Project

HOTEL RESERVATION SYSTEM

Introduction

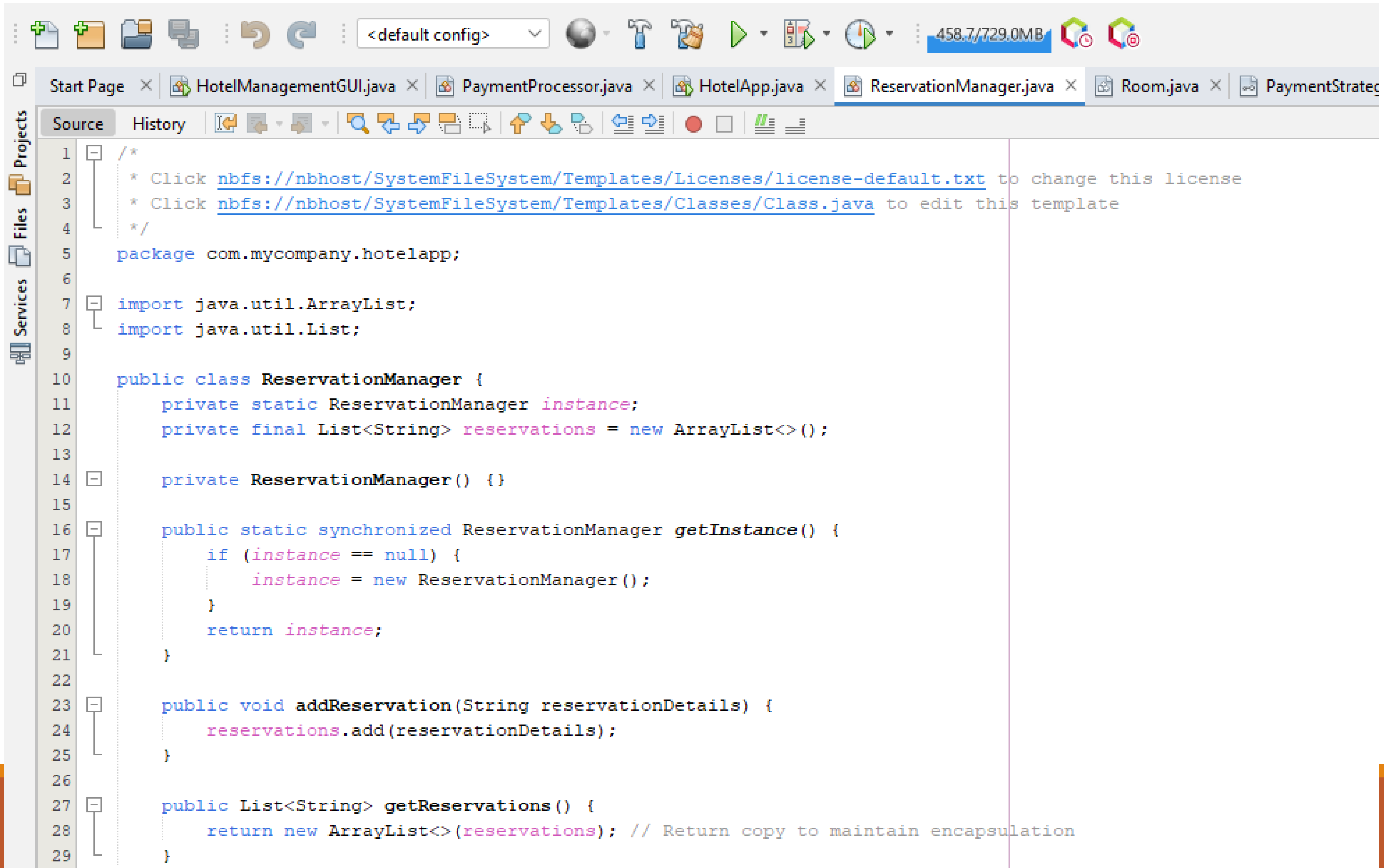
The **Hotel Reservation System** The project is a Java GUI application that allows users to book hotel rooms or make payments, with the ability to choose extras such as breakfast for the rooms. The project is implemented using several design patterns to write clean and extensible code.

Summary

Discrepyion	Design pattern	Class
الملف الرئيسي الذي يحتوي على الواجهة الرسومية ويربط جميع المكونات.	Main	**HotelApp.java**
يُدير جميع بيانات الحجوزات.	Singleton	**ReservationManager**
يُدير عمليات الدفع.	Singleton	**PaymentProcessor**
يُنشئ كائنات الغرف Standard/Deluxe	Factory	RoomFactory
يُضيف ميزات إضافية للغرفة مثل الإفطار.	Decorator	RoomDecorator
يُدير إستراتيجيات الدفع المختلفة Credit/Cash	Strategy	**PaymentStrategy**
يعرض التحديثات (الحجوزات/الدفع) فور إدخالها للمستخدم.	Observer	**JTextArea in HotelApp**

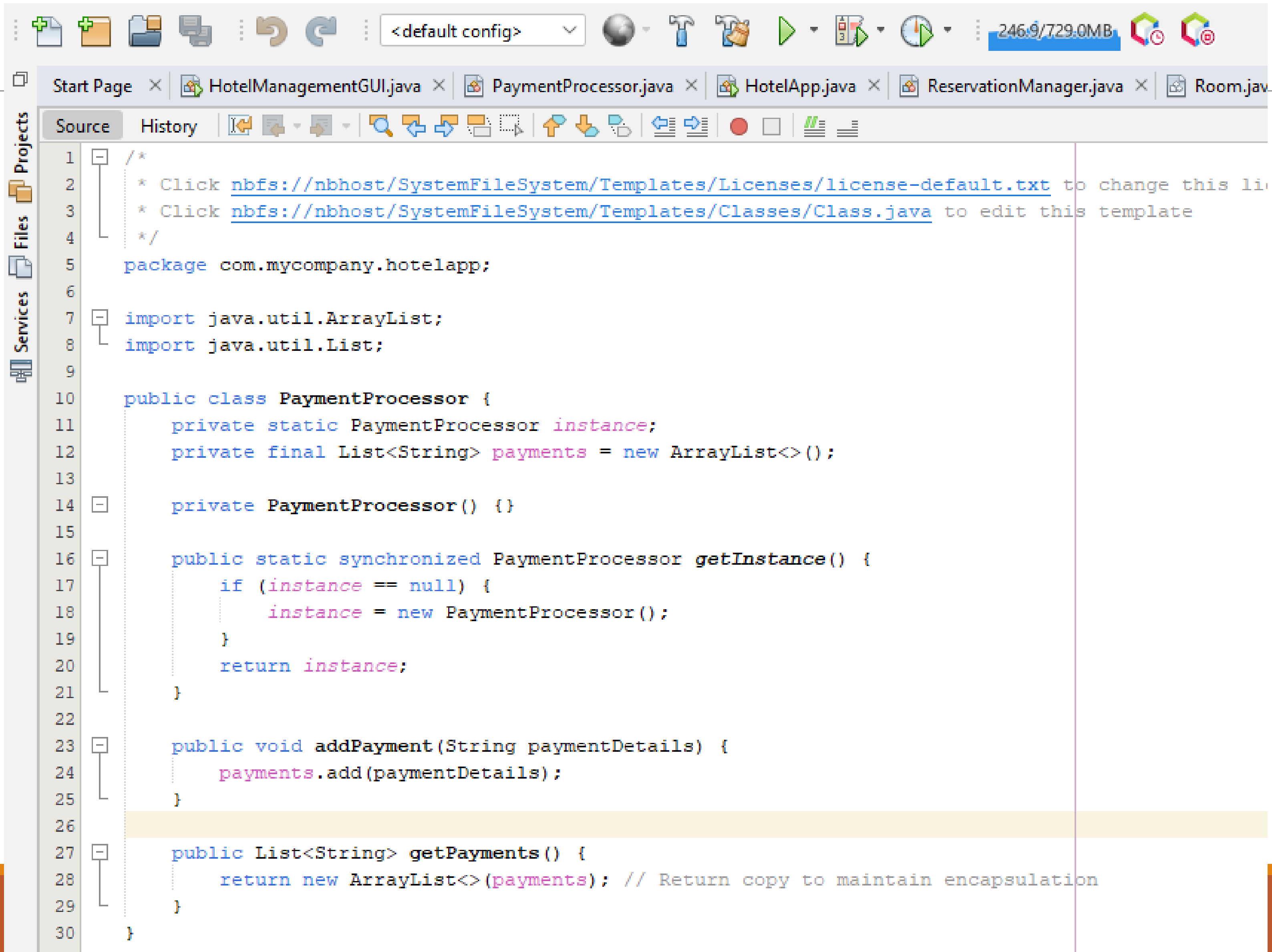
1. Singleton Pattern : Purpose: Ensures that there is only one copy of the base class that runs the main functions.

