Assignment 01

Problem Statement :

Create a Java program to manage a collection of books using a Set collection. Implement operations to add books, remove books by title, find books by author, and display all books in the collection.

- Define the Book Class
  Start by defining a Book class with attributes such as title, author, isbn, etc. Include appropriate constructors, getters, setters, override toString().
- Implement the Main Program (BookManager)
  Create a BookManager class to demonstrate various operations on the set of books using HashSet. Include methods public void addBook(Book book), public void removeBookByTitle(String title), public void findBooksByAuthor(String author), public void displayAllBooks) and main().

Code :

```
import java.util.HashSet;

import java.util.Set;

import java.util.stream.Collectors;

import java.util.Scanner;


class Book

{

protected String title, author;

protected int isbn, edition;

protected double price;


public Book (String title, String author, int isbn, int edition, double price)

{

this.title = title;

this.author = author;
```

```
this.isbn = isbn;
this.edition = edition;
this.price = price;
}

public String getTitle()
{
return title;
}
public void setTitle(String title)
{
this.title = title;
}

public String getAuthor()
{
return author;
}
public void setAuthor(String author)
{
this.author = author;
}

public int getIsbn()
{
return isbn;
}
public void setIsbn(int isbn)
```

```
{
this.isbn = isbn;
}

public int getEdition()
{
return edition;
}
public void setEdition(int edition)
{
this.edition = edition;
}

public double getPrice()
{
return price;
}
public void setprice(double price)
{
this.price = price;
}

@Override
public String toString()
{
return "Book {title = " + title + ", author = " + author + ", isbn = " + isbn + ", edition = " +
edition + ", price = " + price +"}";
}
```

```
@Override
public boolean equals(Object o)
{
if (this == o) return true;
if (o == null || getClass() != o.getClass()) return false;
Book book = (Book) o;
return isbn == book.isbn;
}

@Override
public int hashCode()
{
return Integer.hashCode(isbn);
}
}

public class aBookManager
{
private Set<Book> books;
public aBookManager()
{
books = new HashSet<>();
}

public void addBook(Book book)
{
if (books.add(book))
{
```

```
System.out.println("Book added: " + book);
} else
{
System.out.println("Book with ISBN " + book.getIsbn() + " already exists.");
}
}

public void removeBookByTitle(String title)
{
books.removeIf(book -> book.getTitle().equalsIgnoreCase(title));
}

public void findBooksByAuthor(String author)
{
Set<Book> booksByAuthor = books.stream()
.filter(book -> book.getAuthor().equalsIgnoreCase(author))
.collect(Collectors.toSet());
if (booksByAuthor.isEmpty())
{
System.out.println("No books found by author '" + author + "'.");
} else
{
System.out.println("Following books are found in the record : ");
booksByAuthor.forEach(System.out::println);
}
}

public void displayAllBooks()
```

```
{
books.forEach(System.out::println);
}


public static void main(String[] args)
{
aBookManager bm = new aBookManager();
Scanner sc = new Scanner(System.in);
String title, author;
int choice, isbn, edition;
double price;

do
{
System.out.print("----------------------------------------");
System.out.println("\nMenu:");
System.out.println("1. Add a Book");
System.out.println("2. Remove a Book by Title");
System.out.println("3. Find Books by Author");
System.out.println("4. Display All Books");
System.out.println("5. Exit");
System.out.print("Enter your choice: ");
choice = sc.nextInt();
sc.nextLine();
System.out.println("----------------------------------------");

switch (choice) {
```

```
case 1:
System.out.print("Enter title: ");
title = sc.nextLine();
System.out.print("Enter author: ");
author = sc.nextLine();
System.out.print("Enter ISBN: ");
isbn = sc.nextInt();
System.out.print("Enter edition: ");
edition = sc.nextInt();
System.out.print("Enter price: ");
price = sc.nextDouble();
Book book = new Book(title, author, isbn, edition, price);
bm.addBook(book);
break;

case 2:
System.out.print("Enter title to remove: ");
title = sc.nextLine();
bm.removeBookByTitle(title);
break;

case 3:
System.out.print("Enter author to find: ");
author = sc.nextLine();
bm.findBooksByAuthor(author);
break;

case 4:
```

```
System.out.println("All books in the collection:");
bm.displayAllBooks();
break;

case 5:
System.out.println("Exiting...");
break;
default:
System.out.println("Invalid choice. Please try again.");
}
}
while(choice!=5);

sc.close();
}
}
```

