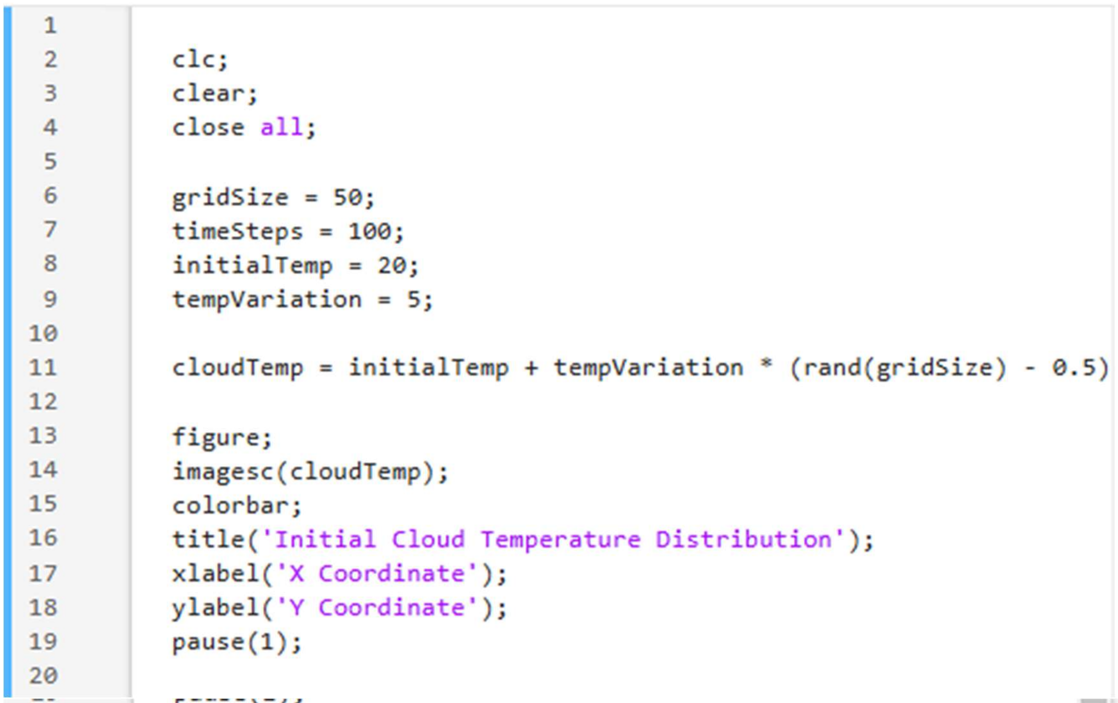
Experiment 1.2

Student Name: Shubham Ranjan **UID: 22BCS16090**
Branch: CSE **Section: IOT – 640(A)**
Semester: 6th **DOP: /2025**
Subject: IOT ML Lab **Subject Code: 22CSP-367**

