SE2012 -OBJECT ORIENTED ANALYSIS AND DESIGN

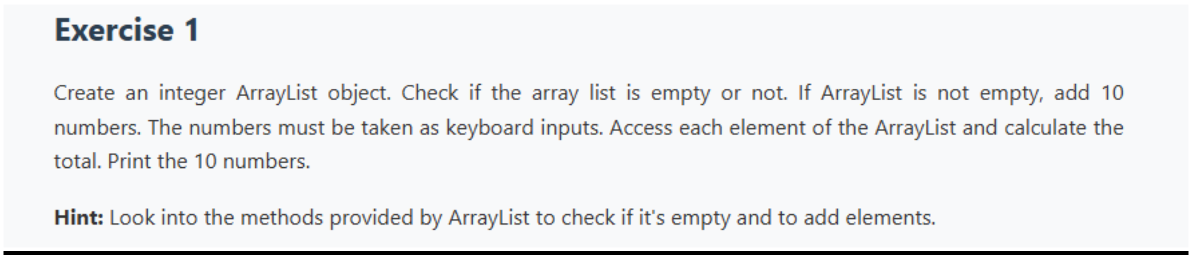
Worksheet 07: Generics & Collections in Java

2025 Semester 2 | Year 2, Semester 1

ID : IT24610328

Collections and collection types

Exercise 1

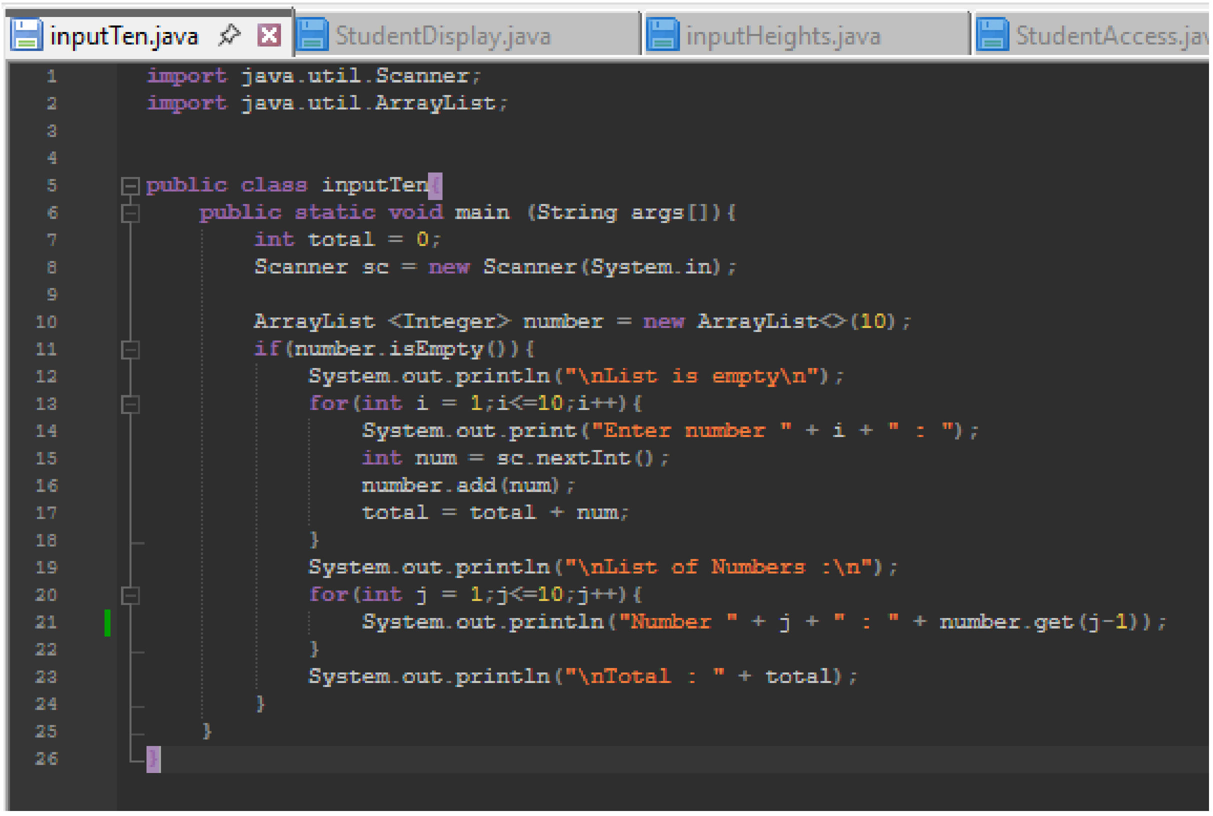


Exercise 1

Create an integer ArrayList object. Check if the array list is empty or not. If ArrayList is not empty, add 10 numbers. The numbers must be taken as keyboard inputs. Access each element of the ArrayList and calculate the total. Print the 10 numbers.

