Model Driven Software Engineering

(COEN 6312)

Project Deliverable 4

Submitted to

Dr. AbdelwahabHamou-Lhadj

Submitted by

Nareshkumar M. Sisodiya 27650817

Arjun Lokhande 27411111

Binu Basil John 27421753

Anant Mathur 27323670

Khushboo Handa 27323794

March 23, 2016

Contents

List of Figures

1. Introduction

State Diagram

A state diagram defines the dynamic behaviour of the objects of any class. As the name suggests a state diagram indicates the various states of an object. At a time the object of a class remains in only one state until it receives a signal such as a method, operation, external or internal signals which triggers it into another state. An action is an instruction or a statement described using an action specification language.

2. User

2.1 State Machine Diagram

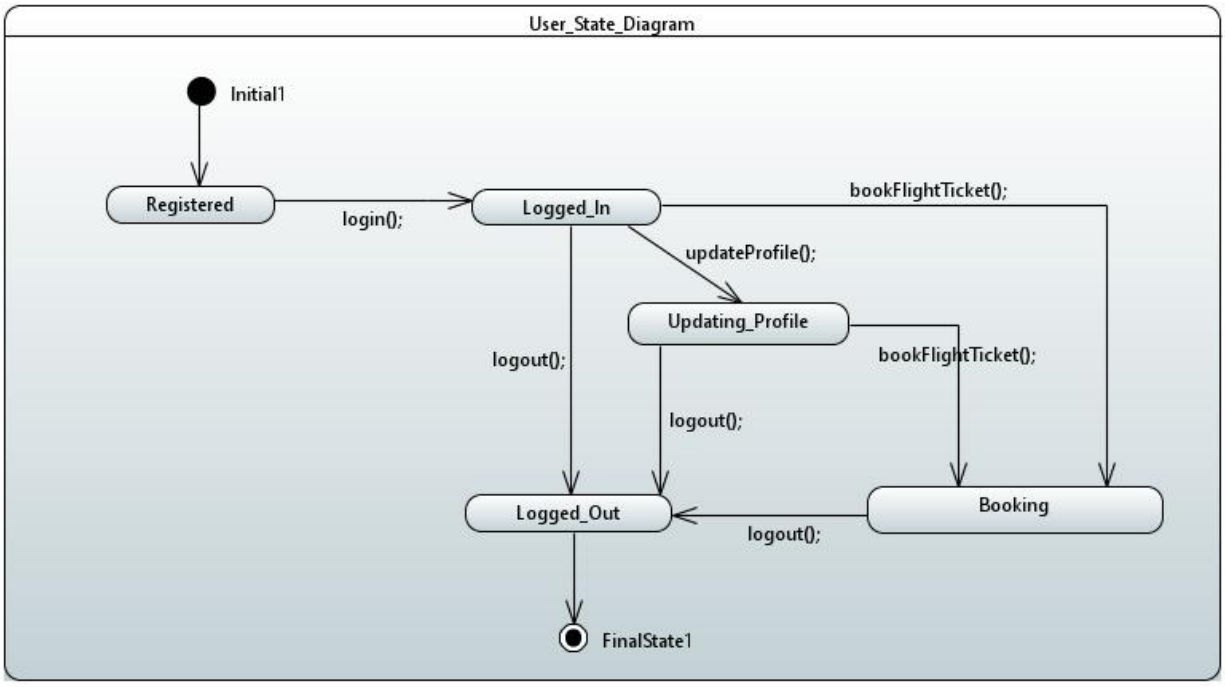

Figure 1: State Machine Diagram - User Class

2.2 Operations

```
package FlyAir;

import java.util.Scanner;

public class User {
	static String userName;
	static String password;
	public static void main(String args[])
	{
```

```
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter Login details");
            userName = sc.nextLine();
            password = sc.nextLine();
            if(authenticate())
            {
            System.out.println("User Successfully Logged in");

            System.out.println("Please Select the operation:");
            System.out.println("\t\t\t1. Book Flight");
            System.out.println("\t\t\t2. Update User Profile");
            System.out.println("\t\t\t3. Logout");
            int op;
            op = sc.nextInt();

            switch (op) {
            case 1:
                  bookFlight();
                  break;
            case 2:
                  updateProfile();
                  break;
            case 3:
                  System.out.println("User Logged Out Succesfully.");
                  break;
            default:
                  break;
            }
            }
            else
            {
                  System.out.println("Invalid Username or password");
            }
      }

      private static boolean authenticate()
      {
            // Code to authenticate user credentials

      }

      private static void updateProfile()
      {
            // Code to update user profile
      }

      private static void bookFlightTicket()
      {
            // Code to book flight ticket

      }

}
```