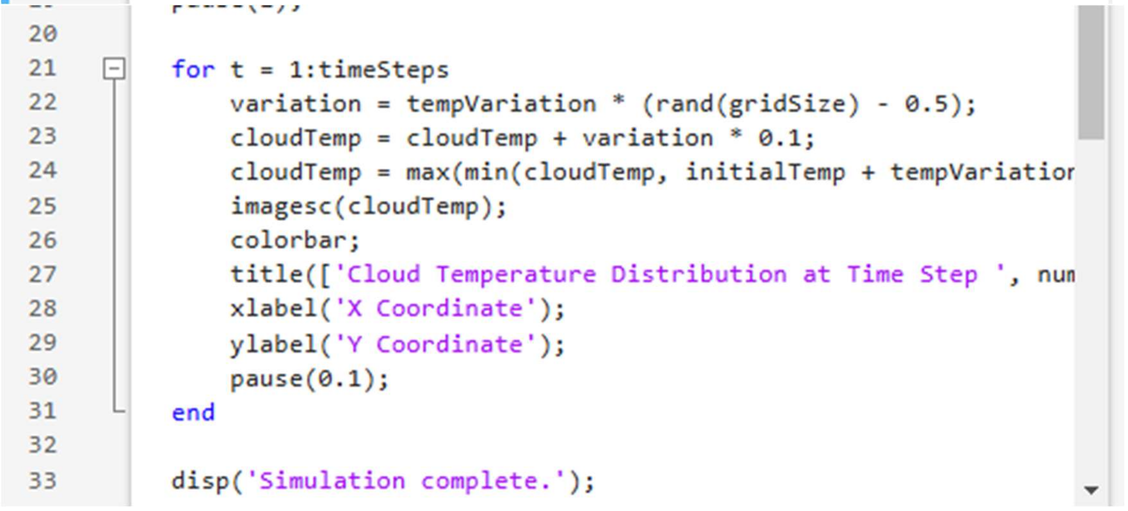
Aim: Simulate a cloud scenario using Matlab and run an algorithm for temperature variations

Objective: The objective of this experiment is to simulate a cloud scenario using MATLAB and run an algorithm to model temperature variations within that environment.

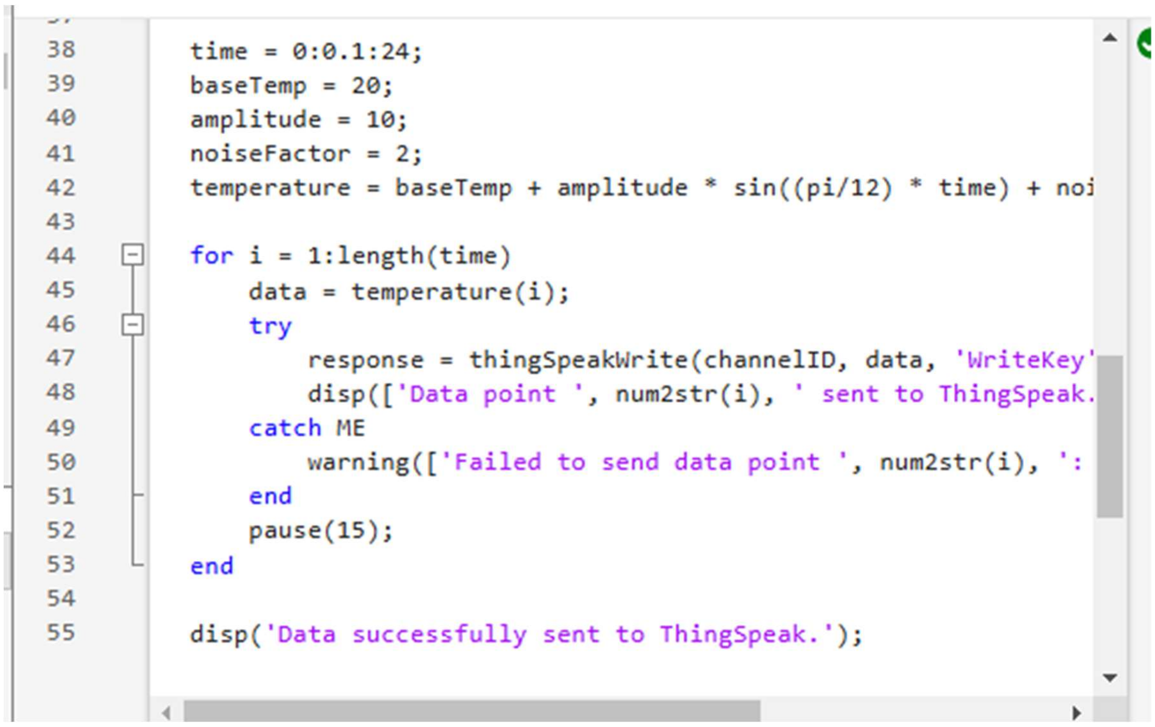
Script:



```
clc;
clear;
close all;

gridSize = 50;
timeSteps = 100;
initialTemp = 20;
tempVariation = 5;

cloudTemp = initialTemp + tempVariation * (rand(gridSize) - 0.5)

figure;
imagesc(cloudTemp);
colorbar;
title('Initial Cloud Temperature Distribution');
xlabel('X Coordinate');
ylabel('Y Coordinate');
pause(1);
```