Hint: Look into the methods provided by ArrayList to check if it's empty and to add elements.

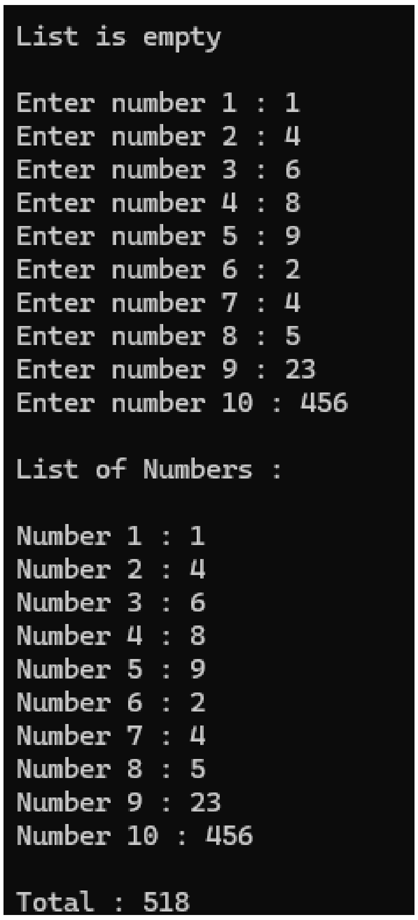
Code



inputTen.java | StudentDisplay.java | inputHeights.java | StudentAccess.jav

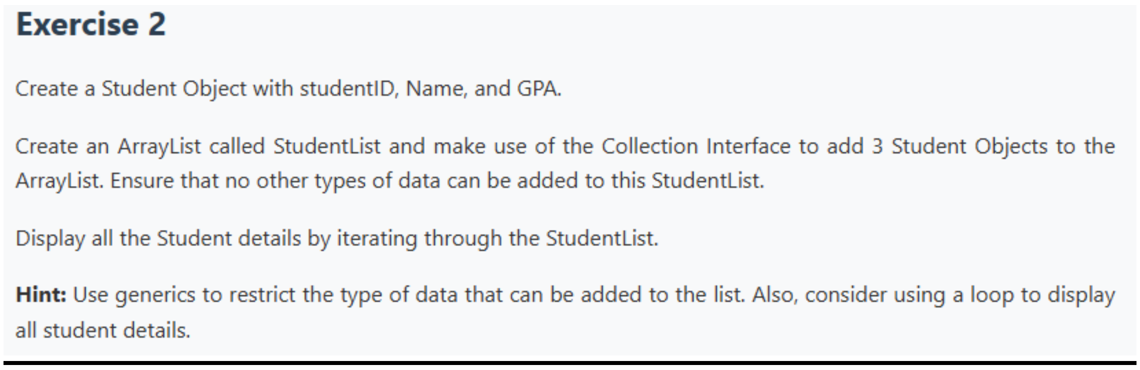
```java
import java.util.Scanner;
import java.util.ArrayList;


public class inputTen{
    public static void main (String args[]){
        int total = 0;
        Scanner sc = new Scanner(System.in);

        ArrayList <Integer> number = new ArrayList<>(10);
        if(number.isEmpty()){
            System.out.println("\nList is empty\n");
            for(int i = 1;i<=10;i++){
                System.out.print("Enter number " + i + " : ");
                int num = sc.nextInt();
                number.add(num);
                total = total + num;
            }
            System.out.println("\nList of Numbers :\n");
            for(int j = 1;j<=10;j++){
                System.out.println("Number " + j + " : " + number.get(j-1));
            }
            System.out.println("\nTotal : " + total);
        }
    }
}
```

Test case



```
List is empty

Enter number 1 : 1
Enter number 2 : 4
Enter number 3 : 6
Enter number 4 : 8
Enter number 5 : 9
Enter number 6 : 2
Enter number 7 : 4
Enter number 8 : 5
Enter number 9 : 23
Enter number 10 : 456

List of Numbers :

Number 1 : 1
Number 2 : 4
Number 3 : 6
Number 4 : 8
Number 5 : 9
Number 6 : 2
Number 7 : 4
Number 8 : 5
Number 9 : 23
Number 10 : 456

Total : 518
```