ReservationManager :Manages all reservations data. The getInstance method ensures that there is only one instance of this class in the application.



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.hotelapp;

import java.util.ArrayList;
import java.util.List;

public class ReservationManager {
    private static ReservationManager instance;
    private final List<String> reservations = new ArrayList<>();

    private ReservationManager() {}

    public static synchronized ReservationManager getInstance() {
        if (instance == null) {
            instance = new ReservationManager();
        }
        return instance;
    }

    public void addReservation(String reservationDetails) {
        reservations.add(reservationDetails);
    }

    public List<String> getReservations() {
        return new ArrayList<>(reservations); // Return copy to maintain encapsulation
    }
```

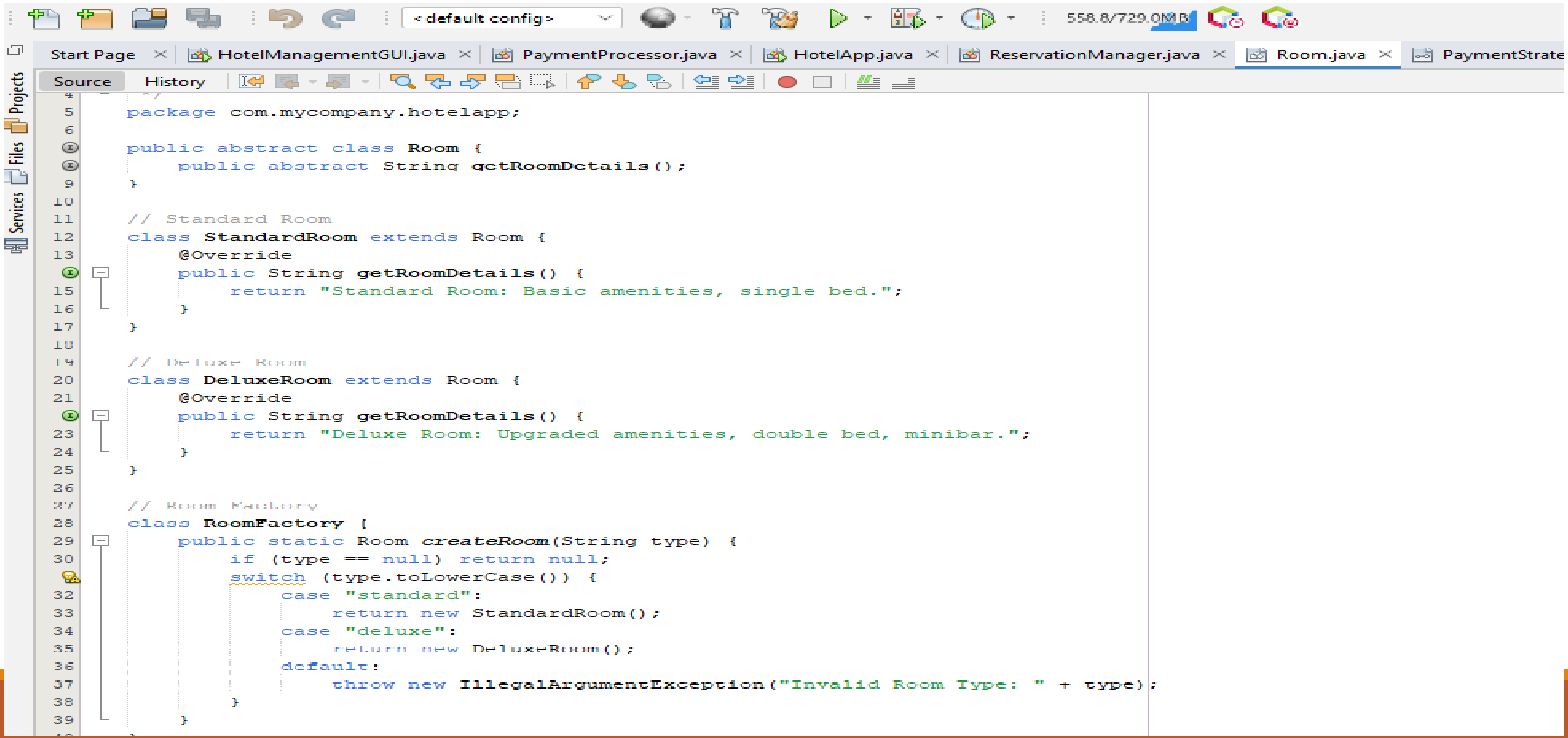
PaymentProcessor : Manages payment processes. Like ReservationManager, Singleton is used to get a single copy.



```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this li
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.hotelapp;

import java.util.ArrayList;
import java.util.List;

public class PaymentProcessor {
    private static PaymentProcessor instance;
    private final List<String> payments = new ArrayList<>();

    private PaymentProcessor() {}

    public static synchronized PaymentProcessor getInstance() {
        if (instance == null) {
            instance = new PaymentProcessor();
        }
        return instance;
    }

    public void addPayment(String paymentDetails) {
        payments.add(paymentDetails);
    }

    public List<String> getPayments() {
        return new ArrayList<>(payments); // Return copy to maintain encapsulation
    }
}
```

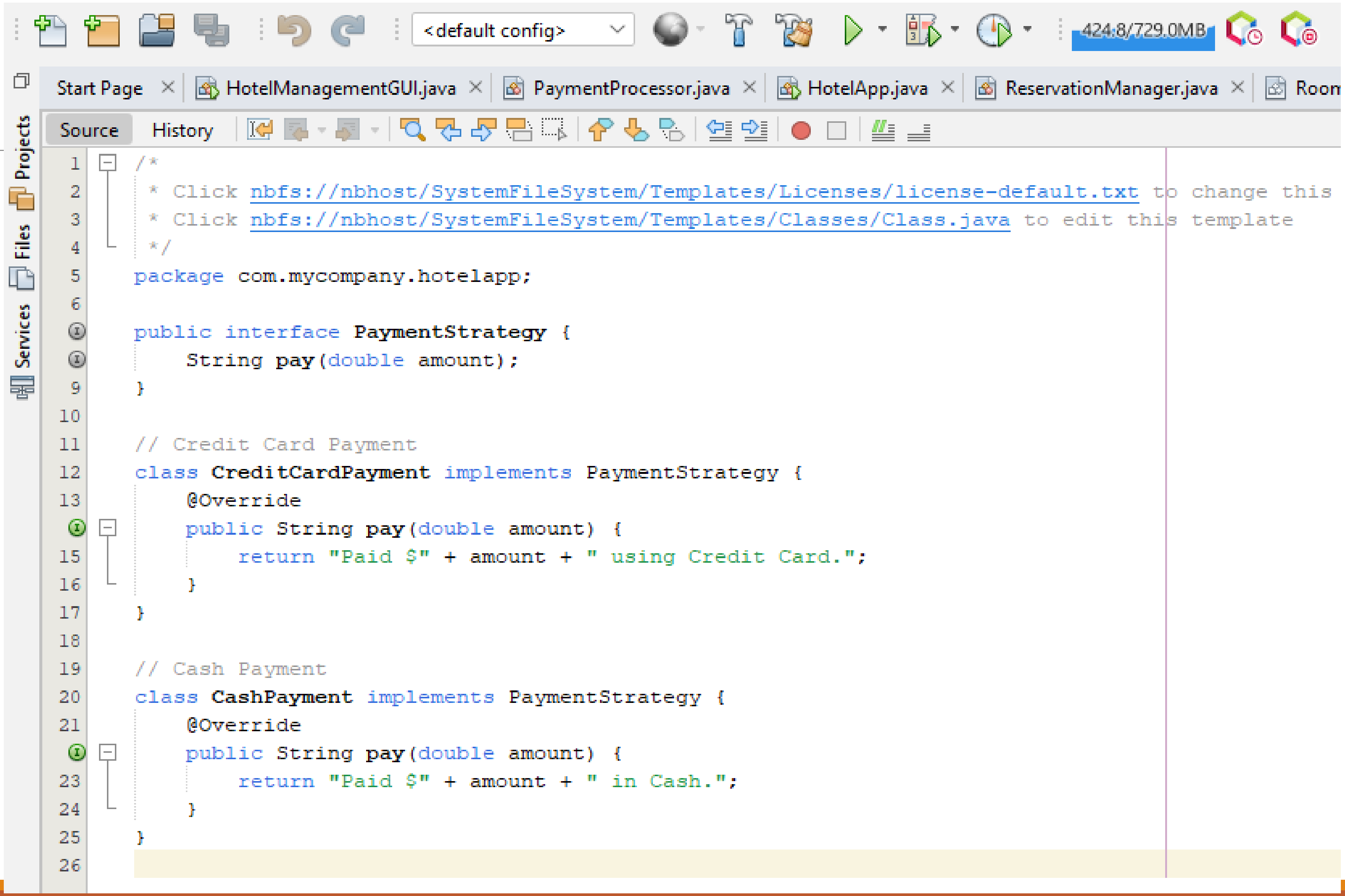
2. Factory Pattern : Purpose: Facilitates the creation of dynamic objects based on user input.

RoomFactory : Creates different types of rooms: StandardRoom and DeluxeRoom.



