

**Texas College of Management & IT**  
**Bachelor of Information Technology (BIT)**

**Project Title:** Fun in Programming with Java and OOP

**Submitted By:**

Name: Anuraj Gautam

LCID: LC00017003334

Semester: 2<sup>nd</sup> Semester, 1<sup>st</sup> year

**Submitted To:**

Instructor Name: Rajan Yadav

Course: Java Programming

Date of Submission :2025/07/19

---

Texas College of Management &IT, Kathmandu

---

## Table of Contents

1. Introduction .....	3
2. Objects.....	3
3. Task 1: OOP implementation -Banking Systems .....	4-6
4. Task 2: inheritance & Polymorphism -Employee Management.....	7-9
5. Task 3: String and Exception Handling -Student Grading .....	10-11
6. Task 4: Java Collections -Library Management System.....	12-13
7. Task 5: Design Patterns – Online food Delivery App.....	14-16
8. Conclusion.....	17
9. References .....	17
10. Declaration of Originality.....	18

## **Introduction:**

Briefly introduce the importance of Flutter for mobile development and Java for backend logic. Mention the objective of combining both for cross-platform app development and logic modeling. Flutter is a powerful open-source UI toolkit by Google used to build natively mobile, web, and desktop applications from a single codebase. It enables fast development with expressive and flexible UI designs. On the other hand, Java is a backend programming language known for its portability and object-oriented capabilities.

## **Objects:**

- To improve foundation of Java programming.
- Help in building real-world projects.
- To represent real-life entities in code like Car, Student, Employee.
- To allow easy reuse of code through object instances.
- Make programs easier to understand and maintain.

## Task 1: OOP implementation -Banking Systems

```
Bank.java X
C: > Users > gauta > OneDrive > Desktop > java class > Bank.java

1  /* Banking System */
2
3  class Customer {
4      private String account_name;
5      private String address;
6
7      public Customer(String account_name, String address) {
8          this.account_name = account_name;
9          this.address = address;
10     }
11
12     public String getAccount_name() {
13         return account_name;
14     }
15
16     public String getAddress() {
17         return address;
18     }
19
20
21     public void displayCustomerDetails() {
22         System.out.println("Account Name: " + account_name);
23         System.out.println("Address: " + address);
24     }
25 }
26
27 class Account {
28     private int account_number;
29     private double balance;
30 }
```

```

    public Account(int account_number, double balance) {
        this.account_number = account_number;
        this.balance = balance;
    }

    public void displayAccount() {
        System.out.println("Account No: " + account_number);
        System.out.println("Balance: Rs " + balance);
    }

    @Override
    protected void finalize(){
        System.out.println("Account object is being deleted");
    }
}

public class Bank {
    public static void main(String[] args) {
        Customer c1 = new Customer("Anuraj", "Kavreplanchowk");
        Account a1 = new Account(101101, 5000);

        a1.displayAccount();
        c1.displayCustomerDetails();
    }
}

```

## Output:

```
Account No: 101101  
Balance: Rs 5000.0  
Account Name: Anuraj  
Address: Kavreplanchowk  
PS C:\Users\gauta\OneDrive\Desktop\java class>
```