Exercise 2



Exercise 2

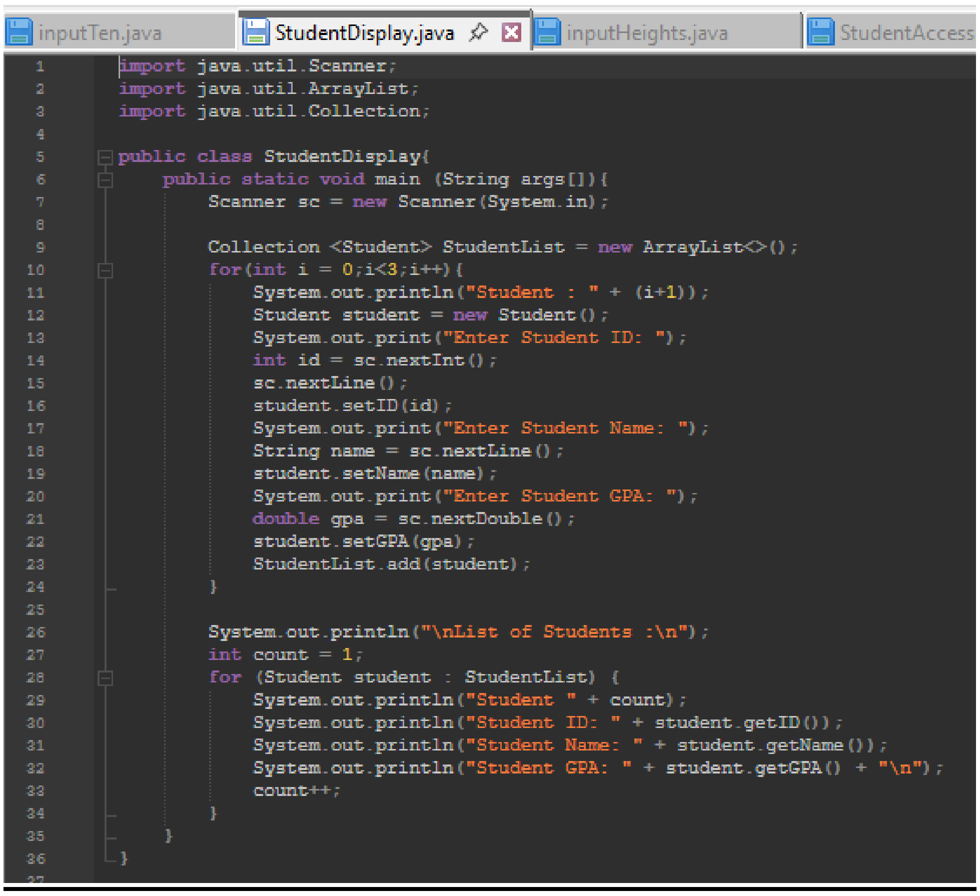
Create a Student Object with studentID, Name, and GPA.

Create an ArrayList called StudentList and make use of the Collection Interface to add 3 Student Objects to the ArrayList. Ensure that no other types of data can be added to this StudentList.

Display all the Student details by iterating through the StudentList.

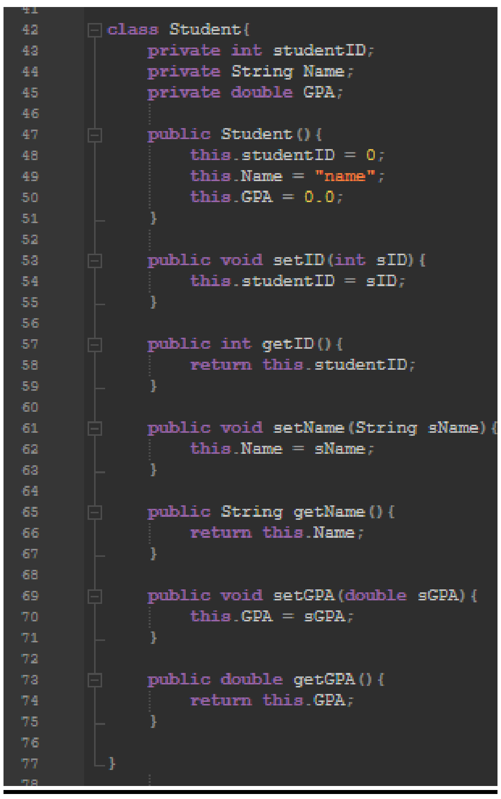
Hint: Use generics to restrict the type of data that can be added to the list. Also, consider using a loop to display all student details.

