

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
55  public class Bankingapp {
70      public static void main(String[] args) {
71          System.out.print(s:"Enter choice: ");
72          int ch = sc.nextInt();
73          sc.nextLine();
74
75          switch (ch) {
76              case 1 -> {
77                  System.out.print(s:"Name: ");
78                  String name = sc.nextLine();
79                  System.out.print(s:"Balance: ");
80                  double bal = sc.nextDouble();
81                  sc.nextLine();
82                  System.out.print(s:"Email: ");
83                  String email = sc.nextLine();
84                  System.out.print(s:"Phone: ");
85                  String phone = sc.nextLine();
86                  accounts[count] = new Account(1000 + count + 1, name, bal, email, phone);
87                  System.out.println("Account created: " + accounts[count].getAccountNumber());
88                  count++;
89              }
90              case 2 -> {
91                  System.out.print(s:"Acc No: ");
92                  int no = sc.nextInt();
93                  System.out.print(s:"Deposit: ");
94                  double amt = sc.nextDouble();
95                  Account a = find(no);
96                  if (a != null) {
97                      a.deposit(amt);
98                  } else {
99                      System.out.println(x:"Not found.");
100                 }
101             }
102             case 3 -> {
103                 System.out.print(s:"Acc No: ");
104                 int no = sc.nextInt();
105                 System.out.print(s:"Withdraw: ");
106                 double amt = sc.nextDouble();
107                 a.withdraw(amt);
108             }
109         }
110     }
111 }
```

```
BankingApp.java 2 ●
JAVA > BankingApp.java > Language Support for Java(TM) by Red Hat > Account
55  public class BankingApp {
56      public static void main(String[] args) {
57          System.out.print(s: "Withdraw. ");
58          double amt = sc.nextDouble();
59          Account a = find(no);
60          if (a != null) {
61              a.withdraw(amt);
62          } else {
63              System.out.println(x:"Not found.");
64          }
65      }
66      case 4 -> {
67          System.out.print(s:"Acc No: ");
68          int no = sc.nextInt();
69          Account a = find(no);
70          if (a != null) {
71              a.show();
72          } else {
73              System.out.println(x:"Not found.");
74          }
75      }
76      case 5 -> {
77          System.out.print(s:"Acc No: ");
78          int no = sc.nextInt();
79          sc.nextLine();
80          System.out.print(s:"New Email: ");
81          String email = sc.nextLine();
82          System.out.print(s:"New Phone: ");
83          String phone = sc.nextLine();
84          Account a = find(no);
85          if (a != null) {
86              a.update(email, phone);
87          } else {
88              System.out.println(x:"Not found.");
89          }
90      }
91      case 6 -> {
92          System.out.println(x:"Exiting. Thank you!");
93          return;
94      }
95  }
```

```
BankingApp.java 2 ●
JAVA > BankingApp.java > Language Support for Java(TM) by Red Hat > Account
55  public class BankingApp {
56      public static void main(String[] args) {
57          Scanner sc = new Scanner(System.in);
58
59          int choice;
60
61          do {
62              System.out.print("Enter your choice: ");
63              choice = sc.nextInt();
64
65              switch (choice) {
66                  case 1 -> {
67                      System.out.print("Enter Account No: ");
68                      int no = sc.nextInt();
69
70                      Account a = find(no);
71
72                      if (a != null) {
73                          a.show();
74                      } else {
75                          System.out.println("Not found.");
76                      }
77                  }
78
79                  case 2 -> {
80                      System.out.print("Enter Account No: ");
81                      int no = sc.nextInt();
82
83                      System.out.print("Enter New Email: ");
84                      String email = sc.nextLine();
85
86                      System.out.print("Enter New Phone: ");
87                      String phone = sc.nextLine();
88
89                      Account a = find(no);
90
91                      if (a != null) {
92                          a.update(email, phone);
93                      } else {
94                          System.out.println("Not found.");
95                      }
96                  }
97
98                  case 3 -> {
99                      System.out.print("Enter Account No: ");
100                     int no = sc.nextInt();
101
102                     Account a = find(no);
103
104                     if (a != null) {
105                         a.show();
106                     } else {
107                         System.out.println("Not found.");
108                     }
109                 }
110
111                 case 4 -> {
112                     System.out.print("Enter Account No: ");
113                     int no = sc.nextInt();
114
115                     Account a = find(no);
116
117                     if (a != null) {
118                         a.show();
119                     } else {
120                         System.out.println("Not found.");
121                     }
122                 }
123
124                 case 5 -> {
125                     System.out.print("Enter Account No: ");
126                     int no = sc.nextInt();
127
128                     System.out.print("Enter New Email: ");
129                     String email = sc.nextLine();
130
131                     System.out.print("Enter New Phone: ");
132                     String phone = sc.nextLine();
133
134                     Account a = find(no);
135
136                     if (a != null) {
137                         a.update(email, phone);
138                     } else {
139                         System.out.println("Not found.");
140                     }
141                 }
142
143                 case 6 -> {
144                     System.out.println("Exiting. Thank you!");
145                     return;
146                 }
147
148                 default -> {
149                     System.out.println("Invalid choice!");
150                 }
151             }
152         }
153     }
154 }
```

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.