Program:

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

class State {
    private String name;
    private Map<Character, Transition> transitions;

    public State(String name) {
        this.name = name;
        this.transitions = new HashMap<>();
    }
    public void addTransition(char input, State nextState, String output) {
        transitions.put(input, new Transition(nextState, output));
    }
    public Transition getTransition(char input) {
        return transitions.get(input);
    }
    @Override
    public String toString() {
        return name;
    }
}

class Transition {
    private State nextState;
    private String output;

    public Transition(State nextState, String output) {
        this.nextState = nextState;
        this.output = output;
    }
    public State getNextState() {
        return nextState;
    }
    public String getOutput() {
        return output;
    }
}

class MealyMachine {
    private State initialState;
    private State currentState;

    public MealyMachine(State initialState) {
        this.initialState = initialState;
```

```
        this.currentState = initialState;
    }
    public void reset() {
        currentState = initialState;
    }
    public String process(String inputs) {
        StringBuilder outputs = new StringBuilder();
        for (char input : inputs.toCharArray()) {
            Transition transition = currentState.getTransition(input);
            outputs.append(transition.getOutput());
            currentState = transition.getNextState();
        }
        return outputs.toString();
    }

    public static void main(String[] args) {
        // Define states
        State s0 = new State("S0");
        State s1 = new State("S1");
        State s2 = new State("S2");
        // Define transitions (current state, input -> next state, output)
        s0.addTransition('a', s1, "1");
        s0.addTransition('b', s0, "0");
        s1.addTransition('a', s1, "1");
        s1.addTransition('b', s2, "0");
        s2.addTransition('a', s1, "1");
        s2.addTransition('b', s0, "0");

        // Create the Mealy machine
        MealyMachine mealyMachine = new MealyMachine(s0);

        // Get user input
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the input string: ");
        String inputs = scanner.nextLine();

        // Process inputs
        String outputs = mealyMachine.process(inputs);
        System.out.println("Inputs: " + inputs);
        System.out.println("Outputs: " + outputs);
    }
}
```

Output:

```
Enter the input string: abba
Inputs: abba
Outputs: 1001
```

```
Enter the input string: abb
Inputs: abb
Outputs: 100
```

```
Enter the input string: abab
Inputs: abab
Outputs: 1010
```

Observation:

The primary objective of this experiment is to design, implement, and analyse the behavior of a Mealy machine. The focus is on understanding how the outputs of the machine are determined by both the current state and the current input, in contrast to a Moore machine, where the output is solely dependent on the current state.

Conclusion:

The Mealy machine was successfully designed and implemented, demonstrating its key characteristic of output dependency on both the current state and input.