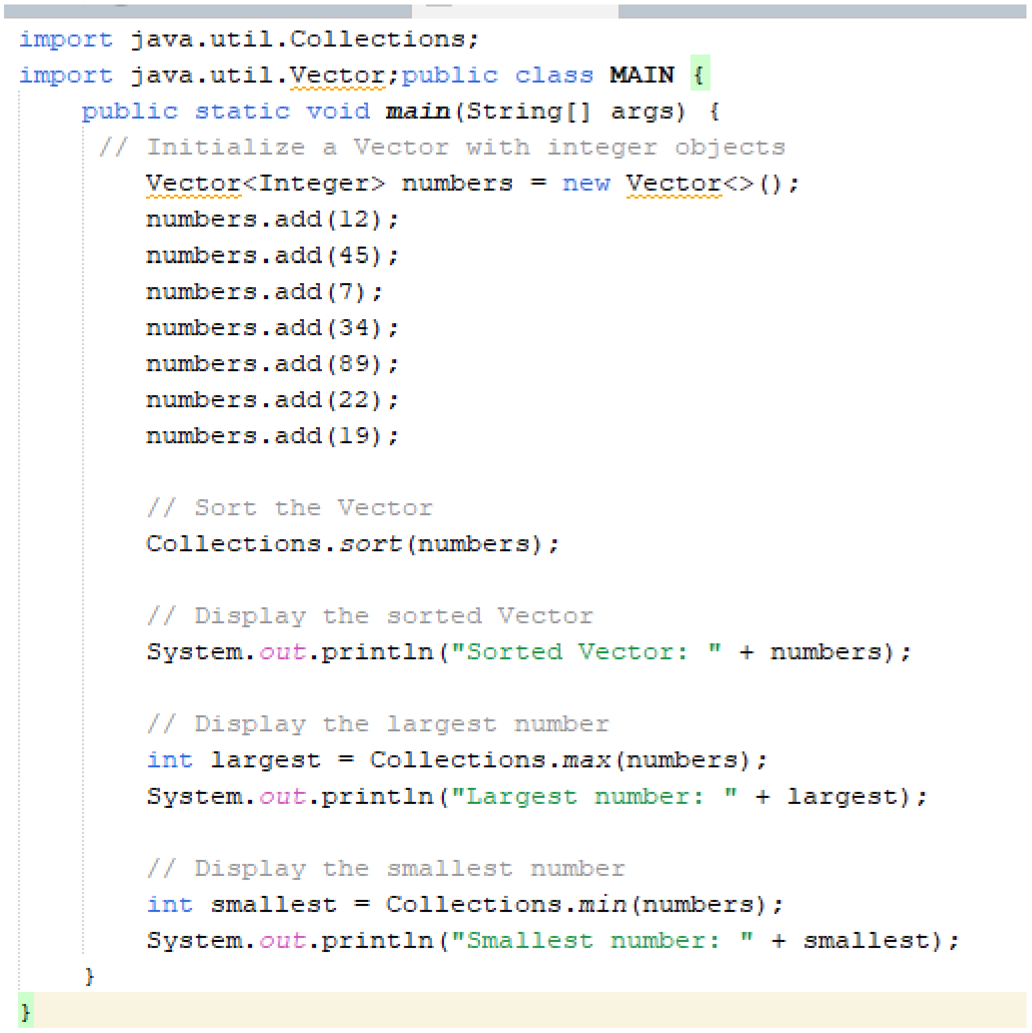


```
run:
Merged ArrayList: [Apple, Banana, Cherry, Date, Fig, Grape]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Home Tasks

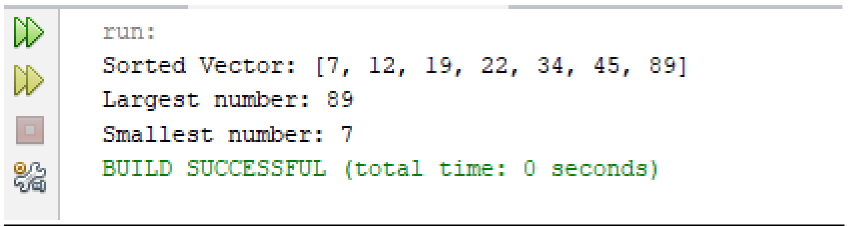
1. Create a Vector storing integer objects as an input.
   a. Sort the vector
   b. Display largest number
   c. Display smallest number

CODE:



```java
import java.util.Collections;
import java.util.Vector;public class MAIN {
    public static void main(String[] args) {
     // Initialize a Vector with integer objects
        Vector<Integer> numbers = new Vector<>();
        numbers.add(12);
        numbers.add(45);
        numbers.add(7);
        numbers.add(34);
        numbers.add(89);
        numbers.add(22);
        numbers.add(19);

        // Sort the Vector
        Collections.sort(numbers);

        // Display the sorted Vector
        System.out.println("Sorted Vector: " + numbers);