Output :

```
----------------------------------------
Menu:
1. Add a Book
2. Remove a Book by Title
3. Find Books by Author
4. Display All Books
5. Exit
Enter your choice: 1
----------------------------------------
Enter title: Wings of Fire
Enter author: A P J Abdul Kalam
Enter ISBN: 123
Enter edition: 1
Enter price: 249
Book added: Book {title = Wings of Fire, author = A P J Abdul Kalam, isbn = 123, edition = 1, price = 249.0}
----------------------------------------
Menu:
1. Add a Book
2. Remove a Book by Title
3. Find Books by Author
4. Display All Books
5. Exit
Enter your choice: 1
----------------------------------------
Enter title: Ignited Minds
Enter author: A P J Abdul Kalam
Enter ISBN: 1234
Enter edition: 1
Enter price: 156
Book added: Book {title = Ignited Minds, author = A P J Abdul Kalam, isbn = 1234, edition = 1, price = 156.0}
----------------------------------------
Menu:
1. Add a Book
2. Remove a Book by Title
3. Find Books by Author
4. Display All Books
5. Exit
Enter your choice: 1
----------------------------------------
Enter title: The Secret
Enter author: R R Bryne
Enter ISBN: 356
Enter edition: 2
Enter price: 100
Book added: Book {title = The Secret, author = R R Bryne, isbn = 356, edition = 2, price = 100.0}
----------------------------------------
```

```
2. Remove a Book by Title
3. Find Books by Author
4. Display All Books
5. Exit
Enter your choice: 1
----------------------------------------
Enter title: A Game of Thrones
Enter author: R R Martin
Enter ISBN: 789
Enter edition: 4
Enter price: 460
Book added: Book {title = A Game of Thrones, author = R R Martin, isbn = 789, edition = 4, price = 460.0}
----------------------------------------
Menu:
1. Add a Book
2. Remove a Book by Title
3. Find Books by Author
4. Display All Books
5. Exit
Enter your choice: 2
----------------------------------------
Enter title to remove: A Game of Thrones
----------------------------------------
Menu:
1. Add a Book
2. Remove a Book by Title
3. Find Books by Author
4. Display All Books
5. Exit
Enter your choice: 3
----------------------------------------
Enter author to find: A P J Abdul Kalam
Following books are found in the record :
Book {title = Ignited Minds, author = A P J Abdul Kalam, isbn = 1234, edition = 1, price = 156.0}
Book {title = Wings of Fire, author = A P J Abdul Kalam, isbn = 123, edition = 1, price = 249.0}
----------------------------------------
Menu:
1. Add a Book
2. Remove a Book by Title
3. Find Books by Author
4. Display All Books
5. Exit
Enter your choice: 4
----------------------------------------
All books in the collection:
Book {title = Ignited Minds, author = A P J Abdul Kalam, isbn = 1234, edition = 1, price = 156.0}
Book {title = The Secret, author = R R Bryne, isbn = 356, edition = 2, price = 100.0}
Book {title = Wings of Fire, author = A P J Abdul Kalam, isbn = 123, edition = 1, price = 249.0}
----------------------------------------
Menu:
1. Add a Book
2. Remove a Book by Title
3. Find Books by Author
4. Display All Books
5. Exit
Enter your choice: 5
----------------------------------------
Exiting...
```

Assignment 02

Problem Statement :

Implement a Java program to manage a collection of products using a Map collection. Define a Product class with attributes like id, name, price, and category. Provide operations to add products, remove products by ID, find products by name or category, and display all products in the collection.

- Define the Product Class
  Start by defining a Product class with attributes such as id, name, price, and category. Include appropriate constructors, getters, setters, toString().
- Implement the Main Program (ProductManager)
  Create a ProductManager class to demonstrate various operations on the map of products using HashMap.

