Week 1:

Project 1: Basic Banking Application

Explanation

This project simulates a simple ATM where users can withdraw, deposit, and check their balance. The program validates user inputs and ensures financial operations are logically sound.

- Withdraw Function: Ensures withdrawal amount is a multiple of 100 or 500 and doesn't exceed the balance.
- Deposit Function: Allows depositing amounts, validating input.
- Balance Check: Displays the current balance.

BankingApp_Driver.java

```
import java.util.Scanner;
class BankingApp_Driver{
	public static void main(String[] args) {
	Scanner sc=new Scanner(System.in);
	boolean b=true;
	BankingApp a = new BankingApp();
	while(b)
	{
	System.out.println("Enter the option");

	System.out.println(" 1.Withdraw\n 2.Deposit\n 3.checkBalance?\n 4.Exit");
	int choice=sc.nextInt();

	switch(choice)
	{
	case 1:
		{
		System.out.println("Enter amount to withdraw");
		double amount=sc.nextDouble();
		a.withdraw(amount);
```

```
                break;
        }
        case 2:
                {
                        System.out.println("Enter the amount to deposit");
                        double amount=sc.nextDouble();
                        a.deposit(amount);
                        break;
                }
        case 3:
                {
                        a.checkBalance();
                        break;
                }
        case 4:
                {
                        b=false;
                        System.out.println("exit");
                        break;
                }
        }
        }
}
}
```

BankingApp.java

```
import java.util.Scanner;
class BankingApp
{
        double balance=20000;
        double  cur_balance;
        public  void withdraw(double amount)
```

```
{
        if(amount>=0&&amount<=balance)
        {
                if((amount%100==0)&&(amount%500==0))
                        {
                                balance=balance-amount;
                                System.out.println("Successfully withdrawn");
  }
        }
        else if (amount>balance)
                System.out.println("amount exceeded");
        else
                System.out.println("invalid amount");
}
public void deposit(double amount)
{
        if(amount>0)
        {
                if((amount%100==0)&&(amount%500==0))
                {
                balance=balance+amount;
                System.out.println("Successfully deposited");
                }
        }
        else
                System.out.println("invalid deposit");
}
public void checkBalance()
{
        System.out.println("current Balance is"+balance);
}
}
```

Tools and Technologies Used

- Java (classes & objects)
- IDE: IntelliJ IDEA

Output:

```
Enter the option
 1.Withdraw
 2.Deposit
 3.checkBalance?
 4.Exit
1
Enter amount to withdraw
500
Successfully withdrawn
Enter the option
 1.Withdraw
 2.Deposit
 3.checkBalance?
 4.Exit
3
current Balance is19500.0
Enter the option
 1.Withdraw
 2.Deposit
 3.checkBalance?
 4.Exit
2
Enter the amount to deposit
1000
Successfully deposited
```

Week 2:

Project 3: Simple Student Management System

Explanation

The program manages student records, allowing users to perform CRUD operations and sort data efficiently. The menu-driven interface ensures usability.

- Data Storage: Utilized HashMap for storing student details.
- Sorting: Implemented sorting by marks and name.

- CRUD Operations: Features to add, update, view, and delete student details.

Program:

```
import java.util.*;
class Student {
    private String name;
    private int id;
    private double marks;

    public Student(String name, int id, double marks) {
        this.name = name;
        this.id = id;
        this.marks = marks;
    }

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public double getMarks() {
        return marks;
    }

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

    public void setMarks(double marks) {
        this.marks = marks;
    }

    @Override
    public String toString() {
```

```
        return "ID: " + id + ", Name: " + name + ", Marks: " + marks;
    }
}

class StudentManagement {
    private Map<Integer, Student> students = new HashMap<>();

    public void addStudent(int id, String name, double marks) {
        if (students.containsKey(id)) {
            System.out.println("Student with this ID already exists.");
        } else {
            students.put(id, new Student(name, id, marks));
            System.out.println("Student added successfully.");
        }
    }

    public void viewStudents() {
        if (students.isEmpty()) {
            System.out.println("No students to display.");
        } else {
            for (Student student : students.values()) {
                System.out.println(student);
            }
        }
    }

    public void updateStudent(int id, String newName, double newMarks) {
        if (students.containsKey(id)) {
            Student student = students.get(id);
            student.setName(newName);
            student.setMarks(newMarks);
            System.out.println("Student updated successfully.");
        } else {
            System.out.println("Student with this ID does not exist.");
        }
    }
```

```
    public void deleteStudent(int id) {
        if (students.remove(id) != null) {
            System.out.println("Student deleted successfully.");
        } else {
            System.out.println("Student with this ID does not exist.");
        }
    }

    public void sortByName() {
        if (students.isEmpty()) {
            System.out.println("No students to sort.");
        } else {
            List<Student> sortedList = new ArrayList<>(students.values());
            sortedList.sort(Comparator.comparing(Student::getName));
            for (Student student : sortedList) {
                System.out.println(student);
            }
        }
    }

    public void sortByMarks() {
        if (students.isEmpty()) {
            System.out.println("No students to sort.");
        } else {
            List<Student> sortedList = new ArrayList<>(students.values());
            sortedList.sort(Comparator.comparingDouble(Student::getMarks).reversed());
            for (Student student : sortedList) {
                System.out.println(student);
            }
        }
    }
}

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

