

K.R. Mangalam University

School of Engineering & Technology

Assignment 1

Java Programming

Submitted by:

Name: Rakesh G

Roll No: 2401201064

Class: BCA (AI & DS)

Submitted to:

Dr. Manish Kumar

GitHub Repository:

https://github.com/rakesh4407/Java_Assignment_Rakesh

CODE:

```
BankingApp.java 2
JAVA > BankingApp.java > Language Support for Java(TM) by Red Hat > Account
1 // RAKESH G
2 // 240120164
3
4 import java.util.Scanner;
5
6 // Account class
7 class Account {
8
9     private int accountNumber;    Field accountNumber can be final
10    private String name, email, phone;    Field name can be final
11    private double balance;
12
13    Account(int accNo, String name, double balance, String email, String phone) {
14        this.accountNumber = accNo;
15        this.name = name;
16        this.balance = balance;
17        this.email = email;
18        this.phone = phone;
19    }
20
21    public int getAccountNumber() {
22        return accountNumber;
23    }
24
25    void deposit(double amt) {
26        if (amt > 0) {
27            balance += amt;
28            System.out.println("Deposit Successful. Balance: " + balance);
29        } else {
30            System.out.println(x:"Invalid deposit amount.");
31        }
32    }
33
34    void withdraw(double amt) {
35        if (amt > 0 && amt <= balance) {
36            balance -= amt;
37            System.out.println("Withdrawal Successful. Balance: " + balance);
38        } else {
```

```
BankingApp.java 2
J D:\RAKESH\VS\JAVA\BankingApp.java • 2 problems in this file. ) by Red Hat > Account
7     class Account {
8
9         void withdraw(double amt) {
10             System.out.println("Withdrawal Successful. Balance: " + balance);
11         } else {
12             System.out.println(x:"Invalid or insufficient balance.");
13         }
14     }
15
16     void show() {
17         System.out.println(accountNumber + " | " + name + " | " + balance + " | " + email + " | " + phone);
18     }
19
20     void update(String email, String phone) {
21         this.email = email;
22         this.phone = phone;
23         System.out.println(x:"Contact updated!");
24     }
25 }
26
27 // Main Banking Application
28 public class BankingApp {
29
30     static Scanner sc = new Scanner(System.in);
31     static Account[] accounts = new Account[100];
32     static int count = 0;
33
34     static Account find(int accNo) {
35         for (int i = 0; i < count; i++) {
36             if (accounts[i].getAccountNumber() == accNo) {
37                 return accounts[i];
38             }
39         }
40         return null;
41     }
42
43     Run main | Debug main | Run | Debug
44     public static void main(String[] args) {
45         while (true) {
46             System.out.println("1. Create 2. Deposit 3. Withdraw 4. View 5. Update 6. Exit");
47             int choice = sc.nextInt();
48             switch (choice) {
49                 case 1:
50                     System.out.println("Enter account number:");
51                     int accNo = sc.nextInt();
52                     System.out.println("Enter name:");
53                     String name = sc.next();
54                     System.out.println("Enter balance:");
55                     double balance = sc.nextDouble();
56                     System.out.println("Enter email:");
57                     String email = sc.next();
58                     System.out.println("Enter phone:");
59                     String phone = sc.next();
60                     Account acc = new Account(accNo, name, balance, email, phone);
61                     accounts[accNo] = acc;
62                     count++;
63                     break;
64                 case 2:
65                     System.out.println("Enter account number:");
66                     int accNo2 = sc.nextInt();
67                     System.out.println("Enter amount to deposit:");
68                     double amt2 = sc.nextDouble();
69                     Account acc2 = find(accNo2);
70                     if (acc2 != null) {
71                         acc2.deposit(amt2);
72                     } else {
73                         System.out.println("Account not found.");
74                     }
75                     break;
76                 case 3:
77                     System.out.println("Enter account number:");
78                     int accNo3 = sc.nextInt();
79                     System.out.println("Enter amount to withdraw:");
80                     double amt3 = sc.nextDouble();
81                     Account acc3 = find(accNo3);
82                     if (acc3 != null) {
83                         acc3.withdraw(amt3);
84                     } else {
85                         System.out.println("Account not found.");
86                     }
87                     break;
88                 case 4:
89                     System.out.println("Enter account number:");
90                     int accNo4 = sc.nextInt();
91                     Account acc4 = find(accNo4);
92                     if (acc4 != null) {
93                         acc4.show();
94                     } else {
95                         System.out.println("Account not found.");
96                     }
97                     break;
98                 case 5:
99                     System.out.println("Enter account number:");
100                    int accNo5 = sc.nextInt();
101                    System.out.println("Enter new email:");
102                    String email5 = sc.next();
103                    System.out.println("Enter new phone:");
104                    String phone5 = sc.next();
105                    Account acc5 = find(accNo5);
106                    if (acc5 != null) {
107                        acc5.update(email5, phone5);
108                    } else {
109                        System.out.println("Account not found.");
110                    }
111                    break;
112                 case 6:
113                     System.out.println("Exiting...");
114                     break;
115                 default:
116                     System.out.println("Invalid choice. Please try again.");
117             }
118         }
119     }
120 }
```