Code



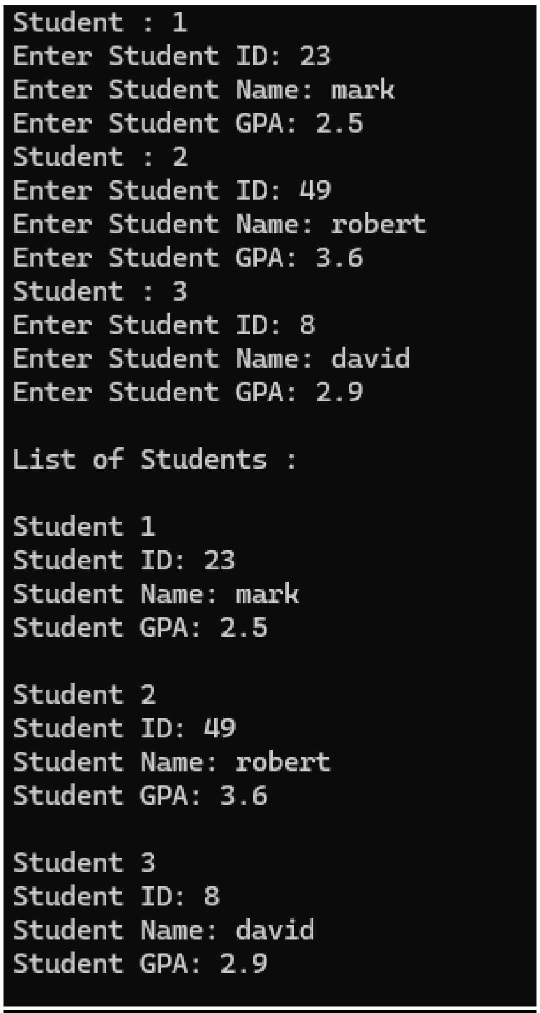
```java
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Collection;

public class StudentDisplay{
    public static void main (String args[]){
        Scanner sc = new Scanner(System.in);

        Collection <Student> StudentList = new ArrayList<>();
        for(int i = 0;i<3;i++){
            System.out.println("Student : " + (i+1));
            Student student = new Student();
            System.out.print("Enter Student ID: ");
            int id = sc.nextInt();
            sc.nextLine();
            student.setID(id);
            System.out.print("Enter Student Name: ");
            String name = sc.nextLine();
            student.setName(name);
            System.out.print("Enter Student GPA: ");
            double gpa = sc.nextDouble();
            student.setGPA(gpa);
            StudentList.add(student);
        }

        System.out.println("\nList of Students :\n");
        int count = 1;
        for (Student student : StudentList) {
            System.out.println("Student " + count);
            System.out.println("Student ID: " + student.getID());
            System.out.println("Student Name: " + student.getName());
            System.out.println("Student GPA: " + student.getGPA() + "\n");
            count++;
        }
    }
}
```

Student Class



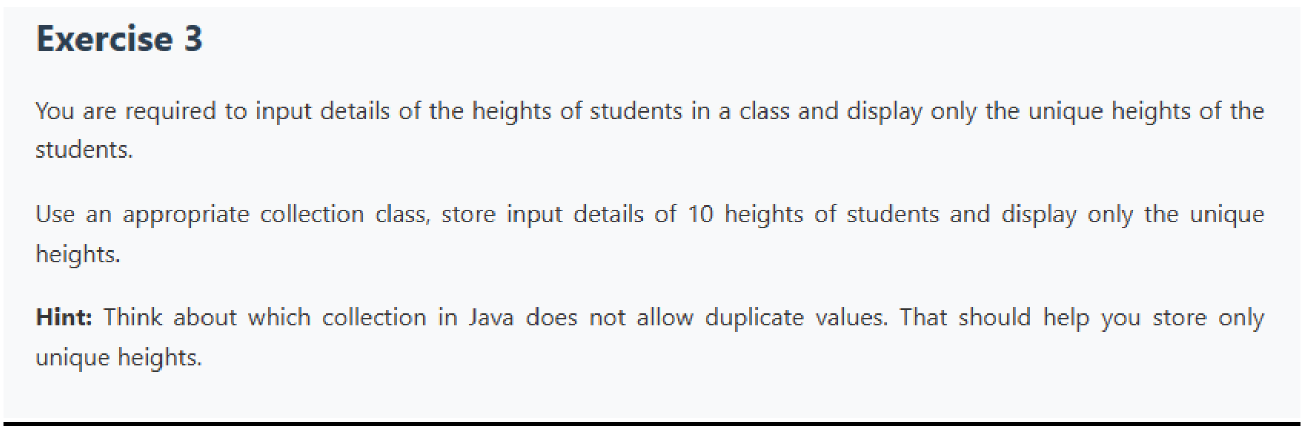
```java
class Student{
    private int studentID;
    private String Name;
    private double GPA;

    public Student(){
        this.studentID = 0;
        this.Name = "name";
        this.GPA = 0.0;
    }

    public void setID(int sID){
        this.studentID = sID;
    }

    public int getID(){
        return this.studentID;
    }

    public void setName(String sName){
        this.Name = sName;
    }

    public String getName(){
        return this.Name;
    }

    public void setGPA(double sGPA){
        this.GPA = sGPA;
    }

    public double getGPA(){
        return this.GPA;
    }
}
```