```
        boolean running = true;

        while (running) {
            System.out.println("\n1. Add Student\n2. View Students\n3. Update Student\n4. Delete
Student\n5. Sort by Name\n6. Sort by Marks\n7. Exit");
            int choice = sc.nextInt();
            switch (choice) {
                case 1 :{
                    System.out.println("Enter ID, Name, and Marks:");
                    int id = sc.nextInt();
                    sc.nextLine();
                    String name = sc.nextLine();
                    double marks = sc.nextDouble();
                    sm.addStudent(id, name, marks);
                    break;
                }
                case 2:
                {
                    sm.viewStudents();
                    break;}
                case 3 : {
                    System.out.println("Enter ID, New Name, and New Marks:");
                    int id = sc.nextInt();
                    sc.nextLine();
                    String name = sc.nextLine();
                    double marks = sc.nextDouble();
                    sm.updateStudent(id, name, marks);
                    break;
                }
                case 4 :{
                    System.out.println("Enter ID to delete:");
                    int id = sc.nextInt();
                    sm.deleteStudent(id);
                    break;
                }
                case 5:{sm.sortByName();break;}
                case 6: {sm.sortByMarks();break;}
```

```
            case 7: {
                running = false;
                System.out.println("Exiting...");
                break;
            }
            default:  System.out.println("Invalid option. Try again.");
        }
    }
    sc.close();
  }
}
```

Tools and Technologies Used

- Java
- Collections (HashMap, ArrayList)
- Intellij IDE

Sample Output:

```
1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Sort by Name
6. Sort by Marks
7. Exit
1
Enter ID, Name, and Marks:
101
ramesh
98
Student added successfully.

1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Sort by Name
6. Sort by Marks
7. Exit
1
Enter ID, Name, and Marks:
102
div
89
Student added successfully.
1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Sort by Name
6. Sort by Marks
7. Exit
5
ID: 102, Name: div, Marks: 89.0
ID: 101, Name: ramesh, Marks: 98.0

1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Sort by Name
6. Sort by Marks
7. Exit
7
Exiting...

Process finished with exit code 0
```

WEEK 3

Project 6: To-Do List Application

Explanation:

The To-Do List application lets users manage their tasks with functionality to persist data across sessions. It distinguishes between pending and completed tasks for clarity.

Task Management: Supports adding, editing, deleting, marking tasks complete, and viewing tasks.

File Operations: Saves and loads tasks from a file.

Task Categorization: Displays completed and pending tasks separately.

Program:

```
import jdk.jfr.Description;

import java.io.*;
import java.util.ArrayList;
import java.util.Scanner;

class Task {
    private String description;
    private boolean isComplete;

    public Task(String description) {
        this.description = description;
        this.isComplete = false;
    }

    public String getDescription() {
        return description;
    }

    public boolean isComplete() {
        return isComplete;
    }
```

```
    public void markComplete() {
        this.isComplete = true;
    }

    public void editDescription(String newDescription) {
        this.description = newDescription;
    }

    @Override
    public String toString() {
        return (isComplete ? "[X] " : "[ ] ") + description;
    }
}

public class ToDoList {
    private ArrayList<Task> tasks;
    private Scanner scanner;

    public ToDoList() {
        tasks = new ArrayList<>();
        scanner = new Scanner(System.in);
    }

    public void addTask(String description) {
        tasks.add(new Task(description));
    }

    public void editTask(int index, String newDescription) {
        if (index >= 0 && index < tasks.size()) {
            tasks.get(index).editDescription(newDescription);
        } else {
            System.out.println("Invalid task index.");
        }
    }

    public void deleteTask(int index) {
        if (index >= 0 && index < tasks.size()) {
```

