```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import java.util.ArrayList;

class ParkingSlot {
    private final int id;
    private boolean isOccupied;
    private double hourlyRate;

    public ParkingSlot(int id, double
initialRate) {
        this.id = id;
        this.isOccupied = false;
        this.hourlyRate = initialRate;
    }

    public int getId() {
        return id;
```

```
    }

    public boolean isOccupied() {
        return isOccupied;
    }

    public void occupy() {
        isOccupied = true;
    }

    public void vacate() {
        isOccupied = false;
    }

    public double getHourlyRate() {
        return hourlyRate;
    }

    public void adjustRate(double
rateChange) {
        hourlyRate += rateChange;
```

```
    }

    @Override
    public String toString() {
        return "Slot " + id + " (Rate: $" +
hourlyRate + "/hr, " +
                (isOccupied ? "Occupied" :
"Available") + ")";
    }
}

class ParkingLot {
    private final List<ParkingSlot>
parkingSlots = new ArrayList<>();
    private final double rateChangeFactor =
1.5;

    public ParkingLot(int totalSlots, double
initialRate) {
        for (int i = 1; i <= totalSlots; i++) {
            parkingSlots.add(new ParkingSlot(i,
```

```
initialRate));
        }
    }

    public List<ParkingSlot>
getAvailableSlots() {
        List<ParkingSlot> availableSlots = new
ArrayList<>();
        for (ParkingSlot slot : parkingSlots) {
            if (!slot.isOccupied()) {
                availableSlots.add(slot);
            }
        }
        return availableSlots;
    }

    public boolean makeReservation(int
slotId) {
        for (ParkingSlot slot : parkingSlots) {
            if (slot.getId() == slotId && !
slot.isOccupied()) {
```

```
            slot.occupy();
            adjustPricing();
            return true;
        }
    }
    return false;
}

public void releaseSlot(int slotId) {
    for (ParkingSlot slot : parkingSlots) {
        if (slot.getId() == slotId &&
slot.isOccupied()) {
            slot.vacate();
            adjustPricing();
            return;
        }
    }
}

private void adjustPricing() {
    double occupancyRate = (double)
```

```
getOccupiedCount() / parkingSlots.size();
        double rateChange = occupancyRate > 0.8 ? rateChangeFactor : -rateChangeFactor;

        for (ParkingSlot slot : parkingSlots) {
            if (!slot.isOccupied()) {
                slot.adjustRate(rateChange);
            }
        }
    }

    private int getOccupiedCount() {
        int count = 0;
        for (ParkingSlot slot : parkingSlots) {
            if (slot.isOccupied()) {
                count++;
            }
        }
        return count;
    }
```

```
    public List<ParkingSlot> getAllSlots() {
        return parkingSlots;
    }
}

public class ParkingSystemSwing extends
JFrame {
    private final ParkingLot parkingLot;
    private final JTextArea statusArea;

    public ParkingSystemSwing() {
        parkingLot = new ParkingLot(10, 10.0);
        statusArea = new JTextArea(10, 30);
        statusArea.setEditable(false);

        setTitle("Parking System");

setDefaultCloseOperation(JFrame.EXIT_ON
_CLOSE);
        setLayout(new BorderLayout());
```

```
        JPanel controlPanel = new JPanel();
        JButton reserveButton = new
JButton("Reserve Slot");
        JButton releaseButton = new
JButton("Release Slot");
        controlPanel.add(reserveButton);
        controlPanel.add(releaseButton);

        reserveButton.addActionListener(new
ReserveActionListener());
        releaseButton.addActionListener(new
ReleaseActionListener());

        add(new JScrollPane(statusArea),
BorderLayout.CENTER);
        add(controlPanel,
BorderLayout.SOUTH);

        updateStatus();
        pack();
```

```
        setVisible(true);
    }

    private void updateStatus() {
        StringBuilder status = new
StringBuilder();
        for (ParkingSlot slot :
parkingLot.getAllSlots()) {
            status.append(slot).append("\n");
        }
        statusArea.setText(status.toString());
    }

    private class ReserveActionListener
implements ActionListener {
        @Override
        public void
actionPerformed(ActionEvent e) {
            String input =
JOptionPane.showInputDialog(
                    ParkingSystemSwing.this, "Enter
```

```
Slot ID to Reserve:");
            try {
                int slotId = Integer.parseInt(input);
                if
(parkingLot.makeReservation(slotId)) {

JOptionPane.showMessageDialog(ParkingS
ystemSwing.this,
                        "Reservation confirmed for
Slot " + slotId + ".");
                } else {

JOptionPane.showMessageDialog(ParkingS
ystemSwing.this,
                        "Reservation failed. Slot may
already be occupied.");
                }
                updateStatus();
            } catch (NumberFormatException ex)
{
```

```
JOptionPane.showMessageDialog(ParkingSystemSwing.this,
                    "Invalid input. Please enter a numeric Slot ID.");
        }
    }
  }

   private class ReleaseActionListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String input = JOptionPane.showInputDialog(
                    ParkingSystemSwing.this, "Enter Slot ID to Release:");
            try {
                int slotId = Integer.parseInt(input);
                parkingLot.releaseSlot(slotId);
```

```
JOptionPane.showMessageDialog(ParkingSystemSwing.this,
                    "Slot " + slotId + " released.");
                updateStatus();
            } catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(ParkingSystemSwing.this,
                    "Invalid input. Please enter a numeric Slot ID.");
            }
        }
    }

    public static void main(String[] args) {

SwingUtilities.invokeLater(ParkingSystemSwing::new);
    }
}
```