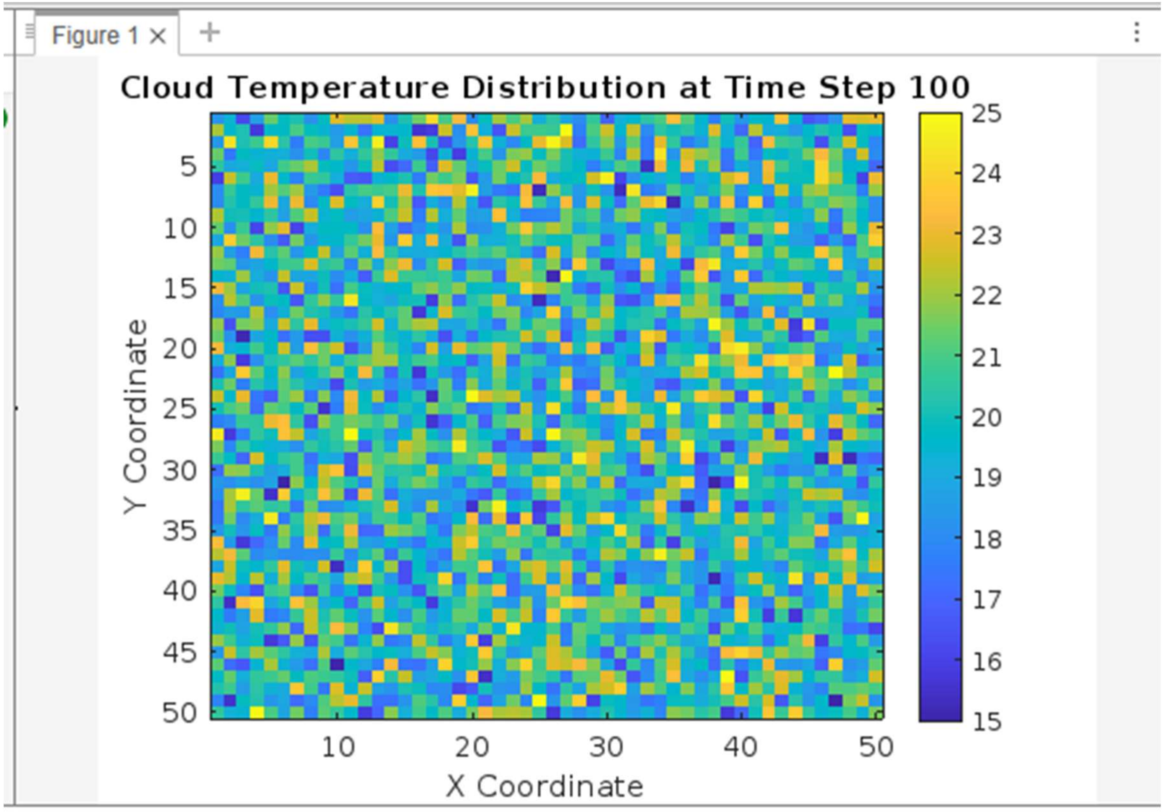


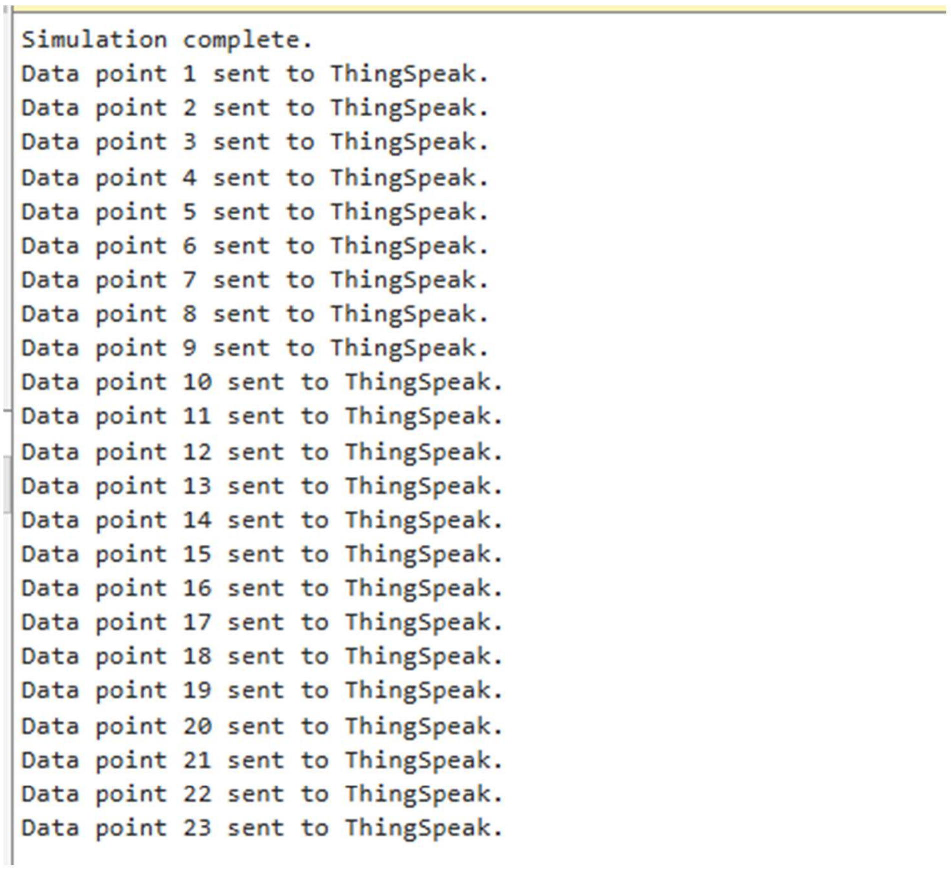
```

for t = 1:timeSteps
    variation = tempVariation * (rand(gridSize) - 0.5);
    cloudTemp = cloudTemp + variation * 0.1;
    cloudTemp = max(min(cloudTemp, initialTemp + tempVariatior
    imagesc(cloudTemp);
    colorbar;
    title(['Cloud Temperature Distribution at Time Step ', nun
    xlabel('X Coordinate');
    ylabel('Y Coordinate');
    pause(0.1);
end

disp('Simulation complete.');
```