```
            tasks.remove(index);
        } else {
            System.out.println("Invalid task index.");
        }
    }

    public void markTaskComplete(int index) {
        if (index >= 0 && index < tasks.size()) {
            tasks.get(index).markComplete();
        } else {
            System.out.println("Invalid task index.");
        }
    }

    public void displayTasks() {
        System.out.println("Pending Tasks:");
        for (int i = 0; i < tasks.size(); i++) {
            if (!tasks.get(i).isComplete()) {
                System.out.println(i + ": " + tasks.get(i));
            }
        }
        System.out.println("\nCompleted Tasks:");
        for (int i = 0; i < tasks.size(); i++) {
            if (tasks.get(i).isComplete()) {
                System.out.println(i + ": " + tasks.get(i));
            }
        }
    }

    public void saveTasksToFile(String filename) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
            for (Task task : tasks) {
                writer.write(task.getDescription() + "|" + (task.isComplete() ? "1" : "0"));
                writer.newLine();
            }
            System.out.println("Tasks saved to " + filename);
        } catch (IOException e) {
```

```
            System.out.println("Error saving tasks: " + e.getMessage());
        }
    }

    public void loadTasksFromFile(String filename) {
        try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split("\\|");
                Task task = new Task(parts[0]);
                if (parts[1].equals("1")) {
                    task.markComplete();
                }
                tasks.add(task);
            }
            System.out.println("Tasks loaded from " + filename);
        } catch (IOException e) {
            System.out.println("Error loading tasks: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        ToDoList todoList = new ToDoList();
        Scanner scanner = new Scanner(System.in);
        String command;

        do {
            System.out.println("\nTo-Do List Menu:");
            System.out.println("1. Add Task");
            System.out.println("2. Edit Task");
            System.out.println("3. Delete Task");
            System.out.println("4. Mark Task Complete");
            System.out.println("5. Display Tasks");
            System.out.println("6. Save Tasks to File");
            System.out.println("7. Load Tasks from File");
            System.out.println("0. Exit");
            System.out.print("Enter your choice: ");
```

```
command = scanner.nextLine();

switch (command) {
    case "1":
        System.out.print("Enter task description: ");
        String description = scanner.nextLine();
        todoList.addTask(description);
        break;
    case "2":
        System.out.print("Enter task index to edit: ");
        int editIndex = Integer.parseInt(scanner.nextLine());
        System.out.print("Enter new task description: ");
        String newDescription = scanner.nextLine();
        todoList.editTask(editIndex, newDescription); // Corrected this line
        break;
    case "3":
        System.out.print("Enter task index to delete: ");
        int deleteIndex = Integer.parseInt(scanner.nextLine());
        todoList.deleteTask(deleteIndex);
        break;
    case "4":
        System.out.print("Enter task index to mark complete: ");
        int completeIndex = Integer.parseInt(scanner.nextLine());
        todoList.markTaskComplete(completeIndex);
        break;
    case "5":
        todoList.displayTasks();
        break;
    case "6":
        System.out.print("Enter filename to save tasks: ");
        String saveFilename = scanner.nextLine();
        todoList.saveTasksToFile(saveFilename);
        break;
    case "7":
        System.out.print("Enter filename to load tasks: ");
        String loadFilename = scanner.nextLine();
        todoList.loadTasksFromFile(loadFilename);
```

```
                break;
            case "0":
                System.out.println("Exiting the application.");
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    } while (!command.equals("0"));
    scanner.close();
  }
}
```

Tools and Technologies Used

- Java
- File I/O (BufferedReader, BufferedWriter)
- IDE: Intellij

Output:

```
To-Do List Menu:
1. Add Task
2. Edit Task
3. Delete Task
4. Mark Task Complete
5. Display Tasks
6. Save Tasks to File
7. Load Tasks from File
0. Exit
Enter your choice: 1
Enter task description: Hi helloo
```

```
To-Do List Menu:
1. Add Task
2. Edit Task
3. Delete Task
4. Mark Task Complete
5. Display Tasks
6. Save Tasks to File
7. Load Tasks from File
0. Exit
Enter your choice: 1
Enter task description: welcome java

To-Do List Menu:
1. Add Task
2. Edit Task
3. Delete Task
4. Mark Task Complete
5. Display Tasks
6. Save Tasks to File
7. Load Tasks from File
0. Exit
Enter your choice: 2
Enter task index to edit: 1
Enter new task description: welcome man
```

```
To-Do List Menu:
1. Add Task
2. Edit Task
3. Delete Task
4. Mark Task Complete
5. Display Tasks
6. Save Tasks to File
7. Load Tasks from File
0. Exit
Enter your choice: 5
Pending Tasks:
0: [ ] Hi helloo
1: [ ] welcome man
```