## Task 2: inheritance & Polymorphism -Employee Management

```
Employees.java X
Employees.java > Employee
1  /* Employee management system*/
2  interface TaxPayer {
3      double calculateTax();
4  }
5
6  abstract class Employee {
7      String name;
8      double salary;
9
10     public Employee(String name, double salary) {
11         this.name = name;
12         this.salary = salary;
13     }
14
15     abstract void work();
16
17     void showDetails() {
18         System.out.println("Name: " + name + ", Salary: " + salary);
19     }
20 }
21
22 class Manager extends Employee implements TaxPayer {
23     public Manager(String name, double salary) {
24         super(name, salary);
25     }
26
27     @Override
28     void work() {
29         System.out.println(name + " is manage team.");
30     }
31
32     public double calculateTax() {
33         return salary * 0.2;
34     }
35 }
36
37 class Developer extends Employee implements TaxPayer {
38     public Developer(String name, double salary) {
39         super(name, salary);
40     }
41
42     @Override
43     void work() {
44         System.out.println(name + " is write code.");
45     }
46 }
```

```

47     public double calculateTax() {
48         return salary * 0.15;
49     }
50 }
51
52 class Intern extends Employee {
53     public Intern(String name, double salary) {
54         super(name, salary);
55     }
56
57     @Override
58     void work() {
59         System.out.println(name + " is learning and assisting.");
60     }
61     public double calculateTax() {
62         return salary * 0.1;
63     }
64 }
65
66 public class Employees{
67     Run | Debug
68     public static void main(String[] args) {
69         Employee emp1 = new Manager(name:"Anuraj", salary:80000);
70         Employee emp2 = new Developer(name:"Gautam", salary:60000);
71         Employee emp3 = new Intern(name:"Chhetri", salary:20000);
72
73         emp1.work();
74         emp2.work();
75         emp3.work();
76
77         if (emp1 instanceof Manager ) {
78             Manager M= ( Manager) emp1;
79             System.out.println("Tax for manager: " + M.calculateTax());
80         }
81         if (emp2 instanceof Developer) {
82             Developer D= (Developer) emp2;
83             System.out.println("Tax for developer: " + D.calculateTax());
84         }
85         if (emp3 instanceof Intern) {
86             Intern I= (Intern) emp3;
87             System.out.println("Tax for intern: " + I.calculateTax());
88         }
89     }
90 }

```

Output:



```
PS C:\Users\gauta\OneDrive\Desktop\java class> cd "C:\
Anuraj is manage team.
Gautam is write code.
Chhetri is learning and assisting.
Tax for manager: 16000.0
Tax for developer: 9000.0
Tax for intern: 2000.0
PS C:\Users\gauta\OneDrive\Desktop\java class>
```

### Task 3: String and Exception Handling -Student Grading

```
Gradingsystem.java X
Gradingsystem.java > ...
1  import java.util.Scanner;
2
3  class InvalidMarkException extends Exception {
4      public InvalidMarkException(String message) {
5          super(message);
6      }
7  }
8
9  public class Gradingsystem {
    Run | Debug
10     public static void main(String[] args) {
11         Scanner sc = new Scanner(System.in);
12
13         try {
14             System.out.print(s:"Enter student name: ");
15             String name = sc.nextLine();
16
17             System.out.print(s:"Enter marks (0-100): ");
18             String markStr = sc.nextLine();
19
20             int marks = Integer.parseInt(markStr);
21             if (marks < 0 || marks > 100) {
22                 throw new InvalidMarkException(message:"Marks must be between 0 and 100.");
23             }
24
25             String grade = getGrade(marks);
26             System.out.println("Student: " + name.toUpperCase());
27             System.out.println("Grade: " + grade);
28
29             if (grade.equals(anObject:"Fail")) {
30                 System.out.println(x:" You have failed, Please try again!");
31             }
32
33         } catch (InvalidMarkException e) {
34             System.out.println("Error: " + e.getMessage());
35         } catch (NumberFormatException e) {
36             System.out.println(x:"Invalid input! Please enter numeric marks.");
37         } finally {
38             System.out.println(x:"Grading operation completed.");
39             sc.close();
40         }
41     }
42
43     static String getGrade(int marks) {
44         if (marks >= 90) return "A";
45         else if (marks >= 75) return "B";
46         else if (marks >= 60) return "C";
47         else if (marks >= 40) return "D";
48         else return "Fail";
49     }
50 }
51
```