        // Display the largest number
        int largest = Collections.max(numbers);
        System.out.println("Largest number: " + largest);

        // Display the smallest number
        int smallest = Collections.min(numbers);
        System.out.println("Smallest number: " + smallest);
    }
}
```

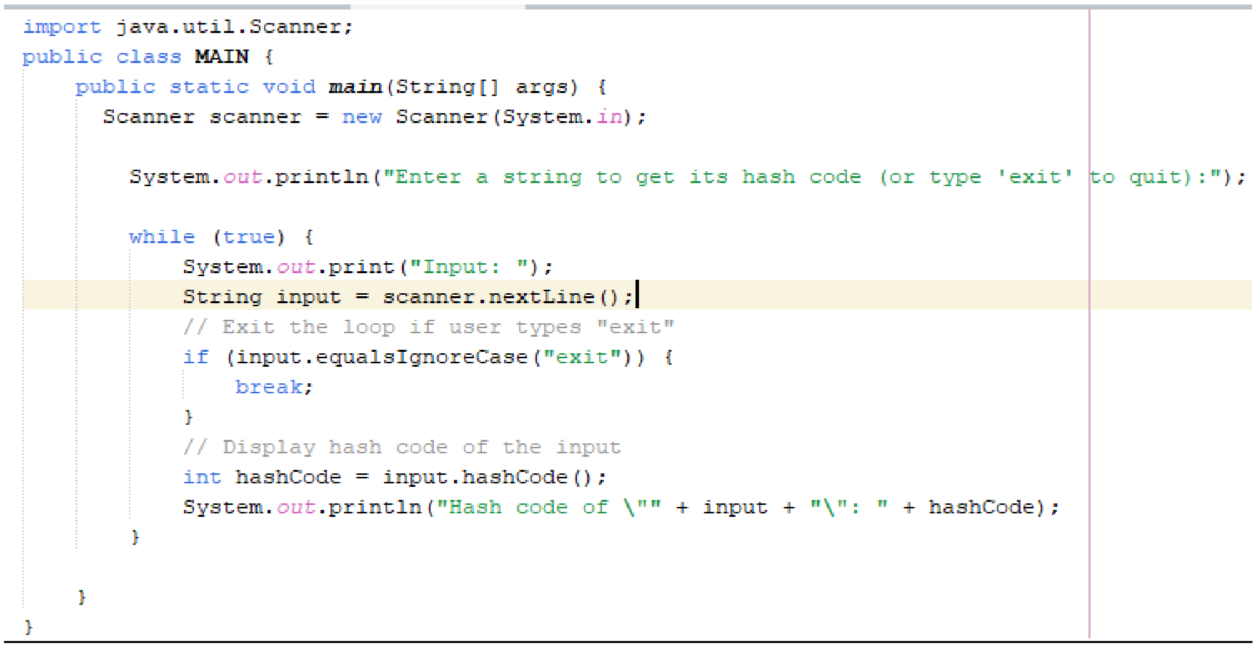
OUTPUT:



```
run:
Sorted Vector: [7, 12, 19, 22, 34, 45, 89]
Largest number: 89
Smallest number: 7
BUILD SUCCESSFUL (total time: 0 seconds)
```

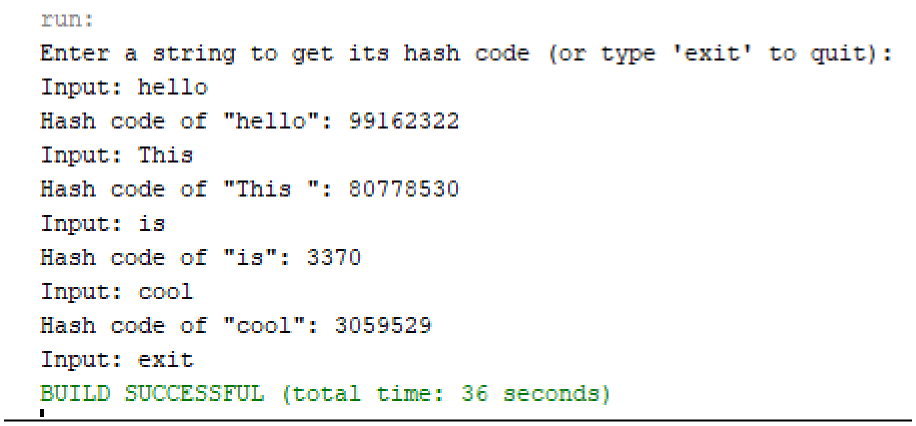
2. Write a java program which takes user input and gives hashcode value of those inputs using hashCode () method

CODE:



```
import java.util.Scanner;
public class MAIN {
    public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a string to get its hash code (or type 'exit' to quit):");

        while (true) {
            System.out.print("Input: ");
            String input = scanner.nextLine();
            // Exit the loop if user types "exit"
            if (input.equalsIgnoreCase("exit")) {
                break;
            }
            // Display hash code of the input
            int hashCode = input.hashCode();
            System.out.println("Hash code of \"" + input + "\": " + hashCode);
        }