```java
package com.mycompany.hotelapp;

public abstract class Room {
    public abstract String getRoomDetails();
}

// Standard Room
class StandardRoom extends Room {
    @Override
    public String getRoomDetails() {
        return "Standard Room: Basic amenities, single bed.";
    }
}

// Deluxe Room
class DeluxeRoom extends Room {
    @Override
    public String getRoomDetails() {
        return "Deluxe Room: Upgraded amenities, double bed, minibar.";
    }
}

// Room Factory
class RoomFactory {
    public static Room createRoom(String type) {
        if (type == null) return null;
        switch (type.toLowerCase()) {
            case "standard":
                return new StandardRoom();
            case "deluxe":
                return new DeluxeRoom();
            default:
                throw new IllegalArgumentException("Invalid Room Type: " + type);
        }
    }
```

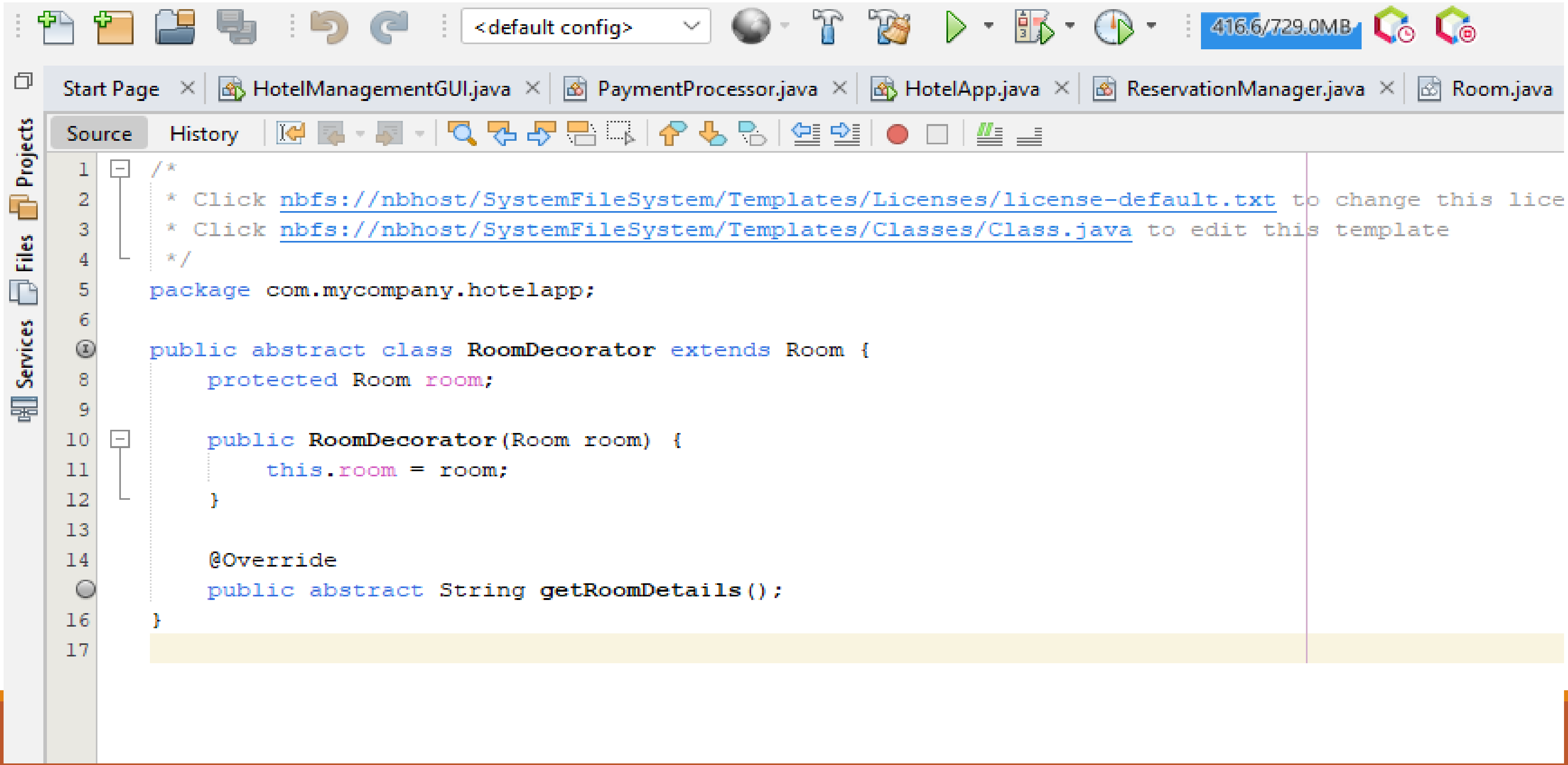
PaymentStrategy : (implicitly, in specifying the payment strategy).



```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.hotelapp;

public interface PaymentStrategy {
    String pay(double amount);
}

// Credit Card Payment
class CreditCardPayment implements PaymentStrategy {
    @Override
    public String pay(double amount) {
        return "Paid $" + amount + " using Credit Card.";
    }
}