Output:

```
PS C:\Users\gauta\OneDrive\Desktop\java class> cd "c:\Users\gauta\OneDrive\Desktop\java class\" ; if ($?) { javac GradingSystem.java
Enter student name: Chhetri
Enter marks (0-100): 65
Student: CHHETRI
Grade: C
Grading operation completed.
PS C:\Users\gauta\OneDrive\Desktop\java class> cd "c:\Users\gauta\OneDrive\Desktop\java class\" ; if ($?) { javac GradingSystem.java
Enter student name: Roshan
Enter marks (0-100): 75
Student: ROSHAN
Grade: B
Grading operation completed.
PS C:\Users\gauta\OneDrive\Desktop\java class> cd "c:\Users\gauta\OneDrive\Desktop\java class\" ; if ($?) { javac GradingSystem.java
Enter student name: Anuraj
Enter marks (0-100): 97
Student: ANURAJ
Grade: A
Grading operation completed.
PS C:\Users\gauta\OneDrive\Desktop\java class> cd "c:\Users\gauta\OneDrive\Desktop\java class\" ; if ($?) { javac GradingSystem.java
Enter student name: Gautam
Enter marks (0-100): 30
Student: GAUTAM
Grade: Fail
You have failed, Please try again!
Grading operation completed.
PS C:\Users\gauta\OneDrive\Desktop\java class>
```

#### Task 4: Java Collections -Library Management System

```

LibraryManagementSystem.java x
LibraryManagementSystem.java > LibraryManagementSystem > main(String[])
1  import java.util.*;
2
3
4  class Book {
5      String title;
6      String author;
7
8      public Book(String title, String author) {
9          this.title = title;
10         this.author = author;
11     }
12
13     @Override
14     public String toString() {
15         return title + " by " + author;
16     }
17 }
18
19 public class LibraryManagementSystem {
20
21     Run | Debug
22     public static void main(String[] args) {
23
24         List<Book> allBooks = new ArrayList<>();
25         allBooks.add(new Book(title:"Java Basics", author:"James Gosling"));
26         allBooks.add(new Book(title:"Effective Java", author:"Joshua Bloch"));
27         allBooks.add(new Book(title:"Clean Code", author:"Robert C. Martin"));
28
29
30         Set<Book> borrowedBooks = new HashSet<>();
31         borrowedBooks.add(allBooks.get(index:1));
32         borrowedBooks.add(allBooks.get(index:2));
33
34         Map<String, List<Book>> memberBooksMap = new HashMap<>();
35         memberBooksMap.put(key:"M001", Arrays.asList(allBooks.get(index:1)));
36         memberBooksMap.put(key:"M002", Arrays.asList(allBooks.get(index:2)));
37
38
39         System.out.println(x:"All Books:");
40         for (Book book : allBooks) {
41             System.out.println(book);
42         }
43
44
45         System.out.println(x:"\nBorrowed Books:");
46         Iterator<Book> it = borrowedBooks.iterator();
47         while (it.hasNext()) {
48             System.out.println(it.next());
49         }
50
51         System.out.println(x:"\nMember Borrowed Books:");
52         for (Map.Entry<String, List<Book>> entry : memberBooksMap.entrySet()) {
53             System.out.println("Member ID: " + entry.getKey());
54             for (Book book : entry.getValue()) {
55                 System.out.println("  -> " + book);
56             }
57         }
58         displayBookTitles(allBooks);
59
60

```

## Output:

```
PS C:\Users\gauta\OneDrive\Desktop\java class> cd "c:\Users\gauta\OneDrive\Desktop\java class\" ; if ($?) { javac LibraryManag
● All Books:
  Java Basics by James Gosling
  Effective Java by Joshua Bloch
  Clean Code by Robert C. Martin

  Borrowed Books:
  Effective Java by Joshua Bloch
  Clean Code by Robert C. Martin

  Member Borrowed Books:
  Member ID: M002
    -> Clean Code by Robert C. Martin
  Member ID: M001
    -> Effective Java by Joshua Bloch

  Book Titles:
  Java Basics
  Effective Java
  Clean Code
○ PS C:\Users\gauta\OneDrive\Desktop\java class>
```