Test case



```
Student : 1
Enter Student ID: 23
Enter Student Name: mark
Enter Student GPA: 2.5
Student : 2
Enter Student ID: 49
Enter Student Name: robert
Enter Student GPA: 3.6
Student : 3
Enter Student ID: 8
Enter Student Name: david
Enter Student GPA: 2.9

List of Students :

Student 1
Student ID: 23
Student Name: mark
Student GPA: 2.5

Student 2
Student ID: 49
Student Name: robert
Student GPA: 3.6

Student 3
Student ID: 8
Student Name: david
Student GPA: 2.9
```

Exercise 3



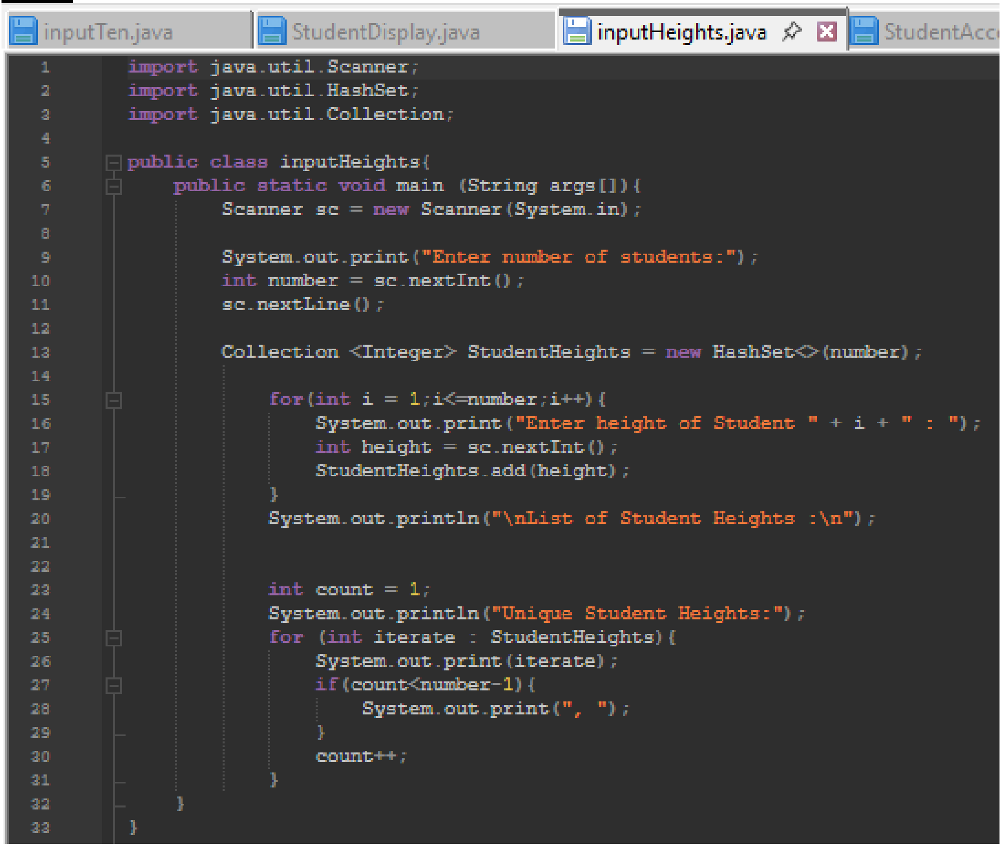
Exercise 3

You are required to input details of the heights of students in a class and display only the unique heights of the students.

Use an appropriate collection class, store input details of 10 heights of students and display only the unique heights.

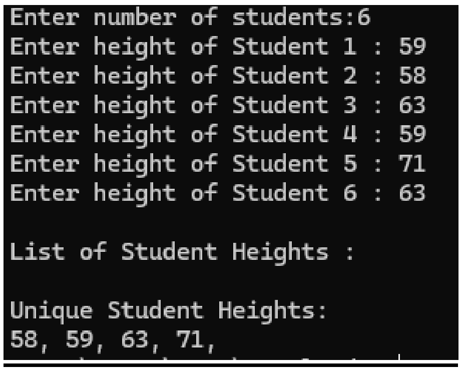
Hint: Think about which collection in Java does not allow duplicate values. That should help you store only unique heights.