// Cash Payment
class CashPayment implements PaymentStrategy {
    @Override
    public String pay(double amount) {
        return "Paid $" + amount + " in Cash.";
    }
}

```

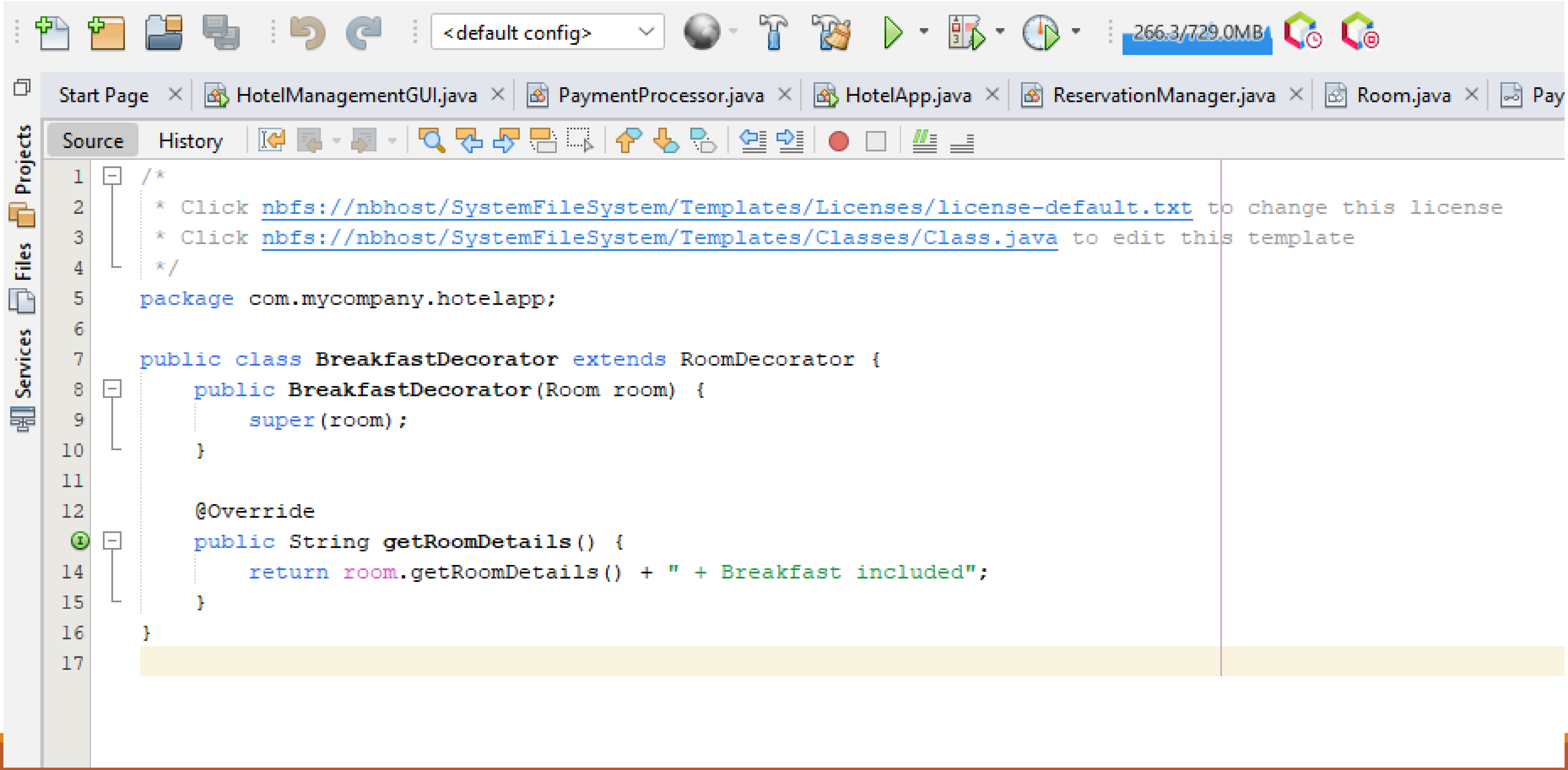
3. Decorator Pattern : Purpose: Allows additional features to be added to rooms such as breakfast, without changing the design of the original objects.

RoomDecorator: An abstract class that is inherited.



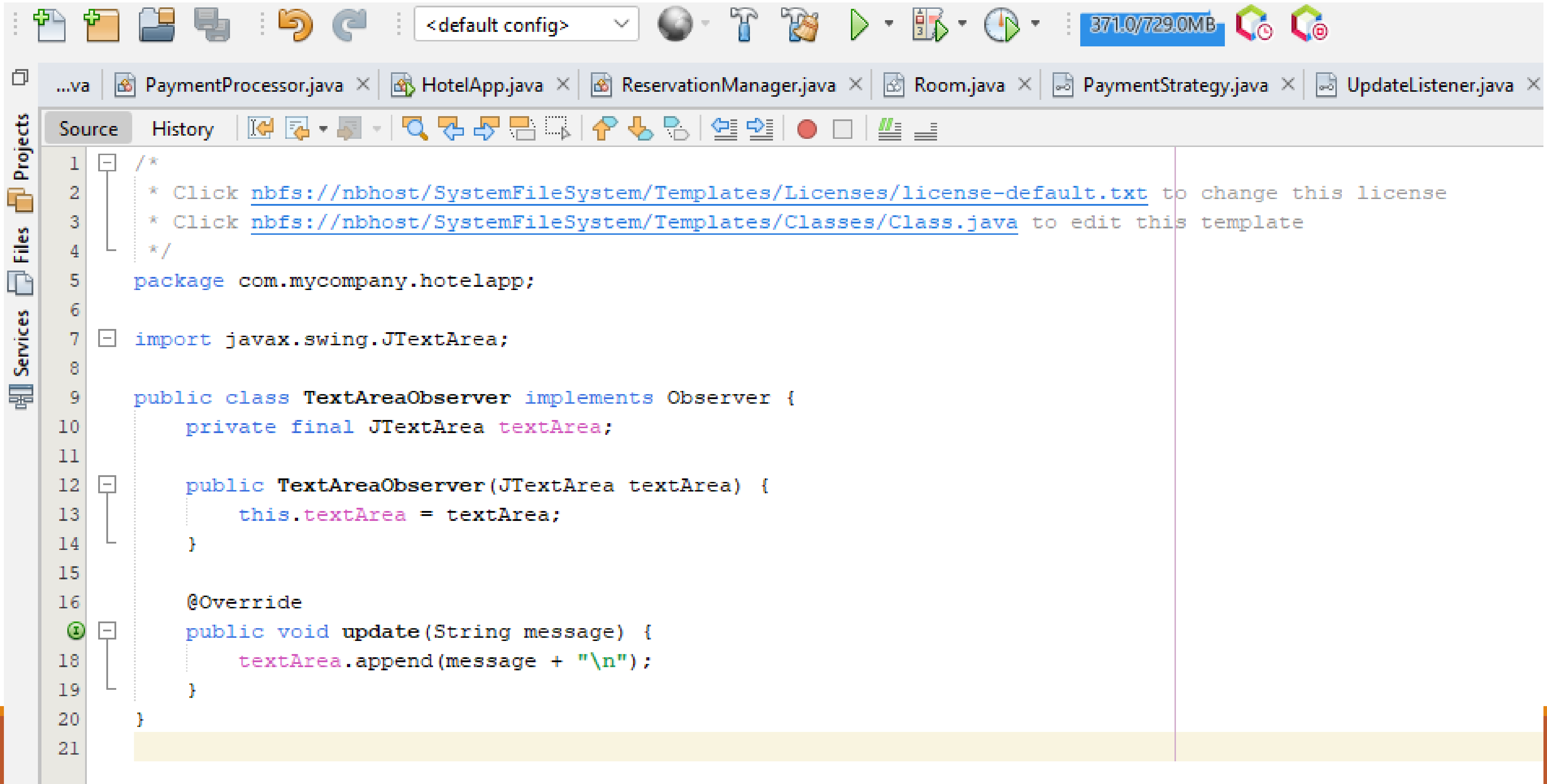
```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this lice
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.hotelapp;

public abstract class RoomDecorator extends Room {
    protected Room room;

    public RoomDecorator(Room room) {
        this.room = room;
    }

    @Override
    public abstract String getRoomDetails();
}
```

BreakfastDecorator: Adds breakfast to the room.



```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.hotelapp;

public class BreakfastDecorator extends RoomDecorator {
    public BreakfastDecorator(Room room) {
        super(room);
    }

    @Override
    public String getRoomDetails() {
        return room.getRoomDetails() + " + Breakfast included";
    }
}
```

Observer Pattern: ReservationManager and PaymentProcessor are now "Subjects", notifying moderators when there is an update.
TextAreaObserver keeps track of data changes, and automatically displays new bookings and payments in the text interface (JTextArea)



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.hotelapp;

import javax.swing.JTextArea;

public class TextAreaObserver implements Observer {
    private final JTextArea textArea;

    public TextAreaObserver(JTextArea textArea) {
        this.textArea = textArea;
    }

    @Override
    public void update(String message) {
        textArea.append(message + "\n");
    }
}
```

How to work the project

Booking:

The user enters the room type, name, and phone number.

Can add additional features (such as breakfast).

Your reservation details appear at the bottom of the text interface.

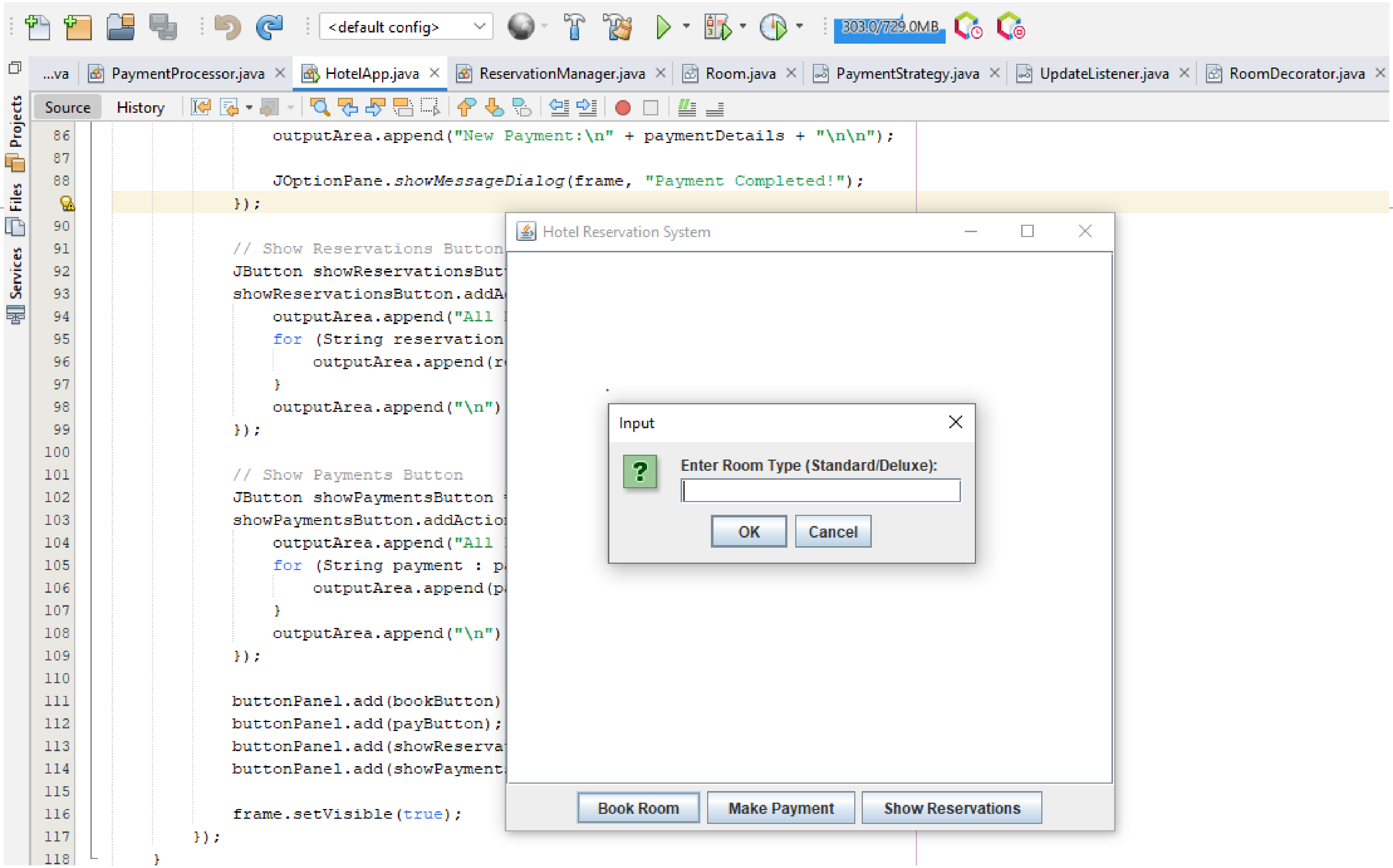
Payment:

User chooses payment method (cash/card).