3. Ticket

3.1 State Machine Diagram

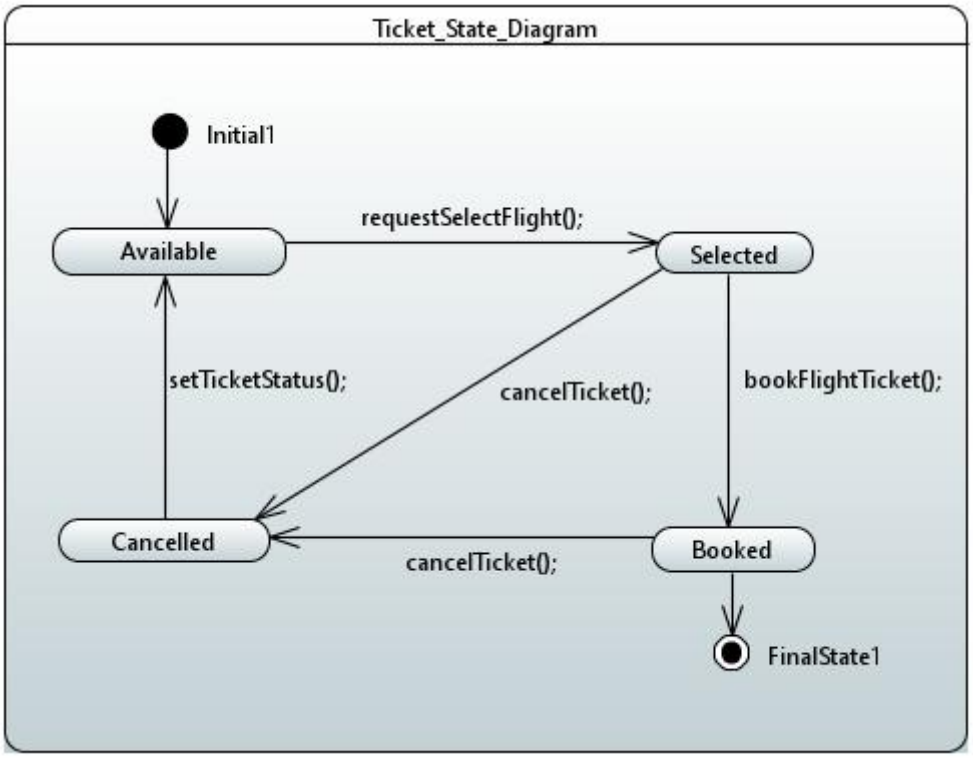

Figure 2: State Machine Diagram - Ticket Class

3.2 Operations

```
import java.util.Scanner;


public class ticket {

	public static void requestSelectFlight()
	{

		//code for selecting the flight
	}

	public static void cancelTicket()
	{

		//code for canceling the flight ticket
	}

	public static void bookFlightTicket()
	{
		boolean success = false;

		//code for booking the flight
```

```
            // if booked then success = true
            if (success)
                  System.out.println("Flight booked successfully");
            else
                  System.out.println("Error in booking Flight ");

      }

      public static boolean getTicketStatus(String flightDetails)
      {
            //checks if the flight is available or not
            //if avaliable then return true else return false
            return true;

      }


      public static void main(String args[])
      {
            Scanner sc = new Scanner(System.in);
            boolean card_validate;
            System.out.println("Enter the flight details");
            String flightDetails = sc.nextLine();

            if(getTicketStatus(flightDetails))
            {
                  System.out.println("Flight is available . Press 'Y'
to book or press 'N' to cancel booking");
                  String booking = sc.nextLine();

                  if(booking.equals('Y'))
                        bookFlightTicket();
                  else if(booking.equals('N'))
                        cancelTicket();
                  else
                        System.out.println("Invalid Option");

            }

            else
            {
                  System.out.println("entered flight is not
avaliable");
            }

      }

}
```