## Task 5: Design Patterns – Online food Delivery App

```
FoodDeliveryapp.java X
FoodDeliveryapp.java > ...
1  import java.util.*;
2  class Database {
3      private static Database instance;
4
5      private Database() {
6          System.out.println("Connecting to the central database...");
7      }
8
9      public static synchronized Database getInstance() {
10         if (instance == null) {
11             instance = new Database();
12         }
13         return instance;
14     }
15
16     public void query(String sql) {
17         System.out.println("Executing query: " + sql);
18     }
19 }
20
21
22 abstract class User {
23     protected String name;
24     public User(String name) { this.name = name; }
25     abstract void showRole();
26 }
27
28 class Customer extends User {
29     public Customer(String name) { super(name); }
30     void showRole() {
31         System.out.println("Hello " + name + ", you are a Customer.");
32     }
33 }
34
35 class DeliveryPerson extends User {
36     public DeliveryPerson(String name) { super(name); }
37     void showRole() {
38         System.out.println("Hello " + name + ", you are a Delivery Person.");
39     }
40 }
41
42 class Admin extends User {
43     public Admin(String name) { super(name); }
44     void showRole() {
45         System.out.println("Hello " + name + ", you are an Admin.");
46     }
47 }
48
49 class UserFactory {
50     public static User createUser(String role, String name) {
51         switch (role.toLowerCase()) {
52             case "customer": return new Customer(name);
53             case "delivery": return new DeliveryPerson(name);
54             case "admin": return new Admin(name);
55             default: throw new IllegalArgumentException("Invalid role!");
56         }
57     }
58 }
```

```

57 }
58 }
59
60
61 interface Observer {
62     void update(String status);
63 }
64
65 class CustomerObserver implements Observer {
66     private String name;
67
68     public CustomerObserver(String name) {
69         this.name = name;
70     }
71
72     public void update(String status) {
73         System.out.println("Customer " + name + ": Order status updated -> " + status);
74     }
75 }
76
77 class DeliveryObserver implements Observer {
78     private String name;
79
80     public DeliveryObserver(String name) {
81         this.name = name;
82     }
83
84     public void update(String status) {
85         System.out.println("Delivery Person " + name + ": Order status updated -> " + status);
86     }
87 }
88
89 class OrderStatus {
90     private List<Observer> observers = new ArrayList<>();
91     private String status;
92
93     public void attach(Observer obs) {
94         observers.add(obs);
95     }
96
97     public void setStatus(String newStatus) {
98         this.status = newStatus;
99         notifyAllObservers();
100     }
101
102     private void notifyAllObservers() {
103         for (Observer obs : observers) {
104             obs.update(status);
105         }
106     }
107 }
108
109 public class FoodDeliveryapp {
110     Run | Debug
111     public static void main(String[] args) {

```

## Output:

```
PS C:\Users\gauta\OneDrive\Desktop\java class> cd "c:\Users\gauta\OneDrive\Desktop\java class\" ; if ($?) { javac FoodDeliveryapp.java }
Connecting to the central database...
Executing query: SELECT * FROM orders
Hello Anuraj, you are a Customer.
Hello Gautam, you are a Delivery Person.
Customer Anuraj: Order status updated -> Preparing
Delivery Person Gautam: Order status updated -> Preparing
Customer Anuraj: Order status updated -> Out for Delivery
Delivery Person Gautam: Order status updated -> Out for Delivery
Customer Anuraj: Order status updated -> Delivered
Delivery Person Gautam: Order status updated -> Delivered
PS C:\Users\gauta\OneDrive\Desktop\java class>
```

## Class Diagram

Database

-Instances

-Database ()

+getInstances

+query

## Benefits

- Adds flexibility to user role creation.
- Easy to extend (e.g. add Admin, Restaurant Owner)

## Limitations

- Requires code changes when new roles are added.
- Hard to manage if many subclasses are needed.



## **Conclusion**

This project highlights the power of java Object-Oriented Programming in modeling real world systems. Each task demonstrates core concepts like encapsulation, inheritance, polymorphism, collection, and design patterns in practical scenarios. These concepts strengthen logical thinking, improve projects structure.

## **Reference:**

This project is based on personal analysis and understanding of real-world platform challenges related to trust. For technical concepts and implementation resources,

- W3School
- Lecture notes and class assignments
- Personal experiment and testing environment.
- YouTube video-codewithmosh, codewithharry Apna College etc.

**Declaration of Originality**

I hereby declare that this assignment is my original hard work, and I have referenced all sources as required. I understand that failure to adhere to academic integrity will in disciplinary action.

Student Name: Anuraj Gautam

Student Id: LC00017003334

Signature: 

Date :2025/07/19