    }
}
```

OUTPUT:



```
run:
Enter a string to get its hash code (or type 'exit' to quit):
Input: hello
Hash code of "hello": 99162322
Input: This
Hash code of "This ": 80778530
Input: is
Hash code of "is": 3370
Input: cool
Hash code of "cool": 3059529
Input: exit
BUILD SUCCESSFUL (total time: 36 seconds)
```

3. **Scenario based**

Create a java project, suppose you work for a company that needs to manage a list of employees. Each employee has a unique combination of a name and an ID. Your goal is to ensure that you can track employees effectively and avoid duplicate entries in your system.

Requirements

a. Employee Class: You need to create an Employee class that includes:

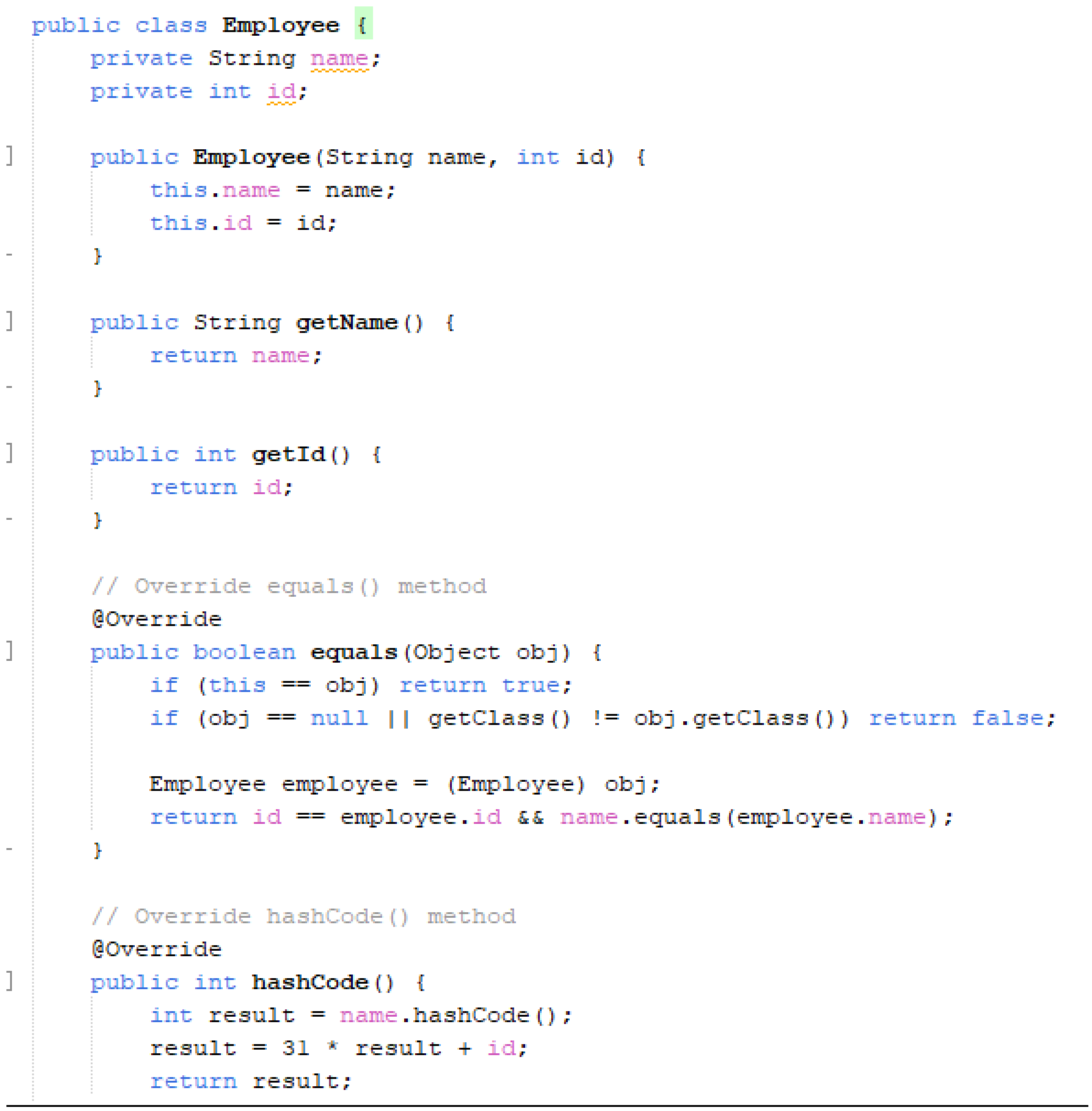
- name: The employee's name (String).
- id: The employee's unique identifier (int).
- Override the hashCode() and equals() methods to ensure that two employees are considered equal if they have the same name and id.