Code :

```
import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;


class Product

{

protected int id;

protected String name, category;

protected double price;



public Product(int id, String name, double price, String category)

{

this.id = id;

this.name = name;
```

```
this.price = price;
this.category = category;
}

public int getId()
{
return id;
}
public void setId(int id)
{
this.id = id;
}

public String getName()
{
return name;
}
public void setName(String name)
{
this.name = name;
}

public double getPrice()
{
return price;
}
public void setPrice(int price)
{
```

```
this.price = price;
}

public String getCategory()
{
return category;
}
public void setCategory(String category)
{
this.category = category;
}

@Override
public String toString()
{
return "Product{id = " + id + ", name = " + name + ", price = " + price + ", category = " +
category +"}";
}
}

public class bProductManager
{
private Map<Integer, Product> productMap;

public bProductManager()
{
productMap = new HashMap<>();
}
```

```
public void addProduct(Product product)
{
productMap.put(product.getId(), product);
System.out.println("Product added successfully.");
}

public void removeProduct(int id)
{
if (productMap.containsKey(id))
{
productMap.remove(id);
System.out.println("Product removed successfully.");
}
else
{
System.out.println("Product with ID " + id + " not found.");
}
}

public void findProductByName(String name)
{
boolean found = false;
for (Product product : productMap.values())
{
if (product.getName().equalsIgnoreCase(name))
{
System.out.println(product);
```

```
found = true;
}
}
if (!found)
{
System.out.println("No products found with the name " + name);
}
}

public void findProductByCategory(String category)
{
boolean found = false;
for (Product product : productMap.values())
{
if (product.getCategory().equalsIgnoreCase(category))
{
System.out.println(product);
found = true;
}
}
if (!found)
{
System.out.println("No products found in the category " + category);
}
}
public void displayAllProducts()
{
if (productMap.isEmpty())
```

```
{
System.out.println("No products available.");
}
else
{
for (Product product : productMap.values())
{
System.out.println(product);
}
}
}
public static void main(String[] args)
{
bProductManager manager = new bProductManager();
Scanner sc = new Scanner(System.in);
int choice, id;
String name, category;
double price;
do
{
System.out.print("----------------------------------------");
System.out.println("\nMenu:");
System.out.println("1. Add Product");
System.out.println("2. Remove Product by ID");
System.out.println("3. Find Product by Name");
System.out.println("4. Find Product by Category");
System.out.println("5. Display All Products");
System.out.println("6. Exit");
```

```
System.out.print("Enter your choice: ");
choice = sc.nextInt();
sc.nextLine();
System.out.println("----------------------------------------");
switch (choice)
{
case 1:
System.out.print("Enter Product ID: ");
id = sc.nextInt();
sc.nextLine();
System.out.print("Enter Product Name: ");
name = sc.nextLine();
System.out.print("Enter Product Price: ");
price = sc.nextDouble();
sc.nextLine();
System.out.print("Enter Product Category: ");
category = sc.nextLine();
manager.addProduct(new Product(id, name, price, category));
break;
case 2:
System.out.print("Enter Product ID to remove: ");
int removeId = sc.nextInt();
manager.removeProduct(removeId);
break;
case 3:
System.out.print("Enter Product Name to find: ");
String findName = sc.nextLine();
manager.findProductByName(findName);
```

```
break;
case 4:
System.out.print("Enter Product Category to find: ");
String findCategory = sc.nextLine();
manager.findProductByCategory(findCategory);
break;
case 5:
manager.displayAllProducts();
break;
case 6:
System.out.println("Exiting...");
break;
default:
System.out.println("Invalid choice. Please try again.");
}
}
while (choice != 6);
sc.close();
}
}
```