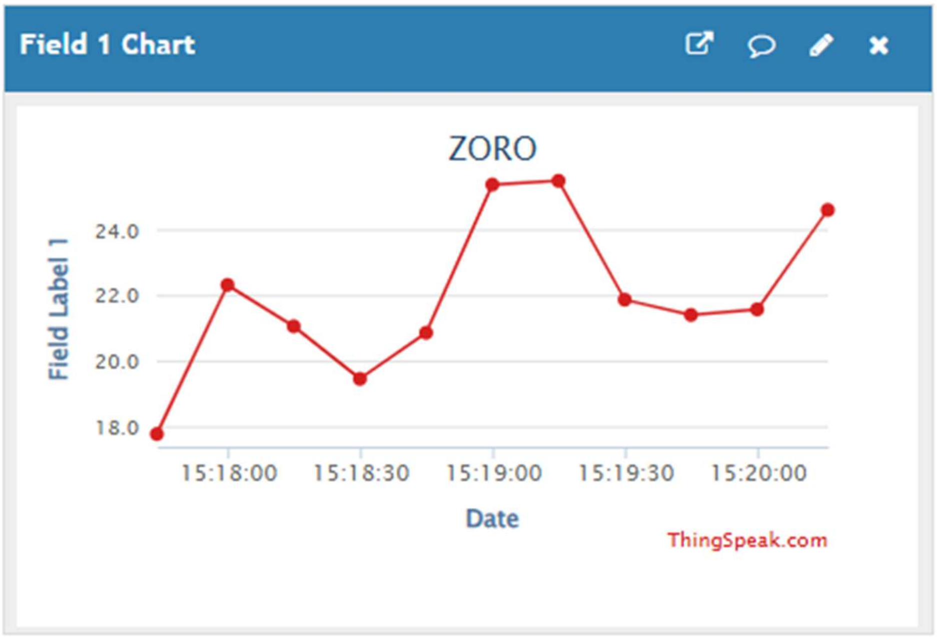
```
time = 0:0.1:24;
baseTemp = 20;
amplitude = 10;
noiseFactor = 2;
temperature = baseTemp + amplitude * sin((pi/12) * time) + noi

for i = 1:length(time)
    data = temperature(i);
    try
        response = thingSpeakWrite(channelID, data, 'WriteKey'
        disp(['Data point ', num2str(i), ' sent to ThingSpeak.
    catch ME
        warning(['Failed to send data point ', num2str(i), ':
    end
    pause(15);
end

disp('Data successfully sent to ThingSpeak.');
```

OUTPUT:

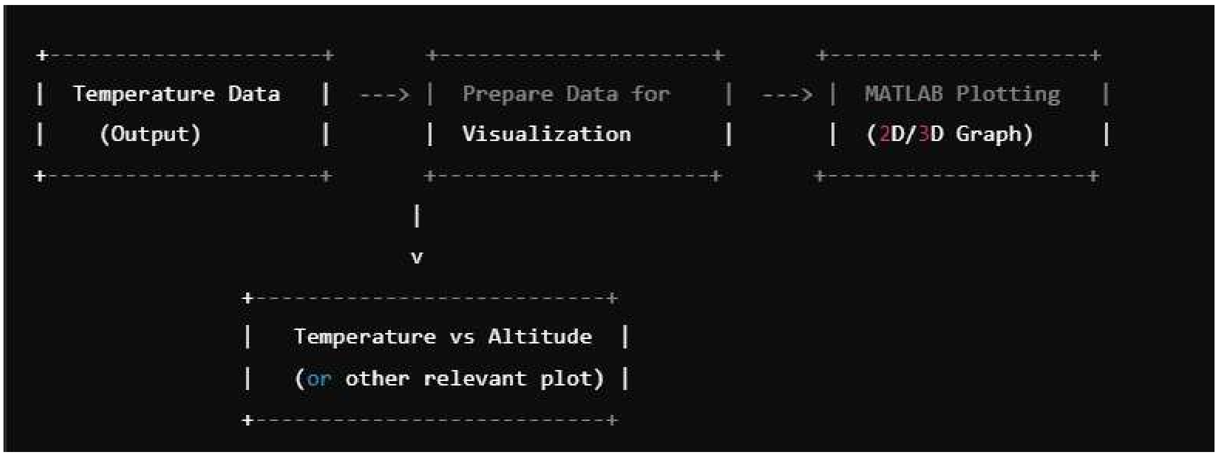




```
Simulation complete.
Data point 1 sent to ThingSpeak.
Data point 2 sent to ThingSpeak.
Data point 3 sent to ThingSpeak.
Data point 4 sent to ThingSpeak.
Data point 5 sent to ThingSpeak.
Data point 6 sent to ThingSpeak.
Data point 7 sent to ThingSpeak.
Data point 8 sent to ThingSpeak.
Data point 9 sent to ThingSpeak.
Data point 10 sent to ThingSpeak.
Data point 11 sent to ThingSpeak.
Data point 12 sent to ThingSpeak.
Data point 13 sent to ThingSpeak.
Data point 14 sent to ThingSpeak.
Data point 15 sent to ThingSpeak.
Data point 16 sent to ThingSpeak.
Data point 17 sent to ThingSpeak.
Data point 18 sent to ThingSpeak.
Data point 19 sent to ThingSpeak.
Data point 20 sent to ThingSpeak.
Data point 21 sent to ThingSpeak.
Data point 22 sent to ThingSpeak.
Data point 23 sent to ThingSpeak.
```