b. Employee Management: You will use a HashSet to store employee records. This will help you avoid duplicate entries.

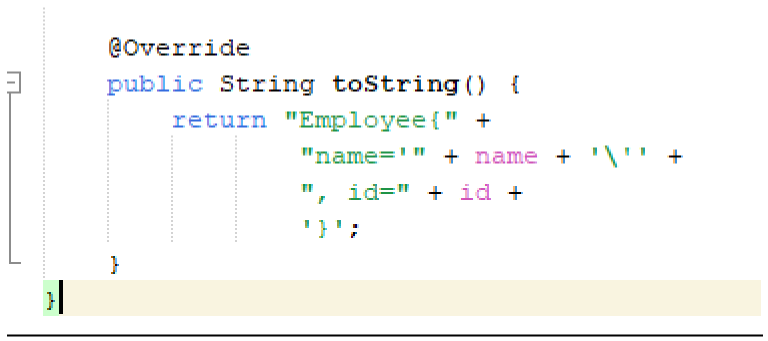
c. Operations: Implement operations to:

- Add new employees to the record.
- Check if an employee already exists in the records.
- Display all employees.

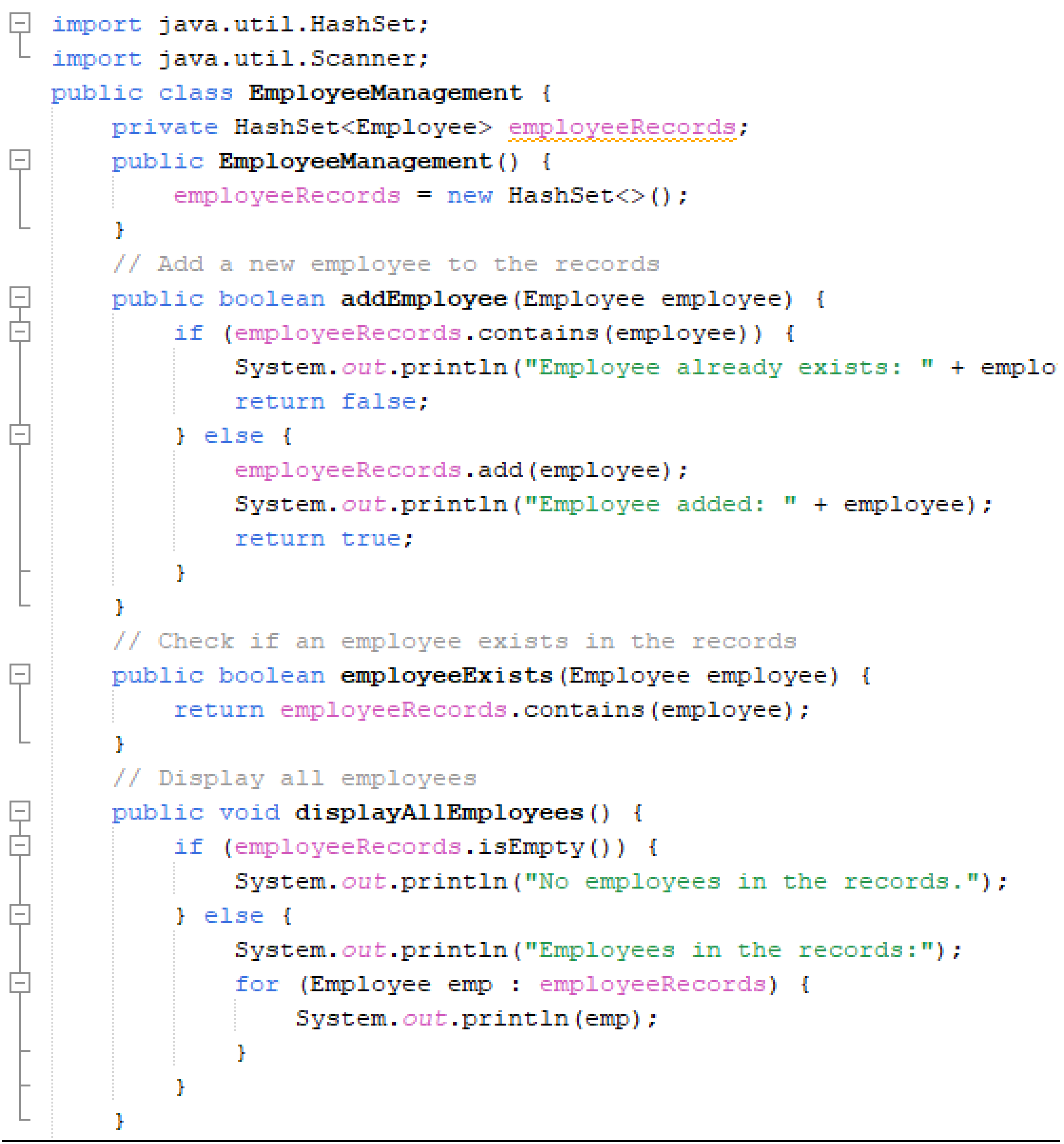
CODE:



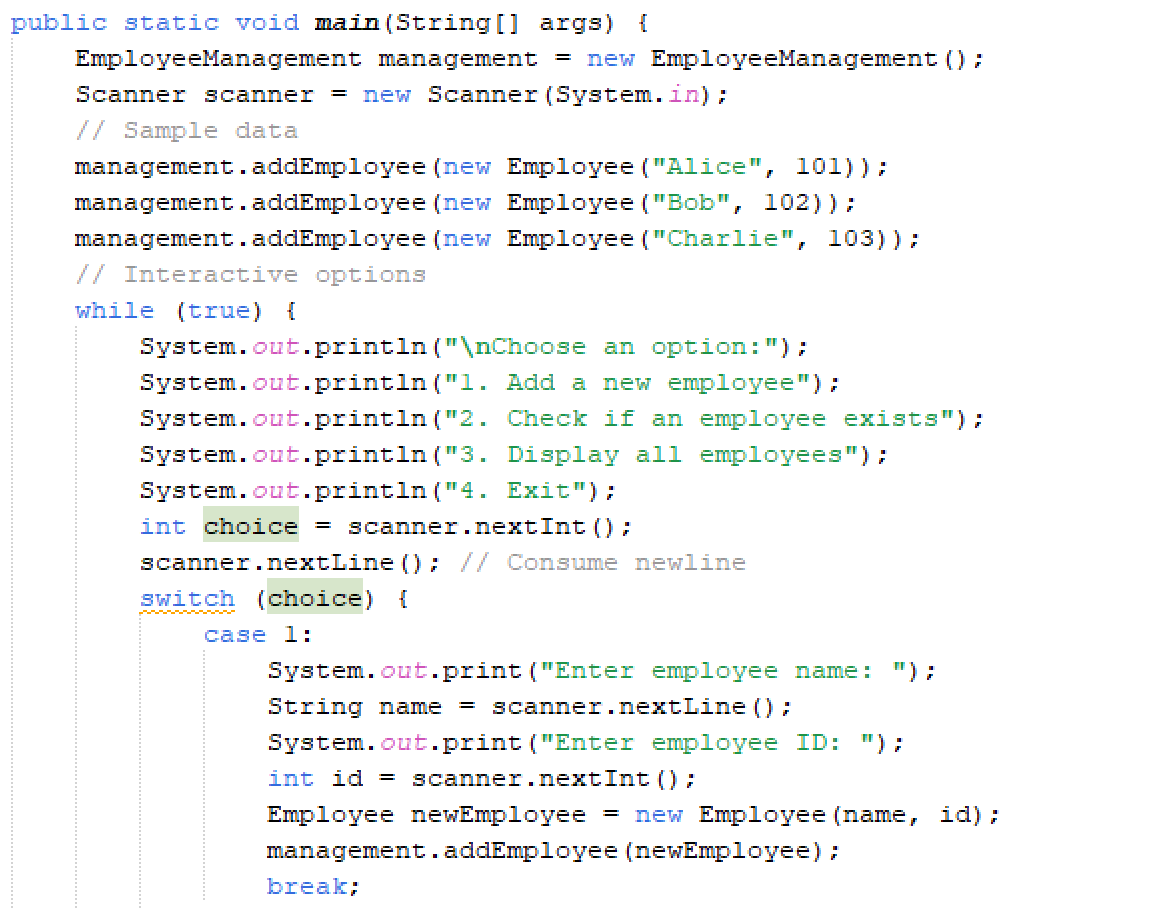
```java
public class Employee {
    private String name;
    private int id;