4. Payment

4.1 State Machine Diagram

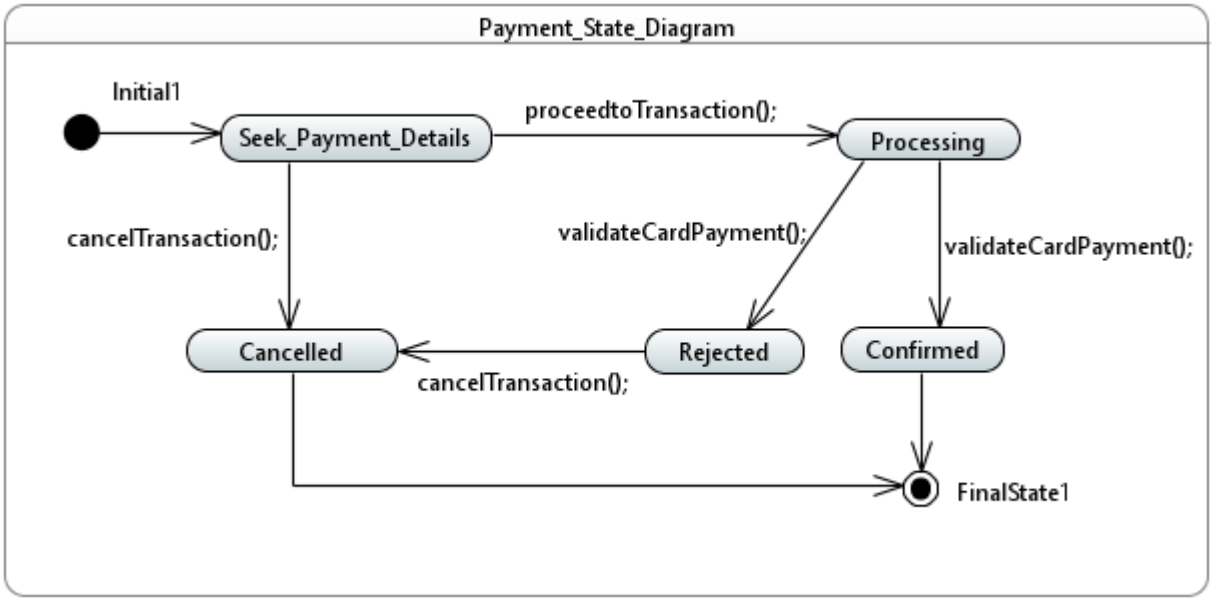

Figure 3: State Machine Diagram - Payment Class

4.2 Operations

```
import java.util.Scanner;

public class payment {

	public static void inputDebitCardInfo()
	{
		//take debit card info
	}

	public static void inputCreditCardInfo()
	{
		//take credit card info
	}

	public static boolean process_transaction()
	{
		//code for transaction processing
		//return true if success else return false
		return true;

	}
	public static boolean validateCard()
	{
		//If valid then return true else return false
```

```
            return true;

      }


      @SuppressWarnings("unused")
      public static void main(String args[])

      {
            Float PaymentAmount;
            Scanner sc = new Scanner(System.in);
            boolean card_validate;
            System.out.println("Enter the Payment Type");
            String PaymentType = sc.nextLine();
            switch(PaymentType)
            {
            case("Debit"): inputDebitCardInfo();
                              if(validateCard())
                                    card_validate = true;

            case("Credit"):inputCreditCardInfo();
                              if(validateCard())
                                    card_validate = true;
            }

            if(card_validate = true)
            {
                  process_transaction();
                  System.out.println("Transaction successful");

            }

            else
            {
                  System.out.println("Invalid card details .
Transaction cancelled");
            }

      }


}
```