Connection/Procedure Diagrams (If Any):



```
+--------------------+      +--------------------+      +-------------------+
|  Temperature Data  | ---> |  Prepare Data for  | ---> |  MATLAB Plotting  |
|    (Output)        |      |  Visualization     |      |  (2D/3D Graph)    |
+--------------------+      +--------------------+      +-------------------+
                               |
                               v
                   +-------------------------+
                   |  Temperature vs Altitude  |
                   |  (or other relevant plot) |
                   +-------------------------+
```

Result:

Running the code will produce a dynamic simulation of cloud temperature distribution over time, starting with an initial color-coded 2D grid representing the cloud's temperature. This visualization evolves over 100 time steps, showing fluctuations constrained within a specified range. After completing the simulation, the message "Simulation complete." appears in the console. Subsequently, temperature data modeled as a sinusoidal function with random noise is sent to a ThingSpeak channel. For each data point, the console indicates whether the transmission was successful or if an error occurred. After all data points are sent, the process concludes with the message "Data successfully sent to ThingSpeak."

Conclusion/Analysis:

The cloud temperature simulation demonstrates the dynamic nature of temperature variations within a controlled environment, constrained by predefined limits. The visualization provides insights into how small random perturbations influence the overall temperature distribution over time. This highlights the importance of boundary conditions in stabilizing such systems. Additionally, the successful integration with ThingSpeak showcases the capability to transmit real-time data to a cloud-based platform for further analysis or monitoring. This combination of simulation and IoT integration can be applied to real-world scenarios, such as environmental monitoring or predictive modeling of atmospheric conditions.

Learning Outcomes:

- Learn temperature dynamics and stabilization using boundary conditions.
- Practice visualizing dynamic datasets with MATLAB.
- Implement time-dependent simulations for real-world modeling.
- Understand IoT integration with platforms like ThingSpeak.
- Master error handling and API compliance in data transmission.

Experiment 3

Student Name: Shubham Ranjan **UID: 22BCS16090**
Branch: BE-CSE **Section/Group: IOT- 640(A)**
Semester: 6th **Date of Performance:25/01/25**
Subject Name: PBL Java with Lab **Subject Code: 22CSH-359**

1. **Aim:** Create An Application to Calculate Interest for FD's and RD's Based on Certain Conditions Using Inheritance