Output :

```
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 1
----------------------------------------
Enter Product ID: 10
Enter Product Name: Apple
Enter Product Price: 100
Enter Product Category: Fruits
Product added successfully.
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 11
----------------------------------------
Invalid choice. Please try again.
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 1
----------------------------------------
Enter Product ID: 11
Enter Product Name: Banana
Enter Product Price: 60
Enter Product Category: Fruits
Product added successfully.
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 1
----------------------------------------
Enter Product ID: 21
Enter Product Name: Earbuds
Enter Product Price: 1500
Enter Product Category: Electronics
Product added successfully.
----------------------------------------
```

```
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 5
----------------------------------------
Product{id = 21, name = Earbuds, price = 1500.0, category = Electronics}
Product{id = 10, name = Apple, price = 100.0, category = Fruits}
Product{id = 11, name = Banana, price = 60.0, category = Fruits}
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 2
----------------------------------------
Enter Product ID to remove: 21
Product removed successfully.
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 5
----------------------------------------
Product{id = 10, name = Apple, price = 100.0, category = Fruits}
Product{id = 11, name = Banana, price = 60.0, category = Fruits}
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 3
----------------------------------------
Enter Product Name to find: Apple
Product{id = 10, name = Apple, price = 100.0, category = Fruits}
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 4
```

```
Enter your choice: 4
----------------------------------------
Enter Product Category to find: Fruits
Product{id = 10, name = Apple, price = 100.0, category = Fruits}
Product{id = 11, name = Banana, price = 60.0, category = Fruits}
----------------------------------------
Menu:
1. Add Product
2. Remove Product by ID
3. Find Product by Name
4. Find Product by Category
5. Display All Products
6. Exit
Enter your choice: 6
----------------------------------------
Exiting...
```

Assignment 03

Problem Statement :

Develop a Java program to manage student records using a Student class. Implement CRUD operations (Create, Read, Update, Delete) to add students, display all students, find a student by ID, update student details, remove a student by ID, and also provide functionality to sort students in descending order based on their marks.

- Student Class Definition
  Start by defining the Student class with attributes such as id, name, and marks. Include constructors, getters, setters, and toString() method for displaying student details.
- StudentManager Class Implementation
  Create a StudentManager class that manages student records using a List collection. Implement methods for CRUD operations (addStudent, displayAllStudents, findStudentById, updateStudent, removeStudent) and for sorting students by marks in descending order (sortStudentsByMarksDescending).

Code :

```
import java.util.Scanner;

import java.util.List;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.Iterator;


class Student

{

protected int id, marks;

protected String name;

public Student(int id, String name, int marks)

{

this.id = id;
```

```
this.name = name;
this.marks = marks;
}
public int getId()
{
return id;
}
public String getName()
{
return name;
}
public int getMarks()
{
return marks;
}
public void setName(String name)
{
this.name = name;
}
public void setMarks(int marks)
{
this.marks = marks;
}
@Override
public String toString()
{
return "Students {ID: " + id + ", Name: " + name + ", Marks: " + marks + "}";
}
```

```
}

public class cStudentManager
{
private List<Student> students;
public cStudentManager()
{
students = new ArrayList<>();
}

public void addStudent(int id, String name, int marks)
{
students.add(new Student(id, name, marks));
System.out.println("Student added successfully.");
}
public void displayAllStudents()
{
if (students.isEmpty())
{
System.out.println("No students in the list.");
}
else
{
System.out.println("List of students:");
for (Student student : students)
{
System.out.println(student);
}
```

```
}
}
public void findStudentById(int id)
{
boolean found = false;
for (Student student : students)
{
if (student.getId() == id)
{
System.out.println("Student found:");
System.out.println(student);
found = true;
break;
}
}
if (!found)
{
System.out.println("Student with ID " + id + " not found.");
}
}
public void updateStudent(int id, String newName, int newMarks)
{
boolean updated = false;
for (Student student : students)
{
if (student.getId() == id)
{
student.setName(newName);
```

```
student.setMarks(newMarks);
System.out.println("Student details updated successfully.");
updated = true;
break;
}
}
if (!updated)
{
System.out.println("Student with ID " + id + " not found. Update failed.");
}
}
public void removeStudent(int id)
{
Iterator<Student> iterator = students.iterator();
boolean removed = false;
while (iterator.hasNext())
{
Student student = iterator.next();
if (student.getId() == id)
{
iterator.remove();
System.out.println("Student removed successfully.");
removed = true;
break;
}
}
if (!removed)
{
```