The amount and card number are entered (to pay by card).

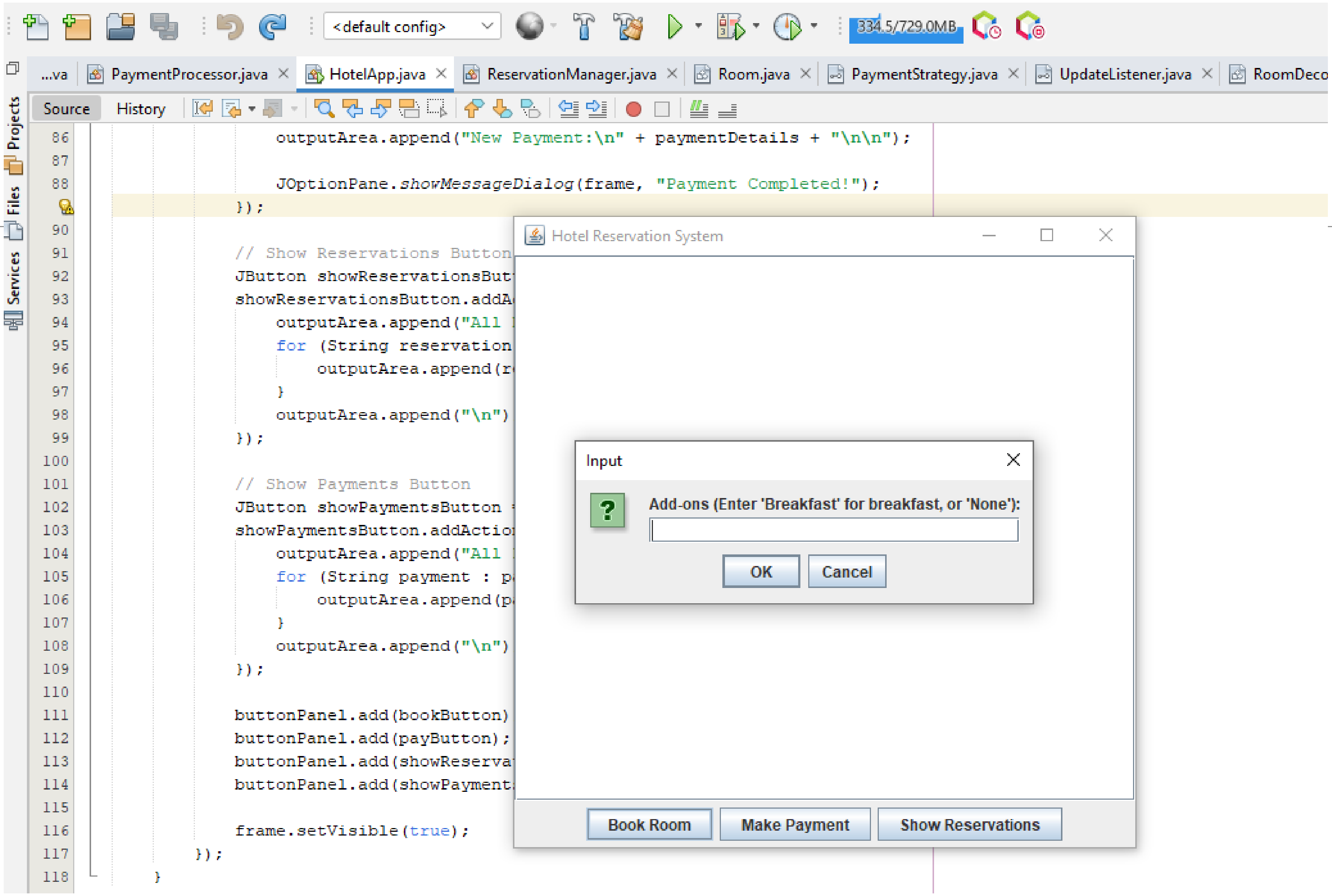
Payment details are displayed in the text interface.

Show Reservation



```
            outputArea.append("New Payment:\n" + paymentDetails + "\n\n");

            JOptionPane.showMessageDialog(frame, "Payment Completed!");
        });

        // Show Reservations Button
        JButton showReservationsBut
        showReservationsButton.addA
            outputArea.append("All
            for (String reservation
                outputArea.append(r
            }
            outputArea.append("\n")
        });

        // Show Payments Button
        JButton showPaymentsButton
        showPaymentsButton.addActio
            outputArea.append("All
            for (String payment : p
                outputArea.append(p
            }
            outputArea.append("\n")
        });

        buttonPanel.add(bookButton)
        buttonPanel.add(payButton);
        buttonPanel.add(showReserva
        buttonPanel.add(showPayment

        frame.setVisible(true);
    });
}
```

Enter Room Type (Standard/Deluxe):

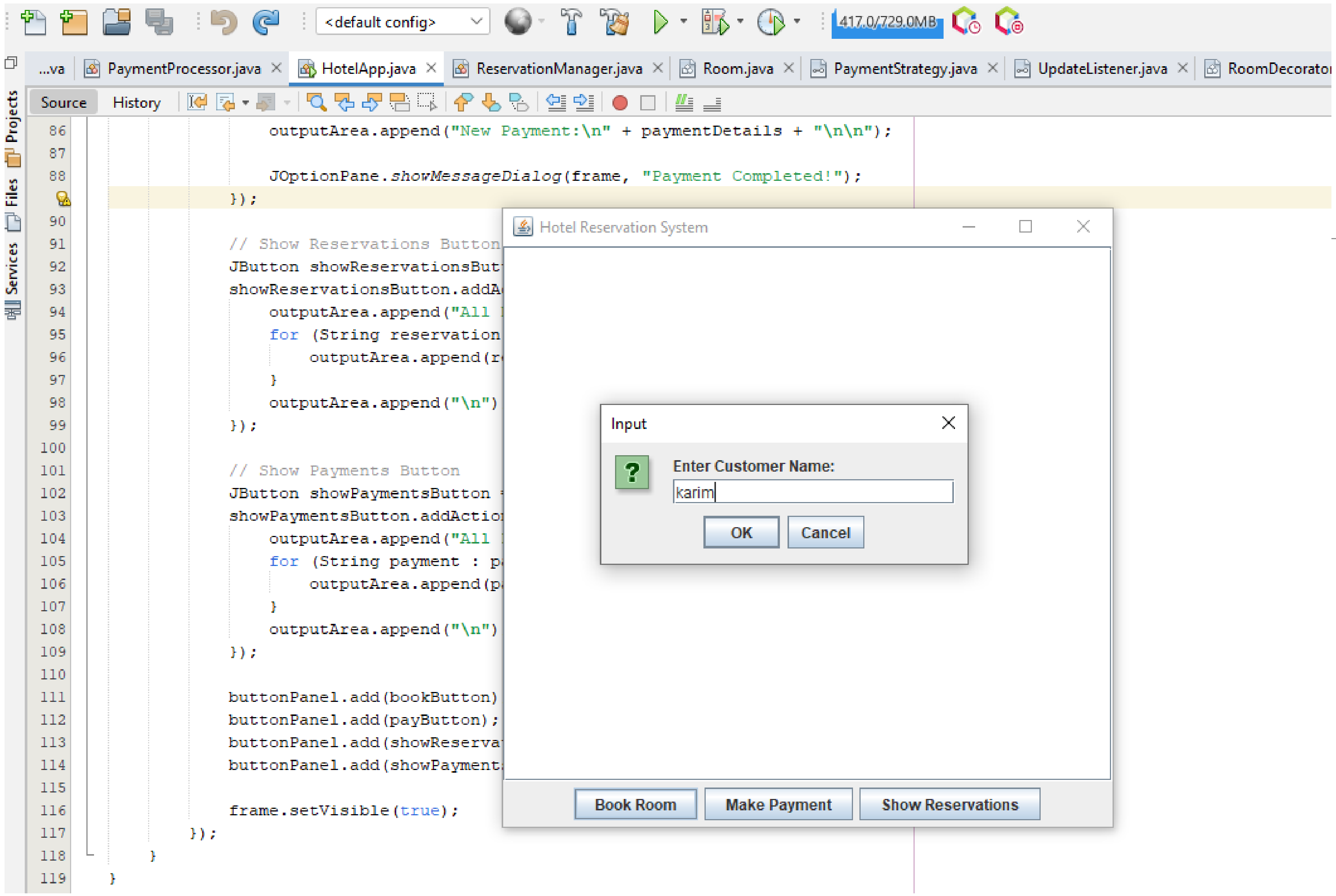


```
                outputArea.append("New Payment:\n" + paymentDetails + "\n\n");

                JOptionPane.showMessageDialog(frame, "Payment Completed!");
            });

            // Show Reservations Button
            JButton showReservationsBut
            showReservationsButton.addA
                outputArea.append("All 
                for (String reservation
                    outputArea.append(r
                }
                outputArea.append("\n")
            });

            // Show Payments Button
            JButton showPaymentsButton 
            showPaymentsButton.addActio
                outputArea.append("All 
                for (String payment : p
                    outputArea.append(p
                }
                outputArea.append("\n")
            });

            buttonPanel.add(bookButton)
            buttonPanel.add(payButton);
            buttonPanel.add(showReserva
            buttonPanel.add(showPayment

            frame.setVisible(true);
        });
    }
```