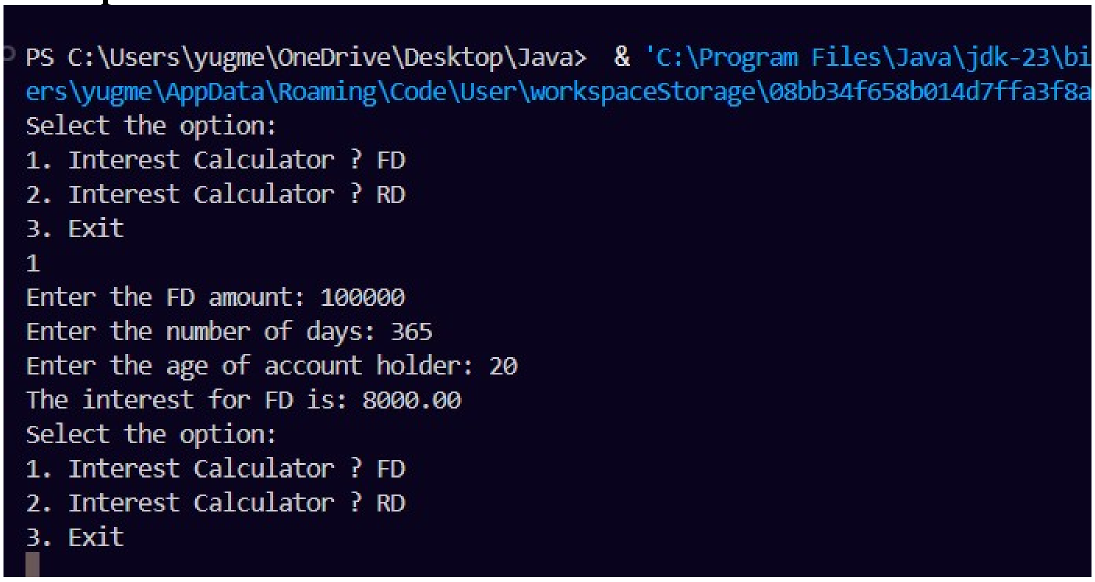
2. **Objective:**

- Implement an Abstract Class: Create an abstract class Account to define common properties and methods for FD and RD accounts.
- Develop Subclasses: Create FD Account and RD Account subclasses that inherit from Account and implement specific interest calculation logic.
- User Input Handling: Design a user interface to collect necessary input data such as amount, days/months, and age for interest calculation.
- Error Management: Implement custom exception handling to manage invalid inputs and provide clear error messages to users.
- Display Results: Calculate and display the interest earned in a user-friendly format, ensuring clarity and ease of understanding.