Code



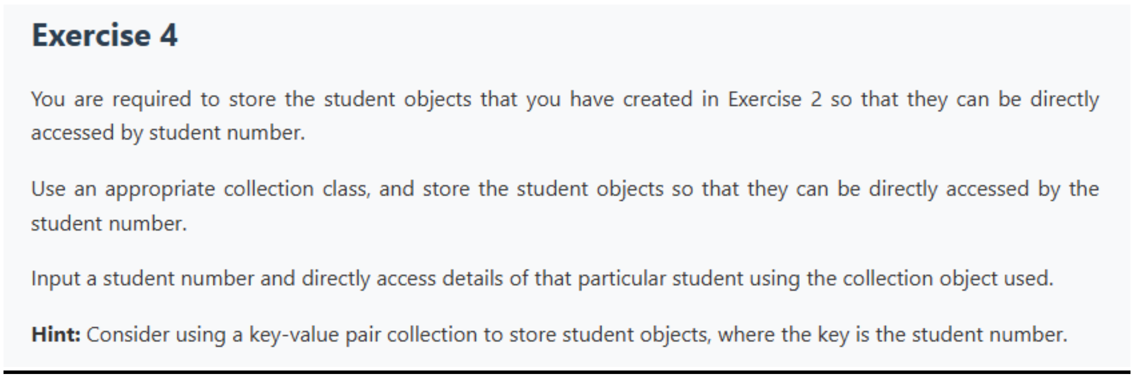
```java
import java.util.Scanner;
import java.util.HashSet;
import java.util.Collection;

public class inputHeights{
    public static void main (String args[]){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of students:");
        int number = sc.nextInt();
        sc.nextLine();

        Collection <Integer> StudentHeights = new HashSet<>(number);

            for(int i = 1;i<=number;i++){
                System.out.print("Enter height of Student " + i + " : ");
                int height = sc.nextInt();
                StudentHeights.add(height);
            }
            System.out.println("\nList of Student Heights :\n");


            int count = 1;
            System.out.println("Unique Student Heights:");
            for (int iterate : StudentHeights){
                System.out.print(iterate);
                if(count<number-1){
                    System.out.print(", ");
                }
                count++;
            }
    }
}
```

Test case



```
Enter number of students:6
Enter height of Student 1 : 59
Enter height of Student 2 : 58
Enter height of Student 3 : 63
Enter height of Student 4 : 59
Enter height of Student 5 : 71
Enter height of Student 6 : 63

List of Student Heights :

Unique Student Heights:
58, 59, 63, 71,
```

Exercise 4



Exercise 4

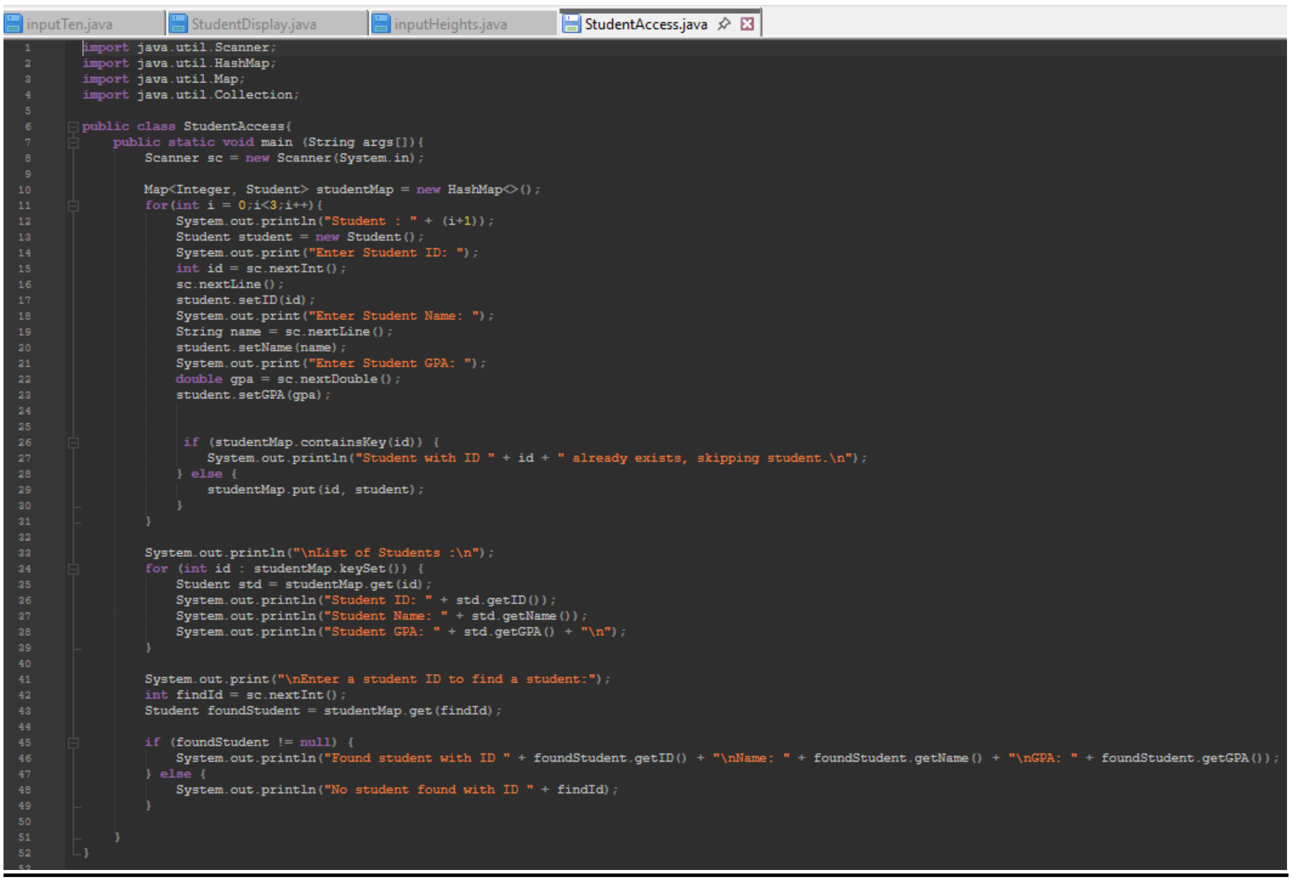
You are required to store the student objects that you have created in Exercise 2 so that they can be directly accessed by student number.