Hotel Reservation System

Input

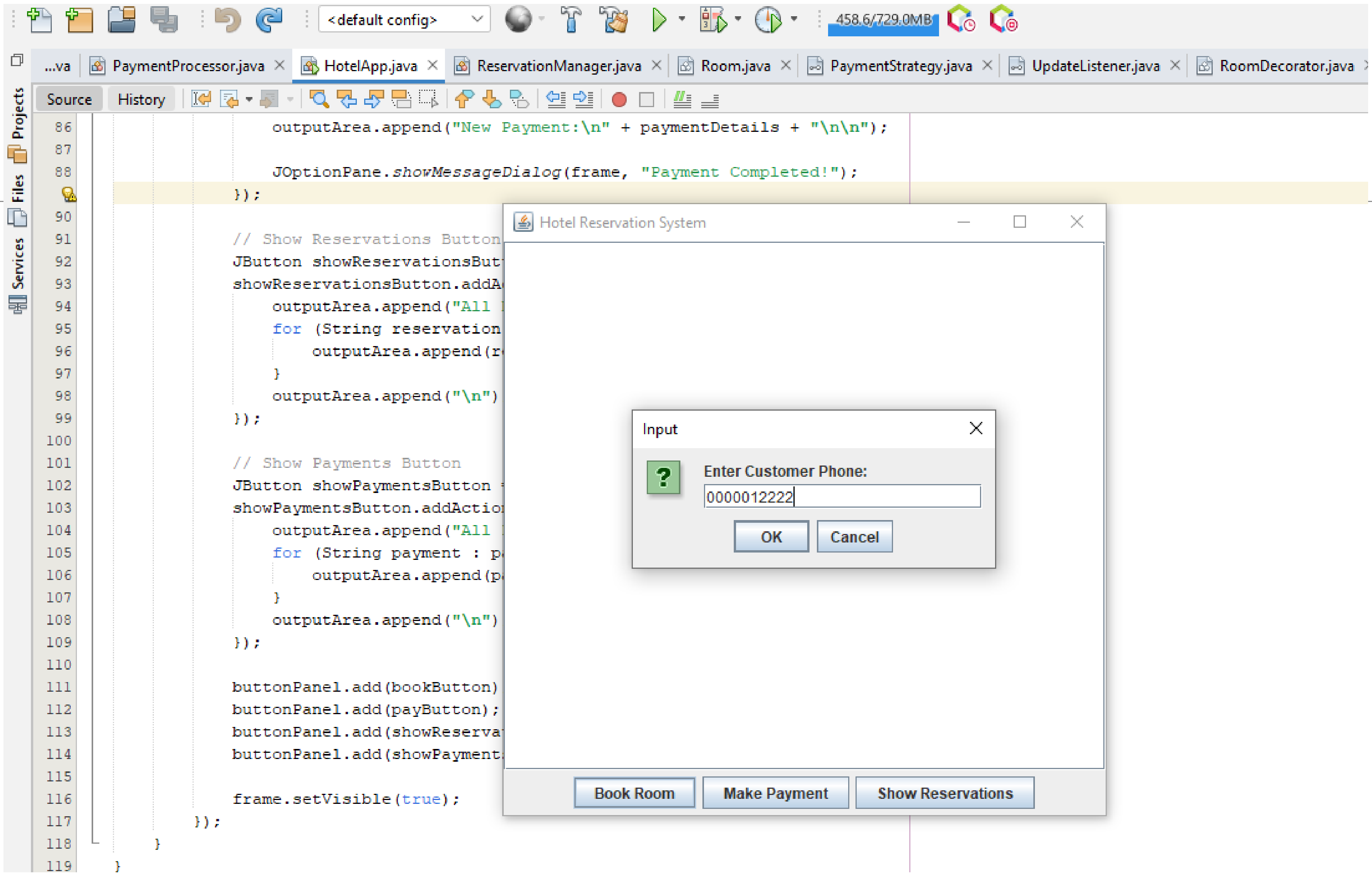
Add-ons (Enter 'Breakfast' for breakfast, or 'None'):



```
                outputArea.append("New Payment:\n" + paymentDetails + "\n\n");

                JOptionPane.showMessageDialog(frame, "Payment Completed!");
            });

            // Show Reservations Button
            JButton showReservationsBut
            showReservationsButton.addA
                outputArea.append("All
                for (String reservation
                    outputArea.append(r
                }
                outputArea.append("\n")
            });

            // Show Payments Button
            JButton showPaymentsButton
            showPaymentsButton.addActio
                outputArea.append("All
                for (String payment : p
                    outputArea.append(p
                }
                outputArea.append("\n")
            });

            buttonPanel.add(bookButton)
            buttonPanel.add(payButton);
            buttonPanel.add(showReserva
            buttonPanel.add(showPayment

            frame.setVisible(true);
        });
    }
}
```

Enter Customer Name:

karim

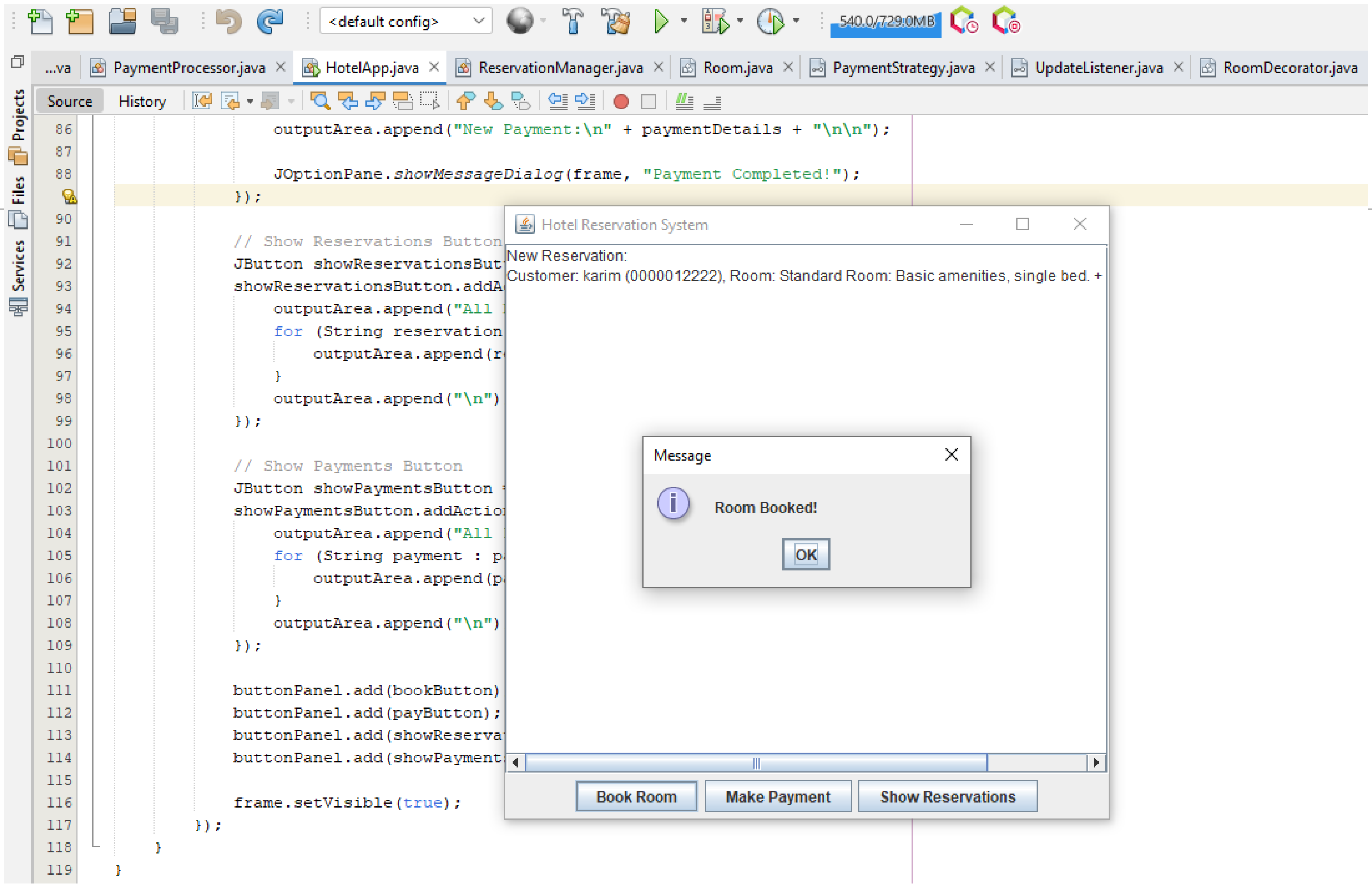


```
                outputArea.append("New Payment:\n" + paymentDetails + "\n\n");

                JOptionPane.showMessageDialog(frame, "Payment Completed!");
            });

            // Show Reservations Button
            JButton showReservationsBut
            showReservationsButton.addA
                outputArea.append("All 
                for (String reservation
                    outputArea.append(r
                }
                outputArea.append("\n")
            });

            // Show Payments Button
            JButton showPaymentsButton 
            showPaymentsButton.addActio
                outputArea.append("All 
                for (String payment : p
                    outputArea.append(p
                }
                outputArea.append("\n")
            });

            buttonPanel.add(bookButton)
            buttonPanel.add(payButton);
            buttonPanel.add(showReserva
            buttonPanel.add(showPayment

            frame.setVisible(true);
        });
    }
}
```