    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    // Override equals() method
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;

        Employee employee = (Employee) obj;
        return id == employee.id && name.equals(employee.name);
    }

    // Override hashCode() method
    @Override
    public int hashCode() {
        int result = name.hashCode();
        result = 31 * result + id;
        return result;
```



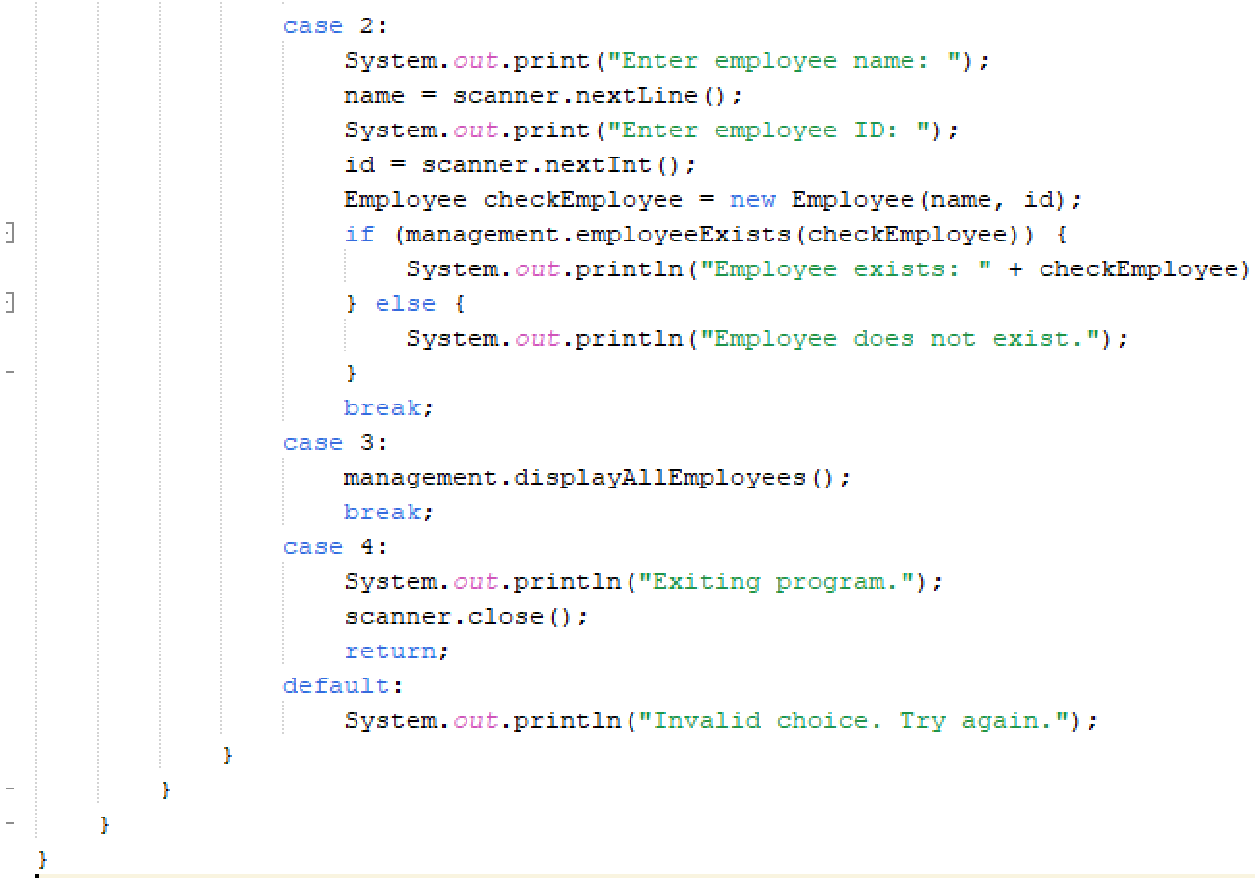
```java
    @Override
    public String toString() {
        return "Employee{" +
                "name='" + name + '\'' +
                ", id=" + id +
                '}';
    }
}
```



```java
import java.util.HashSet;
import java.util.Scanner;
public class EmployeeManagement {
    private HashSet<Employee> employeeRecords;
    public EmployeeManagement() {
        employeeRecords = new HashSet<>();
    }
    // Add a new employee to the records
    public boolean addEmployee(Employee employee) {
        if (employeeRecords.contains(employee)) {
            System.out.println("Employee already exists: " + emplo
            return false;
        } else {
            employeeRecords.add(employee);
            System.out.println("Employee added: " + employee);
            return true;
        }
    }
    // Check if an employee exists in the records
    public boolean employeeExists(Employee employee) {
        return employeeRecords.contains(employee);
    }
    // Display all employees
    public void displayAllEmployees() {
        if (employeeRecords.isEmpty()) {
            System.out.println("No employees in the records.");
        } else {
            System.out.println("Employees in the records:");
            for (Employee emp : employeeRecords) {
                System.out.println(emp);
            }
        }
    }
```

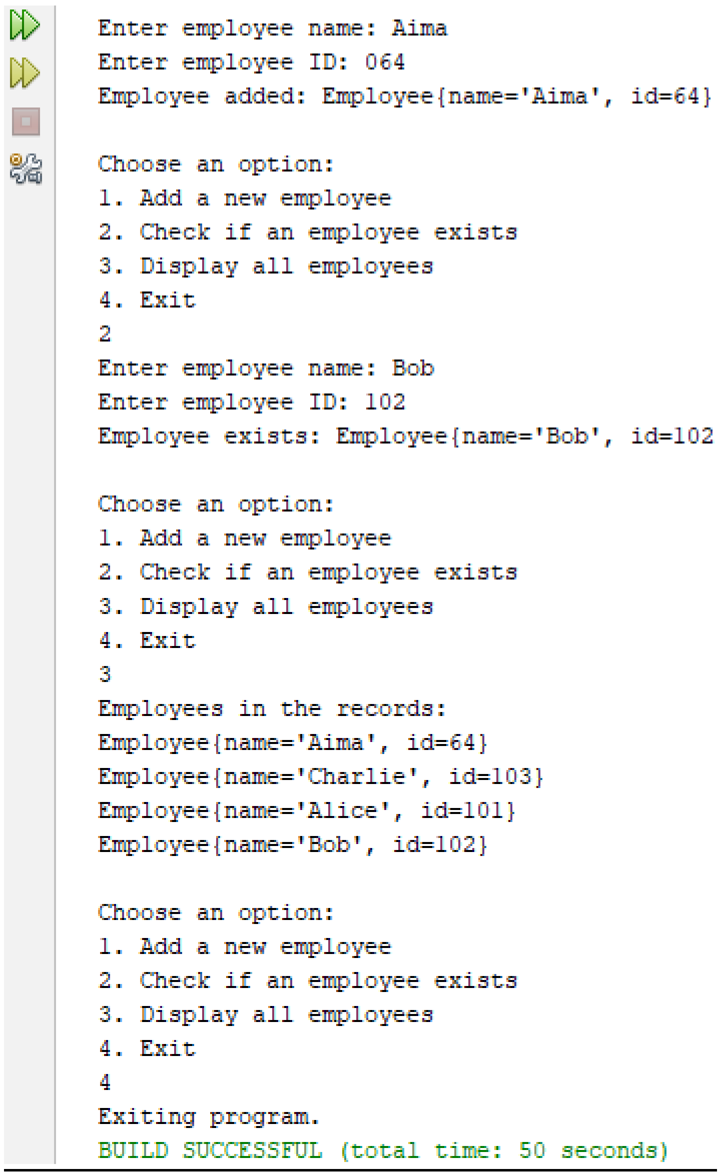