Use an appropriate collection class, and store the student objects so that they can be directly accessed by the student number.

Input a student number and directly access details of that particular student using the collection object used.

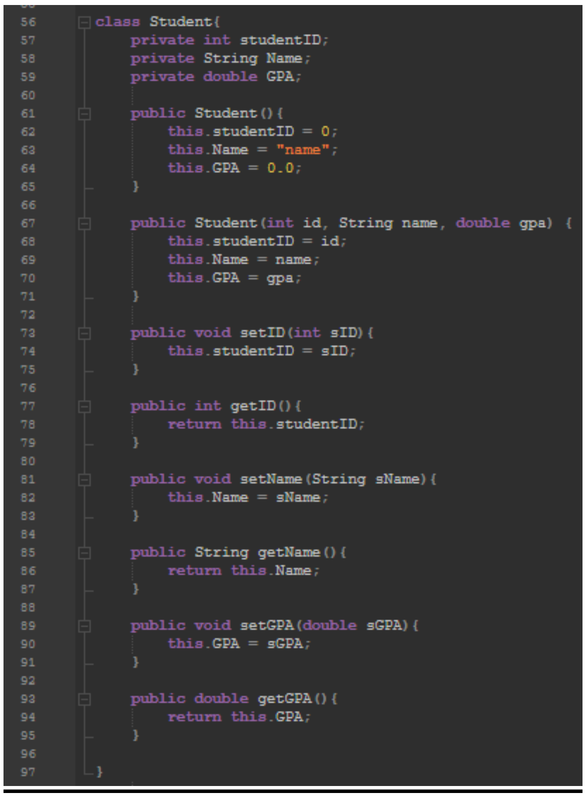
Hint: Consider using a key-value pair collection to store student objects, where the key is the student number.