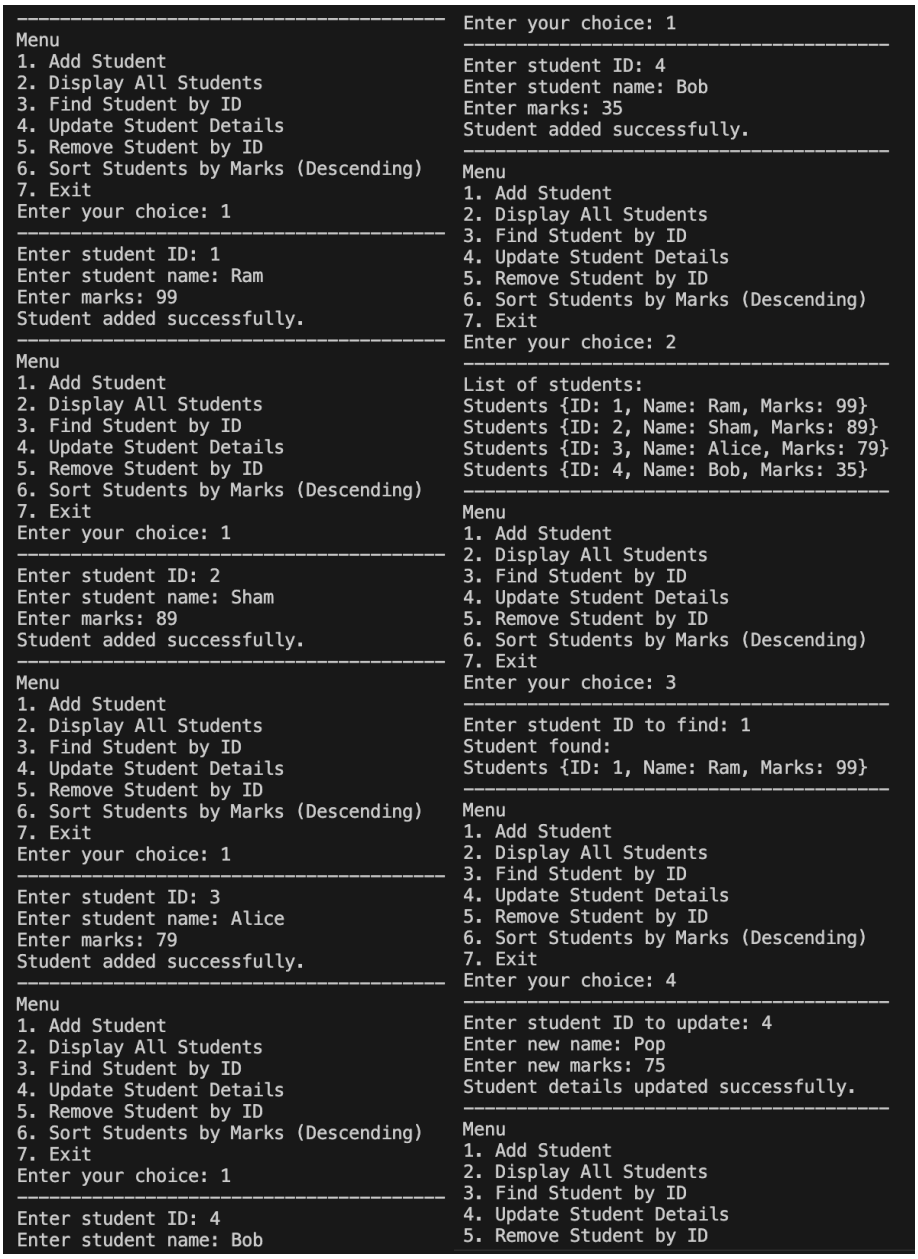
```
System.out.println("Student with ID " + id + " not found. Removal failed.");
}
}

public void sortStudentsByMarksDescending()
{
Collections.sort(students, new Comparator<Student>()
{
@Override
public int compare(Student s1, Student s2)
{
return Integer.compare(s2.getMarks(), s1.getMarks());
}
});
System.out.println("Students sorted by marks (descending order):");
displayAllStudents();
}

public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
cStudentManager manager = new cStudentManager();
int choice, id, marks;
String name;

do
{
System.out.print("----------------------------------------");
```

```
System.out.println("\nMenu");
System.out.println("1. Add Student");
System.out.println("2. Display All Students");
System.out.println("3. Find Student by ID");
System.out.println("4. Update Student Details");
System.out.println("5. Remove Student by ID");
System.out.println("6. Sort Students by Marks (Descending)");
System.out.println("7. Exit");
System.out.print("Enter your choice: ");
choice = sc.nextInt();
sc.nextLine();
System.out.println("----------------------------------------");
switch (choice)
{
case 1:
System.out.print("Enter student ID: ");
id = sc.nextInt();
sc.nextLine();
System.out.print("Enter student name: ");
name = sc.nextLine();
System.out.print("Enter marks: ");
marks = sc.nextInt();
manager.addStudent(id, name, marks);
break;

case 2:
manager.displayAllStudents();
break;
```

```
case 3:
System.out.print("Enter student ID to find: ");
int findId = sc.nextInt();
manager.findStudentById(findId);
break;

case 4:
System.out.print("Enter student ID to update: ");
int updateId = sc.nextInt();
sc.nextLine();
System.out.print("Enter new name: ");
String newName = sc.nextLine();
System.out.print("Enter new marks: ");
int newMarks = sc.nextInt();
manager.updateStudent(updateId, newName, newMarks);
break;

case 5:
System.out.print("Enter student ID to remove: ");
int removeId = sc.nextInt();
manager.removeStudent(removeId);
break;

case 6:
manager.sortStudentsByMarksDescending();
break;
```

```
case 7:
System.out.println("Exiting program...");
break;

default:
System.out.println("Invalid choice. Please enter a number between 1 and 7.");
}
}
while (choice != 7);

sc.close();
}
}
```