Hotel Reservation System

Input

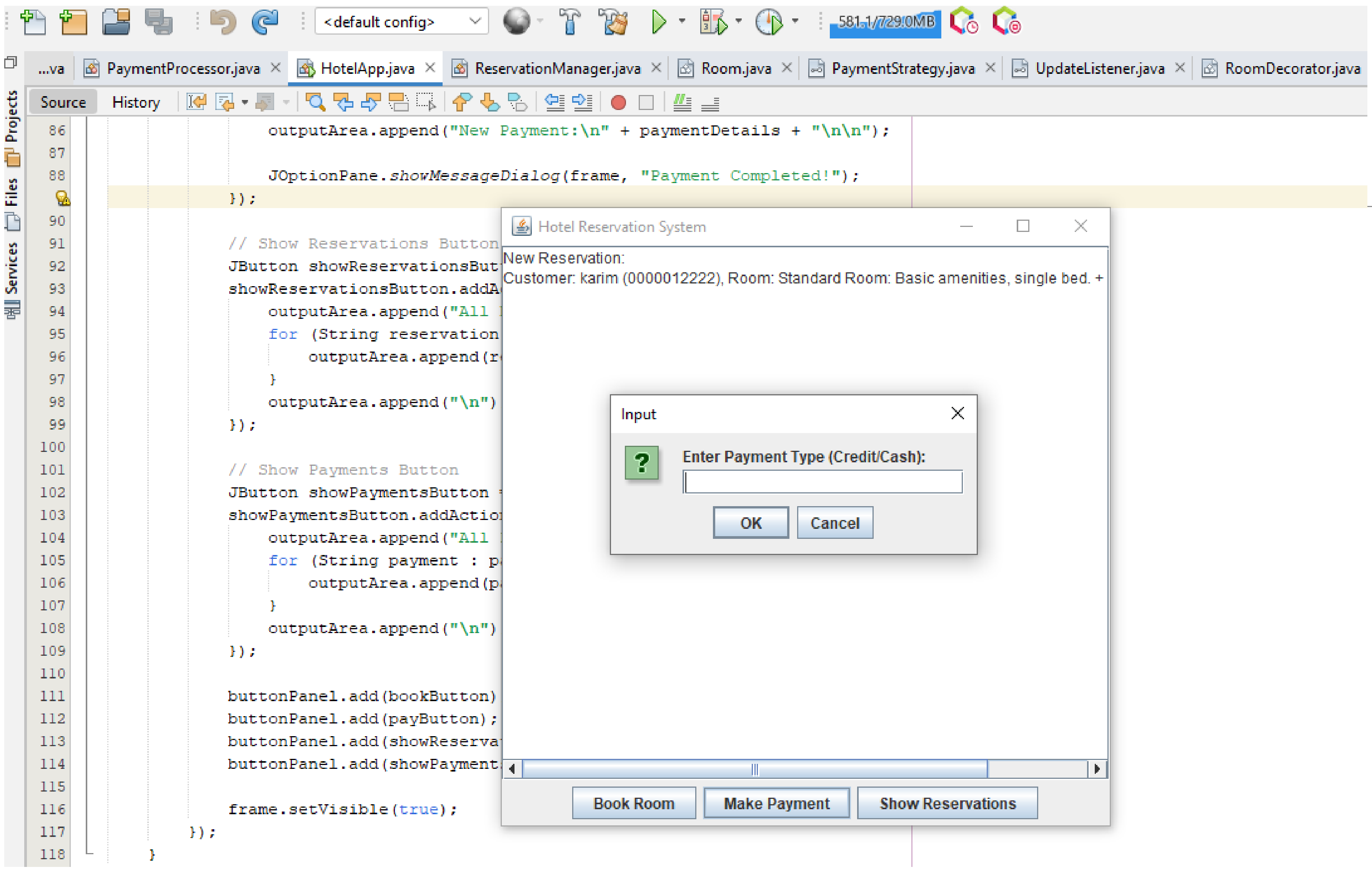
Enter Customer Phone: 0000012222



```
                outputArea.append("New Payment:\n" + paymentDetails + "\n\n");

                JOptionPane.showMessageDialog(frame, "Payment Completed!");
            });

            // Show Reservations Button
            JButton showReservationsBut
            showReservationsButton.addA
                outputArea.append("All 
                for (String reservation
                    outputArea.append(r
                }
                outputArea.append("\n")
            });

            // Show Payments Button
            JButton showPaymentsButton 
            showPaymentsButton.addActio
                outputArea.append("All 
                for (String payment : p
                    outputArea.append(p
                }
                outputArea.append("\n")
            });

            buttonPanel.add(bookButton)
            buttonPanel.add(payButton);
            buttonPanel.add(showReserva
            buttonPanel.add(showPayment

            frame.setVisible(true);
        });
    }
}
```

```
New Reservation:
Customer: karim (0000012222), Room: Standard Room: Basic amenities, single bed. +
```

Room Booked!