BankingApp.java 2

JAVA > BankingApp.java > Language Support for Java(TM) by Red Hat > Account

```
55 public class BankingApp {
70     public static void main(String[] args) {
73         System.out.print(s:"Enter choice: ");
74         int ch = sc.nextInt();
75         sc.nextLine();
76
77         switch (ch) {
78             case 1 -> {
79                 System.out.print(s:"Name: ");
80                 String name = sc.nextLine();
81                 System.out.print(s:"Balance: ");
82                 double bal = sc.nextDouble();
83                 sc.nextLine();
84                 System.out.print(s:"Email: ");
85                 String email = sc.nextLine();
86                 System.out.print(s:"Phone: ");
87                 String phone = sc.nextLine();
88                 accounts[count] = new Account(1000 + count + 1, name, bal, email, phone);
89                 System.out.println("Account created: " + accounts[count].getAccountNumber());
90                 count++;
91             }
92             case 2 -> {
93                 System.out.print(s:"Acc No: ");
94                 int no = sc.nextInt();
95                 System.out.print(s:"Deposit: ");
96                 double amt = sc.nextDouble();
97                 Account a = find(no);
98                 if (a != null) {
99                     a.deposit(amt);
100                 } else {
101                     System.out.println(x:"Not found.");
102                 }
103             }
104             case 3 -> {
105                 System.out.print(s:"Acc No: ");
106                 int no = sc.nextInt();
107                 System.out.print(s:"Withdraw: ");
108                 double amt = sc.nextDouble();
```

BankingApp.java 2

JAVA > BankingApp.java > Language Support for Java(TM) by Red Hat > Account

```
55 public class BankingApp {
70     public static void main(String[] args) {
107         System.out.print(s: "Withdraw. ");
108         double amt = sc.nextDouble();
109         Account a = find(no);
110         if (a != null) {
111             a.withdraw(amt);
112         } else {
113             System.out.println(x: "Not found.");
114         }
115     }
116     case 4 -> {
117         System.out.print(s: "Acc No: ");
118         int no = sc.nextInt();
119         Account a = find(no);
120         if (a != null) {
121             a.show();
122         } else {
123             System.out.println(x: "Not found.");
124         }
125     }
126     case 5 -> {
127         System.out.print(s: "Acc No: ");
128         int no = sc.nextInt();
129         sc.nextLine();
130         System.out.print(s: "New Email: ");
131         String email = sc.nextLine();
132         System.out.print(s: "New Phone: ");
133         String phone = sc.nextLine();
134         Account a = find(no);
135         if (a != null) {
136             a.update(email, phone);
137         } else {
138             System.out.println(x: "Not found.");
139         }
140     }
141     case 6 -> {
142         System.out.println(x: "Exiting. Thank you!");
143         return;
```

```
BankingApp.java 2
JAVA > BankingApp.java > Language Support for Java(TM) by Red Hat > Account
55 public class BankingApp {
70     public static void main(String[] args) {
115     }
116     case 4 -> {
117         System.out.print(s:"Acc No: ");
118         int no = sc.nextInt();
119         Account a = find(no);
120         if (a != null) {
121             a.show();
122         } else {
123             System.out.println(x:"Not found.");
124         }
125     }
126     case 5 -> {
127         System.out.print(s:"Acc No: ");
128         int no = sc.nextInt();
129         sc.nextLine();
130         System.out.print(s:"New Email: ");
131         String email = sc.nextLine();
132         System.out.print(s:"New Phone: ");
133         String phone = sc.nextLine();
134         Account a = find(no);
135         if (a != null) {
136             a.update(email, phone);
137         } else {
138             System.out.println(x:"Not found.");
139         }
140     }
141     case 6 -> {
142         System.out.println(x:"Exiting. Thank you!");
143         return;
144     }
145     default ->
146         System.out.println(x:"Invalid choice!");
147 }
148 }
149 }
150 }
```

OUTPUT:

```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE SPELL CHECKER 1

1.Create 2.Deposit 3.Withdraw 4.View 5.Update 6.Exit
Enter choice: 1
Name: RAKESH
Balance: 1000
Email: 2401201064@gmail.com
Phone: 8610086100
Account created: 1001

1.Create 2.Deposit 3.Withdraw 4.View 5.Update 6.Exit
Enter choice: 6
Exiting. Thank you!

D:\RAKESH\VSC>JAVA ASSINGMENT 1
```

Explanation of the Code

1. Package Creation

```
java Copy code

package studentmanagement;
```

- The program is organized into a package named `studentmanagement`.
- Using a package helps keep related classes together and maintain better project structure.

2. Interface – `RecordActions`

```
java Copy code

interface RecordActions {
    void addStudent();
    void displayStudent();
}
```

- This interface defines two abstract methods:
 - `addStudent()` → To add student details.
 - `displayStudent()` → To display student details.
- The interface ensures that any class implementing it will provide these functionalities.



3. Custom Exception – StudentNotFoundException

```
java Copy code

class StudentNotFoundException extends Exception {
    public StudentNotFoundException(String message) {
        super(message);
    }
}
```

- A **custom checked exception** that handles cases where a student record is not found.
- It extends the `Exception` class, allowing developers to throw meaningful error messages.

4. Thread Class – Loader

```
java Copy code

class Loader implements Runnable {
    public void run() {
        try {
            System.out.print("Loading");
            for (int i = 0; i < 5; i++) {
                Thread.sleep(500);
                System.out.print(".");
            }
            System.out.println();
        } catch (InterruptedException e) {
            System.out.println("Loading interrupted!");
        }
    }
}
```

Key Concepts Used

Concept	Description
Exception Handling	Managed using try-catch-finally blocks and custom exceptions.
Multithreading	Implemented via Runnable interface (Loader class).
Wrapper Classes	Used Integer and Double for type conversion and autoboxing.

Concept	Description
Data Validation	Ensures input marks are within a valid range.
User Interaction	Takes input dynamically using Scanner.

Conclusion

This program efficiently manages student data using object-oriented concepts, exception handling, wrapper classes, and multithreading.

It ensures that user inputs are validated, errors are handled gracefully, and the program remains responsive.