Output :



```
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 1
----------------------------------------
Enter student ID: 1
Enter student name: Ram
Enter marks: 99
Student added successfully.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 1
----------------------------------------
Enter student ID: 2
Enter student name: Sham
Enter marks: 89
Student added successfully.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 1
----------------------------------------
Enter student ID: 3
Enter student name: Alice
Enter marks: 79
Student added successfully.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 1
----------------------------------------
Enter student ID: 4
Enter student name: Bob
Enter marks: 35
Student added successfully.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 2
----------------------------------------
List of students:
Students {ID: 1, Name: Ram, Marks: 99}
Students {ID: 2, Name: Sham, Marks: 89}
Students {ID: 3, Name: Alice, Marks: 79}
Students {ID: 4, Name: Bob, Marks: 35}
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 3
----------------------------------------
Enter student ID to find: 1
Student found:
Students {ID: 1, Name: Ram, Marks: 99}
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 4
----------------------------------------
Enter student ID to update: 4
Enter new name: Pop
Enter new marks: 75
Student details updated successfully.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
```

```
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 2
----------------------------------------
List of students:
Students {ID: 1, Name: Ram, Marks: 99}
Students {ID: 2, Name: Sham, Marks: 89}
Students {ID: 3, Name: Alice, Marks: 79}
Students {ID: 4, Name: Pop, Marks: 75}
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 5
----------------------------------------
Enter student ID to remove: 4
Student removed successfully.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 5
----------------------------------------
Enter student ID to remove: 4
Student with ID 4 not found. Removal failed.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 2
----------------------------------------
List of students:
Students {ID: 1, Name: Ram, Marks: 99}
Students {ID: 2, Name: Sham, Marks: 89}
Students {ID: 3, Name: Alice, Marks: 79}
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
```

```
Enter student ID to remove: 4
Student removed successfully.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 5
----------------------------------------
Enter student ID to remove: 4
Student with ID 4 not found. Removal failed.
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 2
----------------------------------------
List of students:
Students {ID: 1, Name: Ram, Marks: 99}
Students {ID: 2, Name: Sham, Marks: 89}
Students {ID: 3, Name: Alice, Marks: 79}
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 6
----------------------------------------
Students sorted by marks (descending order):
List of students:
Students {ID: 1, Name: Ram, Marks: 99}
Students {ID: 2, Name: Sham, Marks: 89}
Students {ID: 3, Name: Alice, Marks: 79}
----------------------------------------
Menu
1. Add Student
2. Display All Students
3. Find Student by ID
4. Update Student Details
5. Remove Student by ID
6. Sort Students by Marks (Descending)
7. Exit
Enter your choice: 7
----------------------------------------
Exiting program...
```