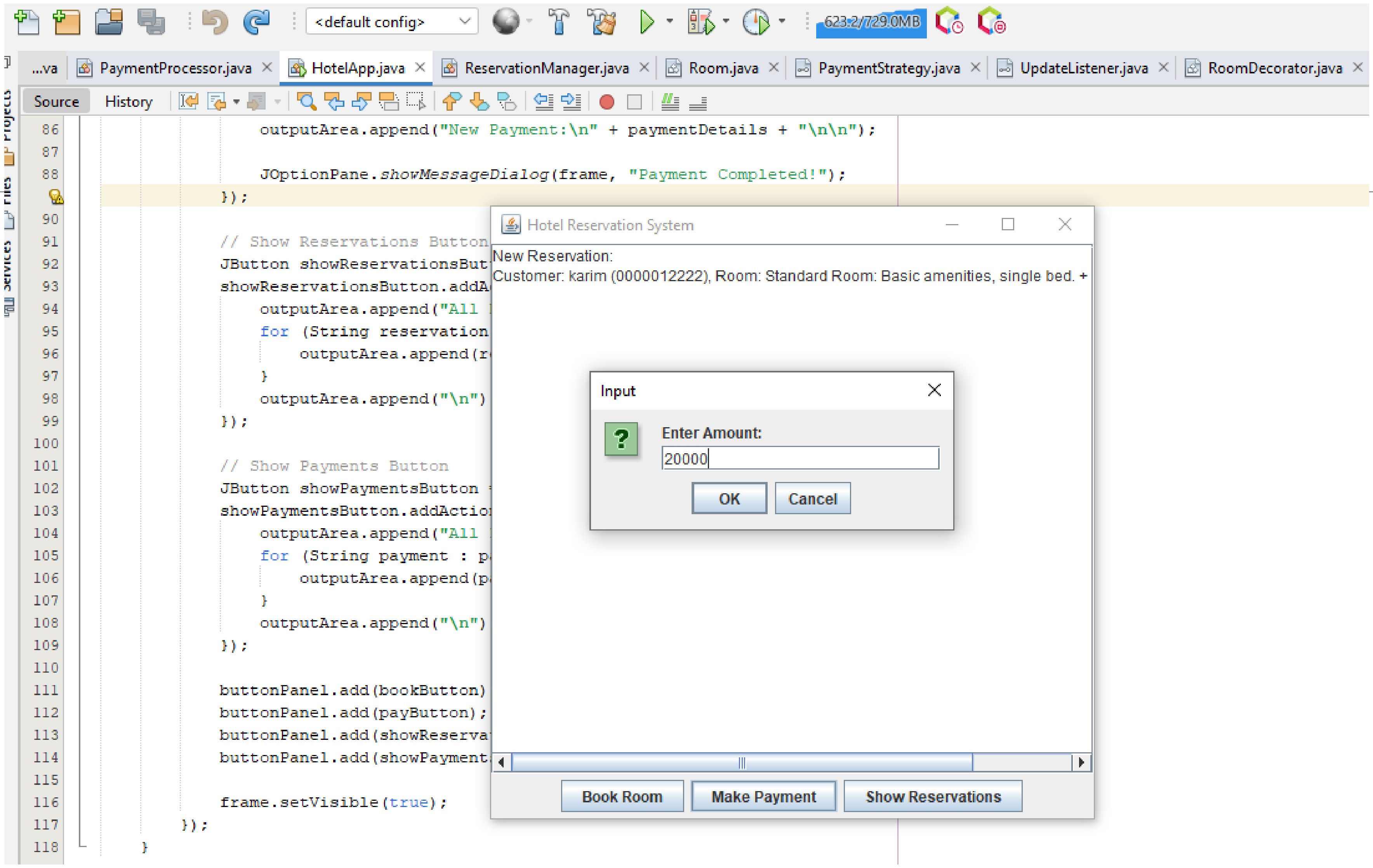
```
                outputArea.append("New Payment:\n" + paymentDetails + "\n\n");

                JOptionPane.showMessageDialog(frame, "Payment Completed!");
            });

            // Show Reservations Button
            JButton showReservationsBut
            showReservationsButton.addA
                outputArea.append("All 
                for (String reservation
                    outputArea.append(r
                }
                outputArea.append("\n")
            });

            // Show Payments Button
            JButton showPaymentsButton 
            showPaymentsButton.addActio
                outputArea.append("All 
                for (String payment : p
                    outputArea.append(p
                }
                outputArea.append("\n")
            });

            buttonPanel.add(bookButton)
            buttonPanel.add(payButton);
            buttonPanel.add(showReserva
            buttonPanel.add(showPayment

            frame.setVisible(true);
        });
    }
```

Hotel Reservation System

```
New Reservation:
Customer: karim (0000012222), Room: Standard Room: Basic amenities, single bed. +
```

Input

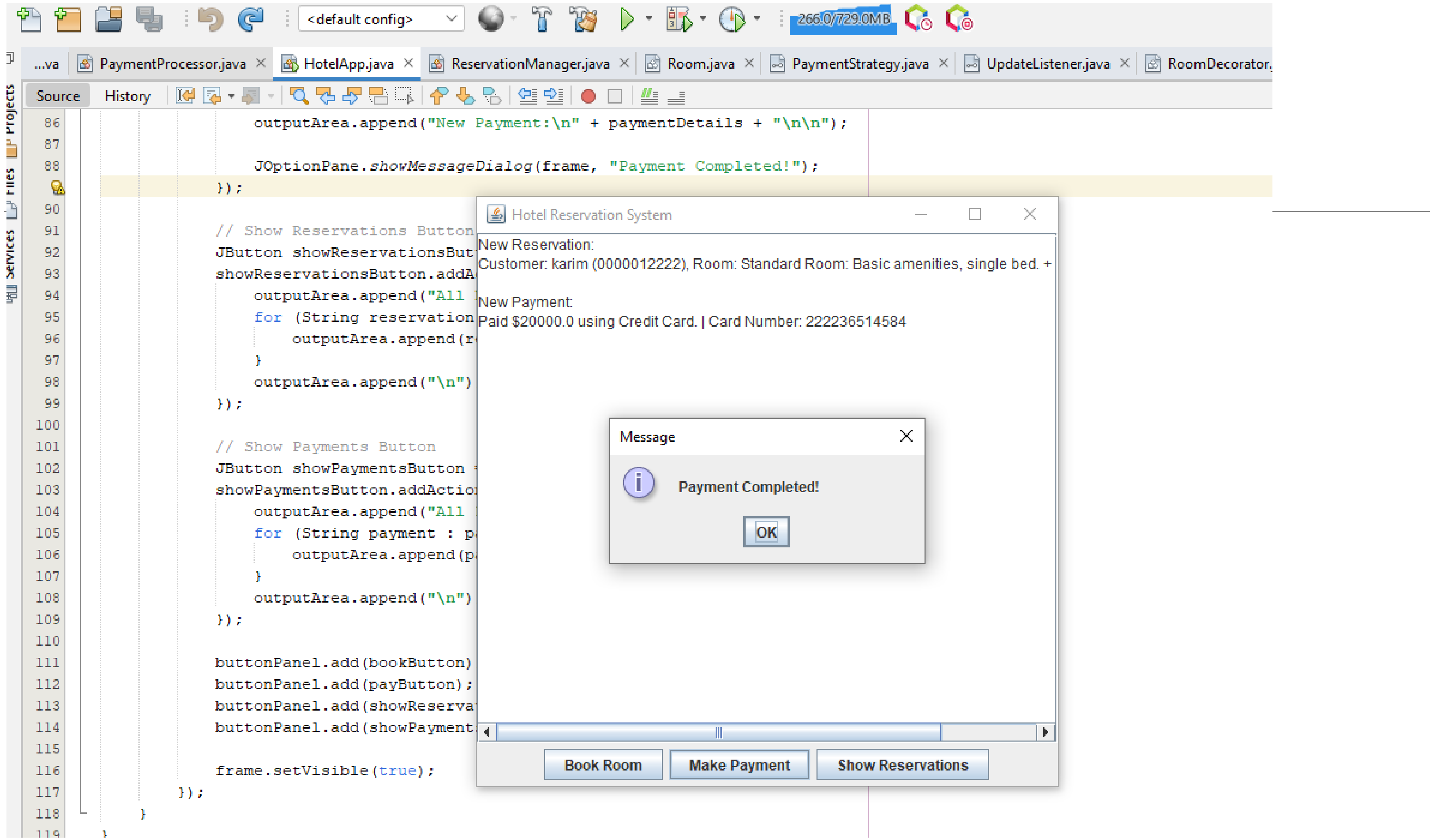
Enter Payment Type (Credit/Cash):



```
                outputArea.append("New Payment:\n" + paymentDetails + "\n\n");

                JOptionPane.showMessageDialog(frame, "Payment Completed!");
            });

            // Show Reservations Button
            JButton showReservationsBut
            showReservationsButton.addA
                outputArea.append("All 
                for (String reservation
                    outputArea.append(r
                }
                outputArea.append("\n")
            });

            // Show Payments Button
            JButton showPaymentsButton =
            showPaymentsButton.addActio
                outputArea.append("All 
                for (String payment : p
                    outputArea.append(p
                }
                outputArea.append("\n")
            });

            buttonPanel.add(bookButton)
            buttonPanel.add(payButton);
            buttonPanel.add(showReserva
            buttonPanel.add(showPayment

            frame.setVisible(true);
        });
    }
```

```
New Reservation:
Customer: karim (0000012222), Room: Standard Room: Basic amenities, single bed. +
```

```
Enter Amount:
20000
```



```java
                outputArea.append("New Payment:\n" + paymentDetails + "\n\n");

                JOptionPane.showMessageDialog(frame, "Payment Completed!");
            });

            // Show Reservations Button
            JButton showReservationsBut
            showReservationsButton.addA
                outputArea.append("All 
                for (String reservation
                    outputArea.append(r
                }
                outputArea.append("\n")
            });

            // Show Payments Button
            JButton showPaymentsButton 
            showPaymentsButton.addActio
                outputArea.append("All 
                for (String payment : p
                    outputArea.append(p
                }
                outputArea.append("\n")
            });

            buttonPanel.add(bookButton)
            buttonPanel.add(payButton);
            buttonPanel.add(showReserva
            buttonPanel.add(showPayment

            frame.setVisible(true);
        });
    }
```

```
New Reservation:
Customer: karim (0000012222), Room: Standard Room: Basic amenities, single bed. +

New Payment:
Paid $20000.0 using Credit Card. | Card Number: 222236514584
```

Payment Completed!

Thanks……