```java
public static void main(String[] args) {
    EmployeeManagement management = new EmployeeManagement();
    Scanner scanner = new Scanner(System.in);
    // Sample data
    management.addEmployee(new Employee("Alice", 101));
    management.addEmployee(new Employee("Bob", 102));
    management.addEmployee(new Employee("Charlie", 103));
    // Interactive options
    while (true) {
        System.out.println("\nChoose an option:");
        System.out.println("1. Add a new employee");
        System.out.println("2. Check if an employee exists");
        System.out.println("3. Display all employees");
        System.out.println("4. Exit");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        switch (choice) {
            case 1:
                System.out.print("Enter employee name: ");
                String name = scanner.nextLine();
                System.out.print("Enter employee ID: ");
                int id = scanner.nextInt();
                Employee newEmployee = new Employee(name, id);
                management.addEmployee(newEmployee);
                break;
```



```java
            case 2:
                System.out.print("Enter employee name: ");
                name = scanner.nextLine();
                System.out.print("Enter employee ID: ");
                id = scanner.nextInt();
                Employee checkEmployee = new Employee(name, id);
                if (management.employeeExists(checkEmployee)) {
                    System.out.println("Employee exists: " + checkEmployee)
                } else {
                    System.out.println("Employee does not exist.");
                }
                break;
            case 3:
                management.displayAllEmployees();
                break;
            case 4:
                System.out.println("Exiting program.");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
}
```

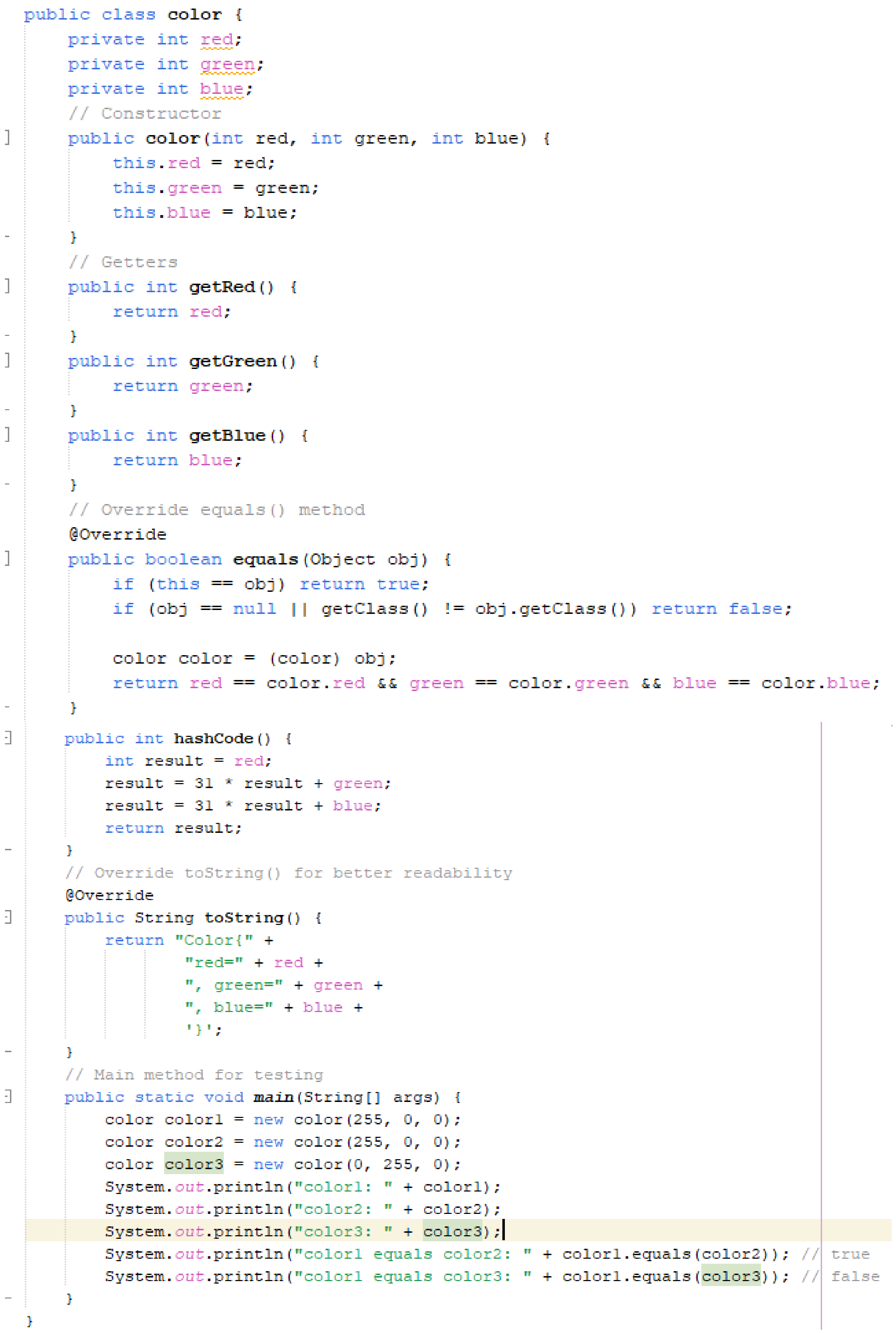
OUTPUT:



```
Enter employee name: Aima
Enter employee ID: 064
Employee added: Employee{name='Aima', id=64}

Choose an option:
1. Add a new employee
2. Check if an employee exists
3. Display all employees
4. Exit
2
Enter employee name: Bob
Enter employee ID: 102
Employee exists: Employee{name='Bob', id=102

Choose an option:
1. Add a new employee
2. Check if an employee exists
3. Display all employees
4. Exit
3
Employees in the records:
Employee{name='Aima', id=64}
Employee{name='Charlie', id=103}
Employee{name='Alice', id=101}
Employee{name='Bob', id=102}

Choose an option:
1. Add a new employee
2. Check if an employee exists
3. Display all employees
4. Exit
4
Exiting program.
BUILD SUCCESSFUL (total time: 50 seconds)
```

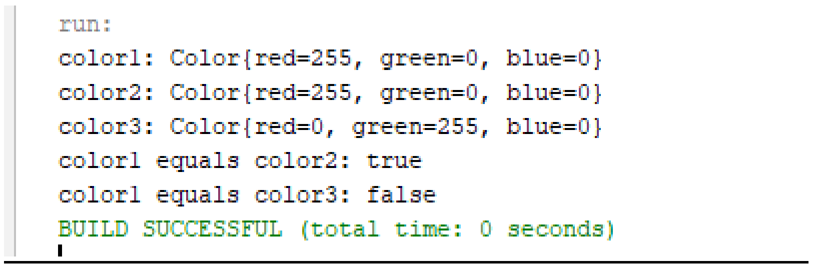
4.Create a Color class that has red, green, and blue values. Two colors are considered equal if their RGB values are the same

CODE:



```java
public class color {
    private int red;
    private int green;
    private int blue;
    // Constructor
    public color(int red, int green, int blue) {
        this.red = red;
        this.green = green;
        this.blue = blue;
    }
    // Getters
    public int getRed() {
        return red;
    }
    public int getGreen() {
        return green;
    }
    public int getBlue() {
        return blue;
    }
    // Override equals() method
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;

        color color = (color) obj;
        return red == color.red && green == color.green && blue == color.blue;
    }
    public int hashCode() {
        int result = red;
        result = 31 * result + green;
        result = 31 * result + blue;
        return result;
    }
    // Override toString() for better readability
    @Override
    public String toString() {
        return "Color{" +
                "red=" + red +
                ", green=" + green +
                ", blue=" + blue +
                '}';
    }
    // Main method for testing
    public static void main(String[] args) {
        color color1 = new color(255, 0, 0);
        color color2 = new color(255, 0, 0);
        color color3 = new color(0, 255, 0);
        System.out.println("color1: " + color1);
        System.out.println("color2: " + color2);
        System.out.println("color3: " + color3);
        System.out.println("color1 equals color2: " + color1.equals(color2)); // true
        System.out.println("color1 equals color3: " + color1.equals(color3)); // false
    }
}
```

OUTPUT:



```
run:
color1: Color{red=255, green=0, blue=0}
color2: Color{red=255, green=0, blue=0}
color3: Color{red=0, green=255, blue=0}
color1 equals color2: true
color1 equals color3: false
BUILD SUCCESSFUL (total time: 0 seconds)
```