Code



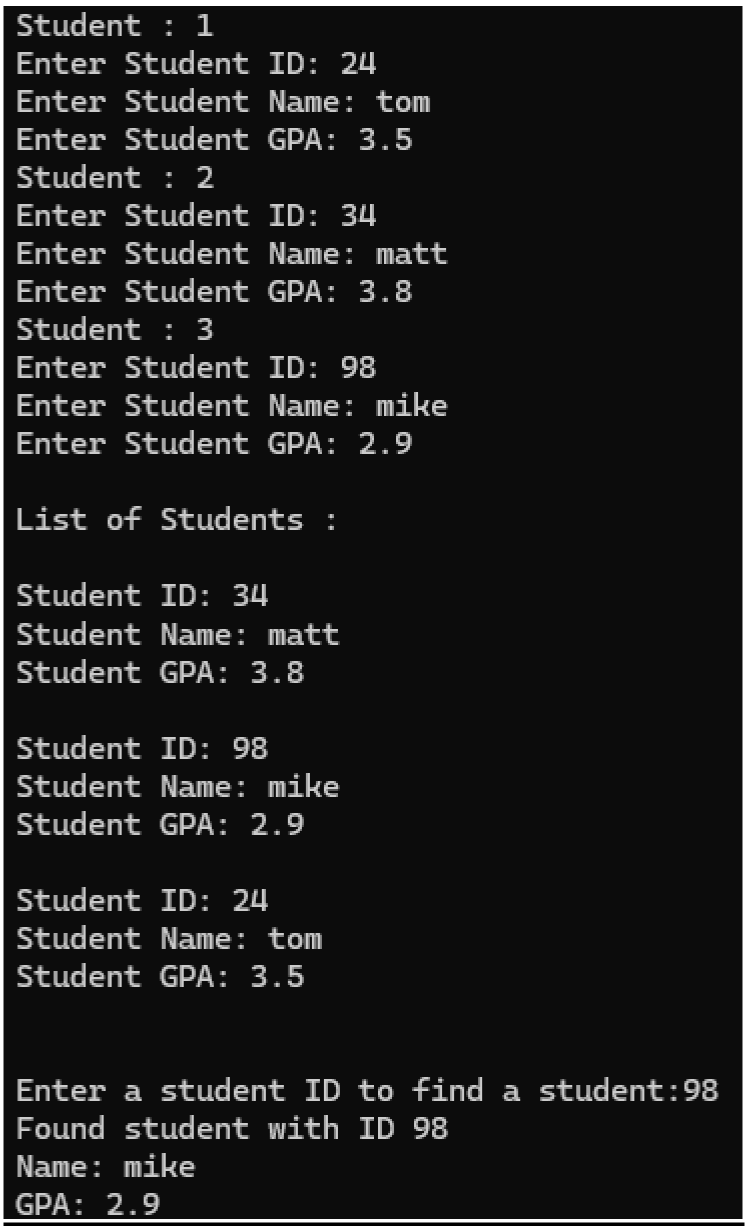
```java
import java.util.Scanner;
import java.util.HashMap;
import java.util.Map;
import java.util.Collection;

public class StudentAccess{
    public static void main (String args[]){
        Scanner sc = new Scanner(System.in);

        Map<Integer, Student> studentMap = new HashMap<>();
        for(int i = 0;i<3;i++){
            System.out.println("Student : " + (i+1));
            Student student = new Student();
            System.out.print("Enter Student ID: ");
            int id = sc.nextInt();
            sc.nextLine();
            student.setID(id);
            System.out.print("Enter Student Name: ");
            String name = sc.nextLine();
            student.setName(name);
            System.out.print("Enter Student GPA: ");
            double gpa = sc.nextDouble();
            student.setGPA(gpa);


             if (studentMap.containsKey(id)) {
                System.out.println("Student with ID " + id + " already exists, skipping student.\n");
            } else {
                studentMap.put(id, student);
            }
        }

        System.out.println("\nList of Students :\n");
        for (int id : studentMap.keySet()) {
            Student std = studentMap.get(id);
            System.out.println("Student ID: " + std.getID());
            System.out.println("Student Name: " + std.getName());
            System.out.println("Student GPA: " + std.getGPA() + "\n");
        }

        System.out.print("\nEnter a student ID to find a student:");
        int findId = sc.nextInt();
        Student foundStudent = studentMap.get(findId);

        if (foundStudent != null) {
            System.out.println("Found student with ID " + foundStudent.getID() + "\nName: " + foundStudent.getName() + "\nGPA: " + foundStudent.getGPA());
        } else {
            System.out.println("No student found with ID " + findId);
        }

    }
}
```

Student Class (Identical to previous)



```java
class Student{
    private int studentID;
    private String Name;
    private double GPA;

    public Student(){
        this.studentID = 0;
        this.Name = "name";
        this.GPA = 0.0;
    }

    public Student(int id, String name, double gpa) {
        this.studentID = id;
        this.Name = name;
        this.GPA = gpa;
    }

    public void setID(int sID){
        this.studentID = sID;
    }

    public int getID(){
        return this.studentID;
    }

    public void setName(String sName){
        this.Name = sName;
    }

    public String getName(){
        return this.Name;
    }

    public void setGPA(double sGPA){
        this.GPA = sGPA;
    }

    public double getGPA(){
        return this.GPA;
    }

}
```

Test Case



```
Student : 1
Enter Student ID: 24
Enter Student Name: tom
Enter Student GPA: 3.5
Student : 2
Enter Student ID: 34
Enter Student Name: matt
Enter Student GPA: 3.8
Student : 3
Enter Student ID: 98
Enter Student Name: mike
Enter Student GPA: 2.9

List of Students :

Student ID: 34
Student Name: matt
Student GPA: 3.8

Student ID: 98
Student Name: mike
Student GPA: 2.9

Student ID: 24
Student Name: tom
Student GPA: 3.5


Enter a student ID to find a student:98
Found student with ID 98
Name: mike
GPA: 2.9
```