3. **Code:**

```
import java.util.Scanner;

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

abstract class Account {
    protected double amount;

    public Account(double amount) throws InvalidInputException {
        if (amount < 0) {
            throw new InvalidInputException("Amount cannot be negative.");
        }
        this.amount = amount;
    }
    public abstract double calculateInterest();
}

class FDAccount extends Account {
    private int noOfDays;
    private int ageOfACHolder;

    public FDAccount(double amount, int noOfDays, int ageOfACHolder) throws
      InvalidInputException {
        super(amount);
        if (noOfDays < 0) {
            throw new InvalidInputException("Number of days cannot be negative.");
        }
        this.noOfDays = noOfDays;
        this.ageOfACHolder = ageOfACHolder;
    }

    @Override
    public double calculateInterest() {
        double rate = 0.0;

        if (amount < 1_00_00_000) { // Below 1 Crore
            if (noOfDays <= 14) {
                rate = (ageOfACHolder < 60) ? 4.50 : 5.00;
            } else if (noOfDays <= 29) {
                rate = (ageOfACHolder < 60) ? 4.75 : 5.25;
            } else if (noOfDays <= 45) {
                rate = (ageOfACHolder < 60) ? 5.50 : 6.00;
            } else if (noOfDays <= 60) {
                rate = (ageOfACHolder < 60) ? 7.00 : 7.50;
            } else if (noOfDays <= 184) {
                rate = (ageOfACHolder < 60) ? 7.50 : 8.00;
            } else if (noOfDays <= 365) {
                rate = (ageOfACHolder < 60) ? 8.00 : 8.50;
            }
        } else { // Above 1 Crore
            if (noOfDays <= 14) {
                rate = 6.50;
            } else if (noOfDays <= 29) {
                rate = 6.75;
            } else if (noOfDays <= 45) {
                rate = 6.75;
            } else if (noOfDays <= 60) {
                rate = 8.00;
            } else if (noOfDays <= 184) {
                rate = 8.50;
            } else if (noOfDays <= 365) {
                rate = 10.00;
            }
        }

        return (amount * rate * noOfDays) / (100 * 365);
    }
}

// RD Account Class
class RDAccount extends Account {
    private int noOfMonths;
    private int ageOfACHolder;

    public RDAccount(double amount, int noOfMonths, int ageOfACHolder) throws
      InvalidInputException {
        super(amount);
        if (noOfMonths < 0) {
            throw new InvalidInputException("Number of months cannot be negative.");
        }
        this.noOfMonths = noOfMonths;
        this.ageOfACHolder = ageOfACHolder;
    }

    @Override
    public double calculateInterest() {
        double rate = 0.0;

        switch (noOfMonths) {
            case 6:
                rate = (ageOfACHolder < 60) ? 7.50 : 8.00;
                break;
            case 9:
                rate = (ageOfACHolder < 60) ? 7.75 : 8.25;
                break;
            case 12:
                rate = (ageOfACHolder < 60) ? 8.00 : 8.50;
                break;
            case 15:
                rate = (ageOfACHolder < 60) ? 8.25 : 8.75;
                break;
            case 18:
                rate = (ageOfACHolder < 60) ? 8.50 : 9.00;
                break;
            case 21:
                rate = (ageOfACHolder < 60) ? 8.75 : 9.25;
                break;
            default:
                throw new IllegalArgumentException("Invalid number of months for RD.");
        }
        return (amount * rate * noOfMonths) / (100 * 12);
    }
}

// Main Class to Run the Interest Calculator
public class InterestCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("Select the option:");
            System.out.println("1. Interest Calculator – FD");
            System.out.println("2. Interest Calculator – RD");
            System.out.println("3. Exit");
            int choice = scanner.nextInt();

            try {
                if (choice == 1) {
                    System.out.print("Enter the FD amount: ");
                    double amount = scanner.nextDouble();
                    System.out.print("Enter the number of days: ");
                    int noOfDays = scanner.nextInt();
                    System.out.print("Enter the age of account holder: ");
                    int ageOfACHolder = scanner.nextInt();

                    FDAccount fdAccount = new FDAccount(amount, noOfDays,
      ageOfACHolder);
                    double interest = fdAccount.calculateInterest();
                    System.out.printf("The interest for FD is: %.2f%n", interest);
                } else if (choice == 2) {
                    System.out.print("Enter the RD amount: ");
                    double amount = scanner.nextDouble();
                    System.out.print("Enter the number of months: ");
                    int noOfMonths = scanner.nextInt();
                    System.out.print("Enter the age of account holder: ");
                    int ageOfACHolder = scanner.nextInt();

                    RDAccount rdAccount = new RDAccount(amount, noOfMonths,
      ageOfACHolder);
                    double interest = rdAccount.calculateInterest();
                    System.out.printf("The interest for RD is: %.2f%n", interest);
                } else if (choice == 3) {
                    System.out.println("Exiting the application.");
                    break;
                } else {
                    System.out.println("Invalid choice. Please try again.");
                }
            } catch (InvalidInputException | IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }

        scanner.close();
    }
}
```

4. Output



```
PS C:\Users\yugme\OneDrive\Desktop\Java>  & 'C:\Program Files\Java\jdk-23\bi
ers\yugme\AppData\Roaming\Code\User\workspaceStorage\08bb34f658b014d7ffa3f8a
Select the option:
1. Interest Calculator ? FD
2. Interest Calculator ? RD
3. Exit
1
Enter the FD amount: 100000
Enter the number of days: 365
Enter the age of account holder: 20
The interest for FD is: 8000.00
Select the option:
1. Interest Calculator ? FD
2. Interest Calculator ? RD
3. Exit
```

5. Learning Outcome

- OOP Principles: Understand and apply object-oriented programming concepts such as inheritance and polymorphism.
- Inheritance Application: Learn to create a base class and derive specific subclasses for enhanced code reusability.
- Exception Handling: Develop skills in managing invalid inputs through effective custom exception handling.
- Interest Calculation Logic: Grasp the financial concepts related to interest calculation for Fixed Deposits and Recurring Deposits.
- User Input Management: Effectively gather and display user input and output in a clear and user-friendly manner.