QUESTION:

Design and implement a console-based Property Rental system to list properties, manage tenants, create leases, collect rent, and generate statements using OOP in Java.

Requirements:

1. Create at least 4 classes:
   - o Property – propertyId, address, type (Apartment/House), rent, status.
   - o Tenant – tenantId, name, contact, deposit, activeLeases.
   - o Lease – leaseId, propertyId, tenantId, startDate, endDate, rentCycle, status.
   - o RentalService – lists properties, creates leases, records payments, statements.

2. Each class must include:
   - o ≥4 instance/static variables.
   - o A constructor to initialize values.
   - o ≥5 methods (getters/setters, listProperty(), leaseProperty(), collectRent(), terminateLease()).

3. Demonstrate OOPS Concepts:
   - o Inheritance → Apartment/House extend Property with fees/rules.
   - o Method Overloading → collectRent() for full/partial/late fee scenarios.
   - o Method Overriding → property-specific maintenanceCharge()/display.
   - o Polymorphism → compute charges from List<Property> with overrides.
   - o Encapsulation → protect lease state transitions and tenant deposit.

4. Write a Main class (RentalAppMain) to test:
   - o Add properties and tenants, create leases.
   - o Record rent payments, handle late fees, terminate leases.
   - o Print monthly income and occupancy reports.

JAVA CODE:

Property.java

```
package rental;

public class Property {
    protected int propertyId;
    protected String address;
    protected String type; // Apartment or House
    protected double rent;
    protected boolean status; // true = rented, false = available

    public Property(int propertyId, String address, String type, double rent) {
        this.propertyId = propertyId;
        this.address = address;
        this.type = type;
        this.rent = rent;
        this.status = false;
    }

    // Getters and Setters
    public int getPropertyId() { return propertyId; }
    public String getAddress() { return address; }
    public String getType() { return type; }
    public double getRent() { return rent; }
    public boolean isRented() { return status; }
    public void setStatus(boolean status) { this.status = status; }

    // Methods
    public void display() {
        System.out.println("Property ID: " + propertyId + ", Address: " + address +
                ", Type: " + type + ", Rent: " + rent + ", Status: " + (status ? "Rented" :
"Available"));
    }

    public double maintenanceCharge() {
        return 100.0;
    }
}
```

Apartment.java

```
package rental;

public class Apartment extends Property {
    private boolean hasGym;
    private boolean hasPool;

    public Apartment(int propertyId, String address, double rent, boolean hasGym,
boolean hasPool) {
        super(propertyId, address, "Apartment", rent);
        this.hasGym = hasGym;
        this.hasPool = hasPool;
    }

    @Override
    public void display() {
        super.display();
        System.out.println("Gym: " + hasGym + ", Pool: " + hasPool);
    }

    @Override
    public double maintenanceCharge() {
        return 150.0;
    }
}
```

House.java

```
package rental;

public class House extends Property {
    private int gardenSize; // in sq meters
    private boolean hasGarage;

    public House(int propertyId, String address, double rent, int gardenSize,
boolean hasGarage) {
        super(propertyId, address, "House", rent);
        this.gardenSize = gardenSize;
        this.hasGarage = hasGarage;
```

```
    }

    @Override
    public void display() {
        super.display();
        System.out.println("Garden Size: " + gardenSize + " sq.m, Garage: " +
hasGarage);
    }

    @Override
    public double maintenanceCharge() {
        return 120.0 + gardenSize * 0.5;
    }
}
```

Tenant.java

```
package rental;

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

public class Tenant {
    private int tenantId;
    private String name;
    private String contact;
    private double deposit;
    private List<Lease> activeLeases;

    public Tenant(int tenantId, String name, String contact, double deposit) {
        this.tenantId = tenantId;
        this.name = name;
        this.contact = contact;
        this.deposit = deposit;
        this.activeLeases = new ArrayList<>();
    }

    // Getters
    public int getTenantId() { return tenantId; }
    public String getName() { return name; }
```

```
    public String getContact() { return contact; }
    public double getDeposit() { return deposit; }
    public List<Lease> getActiveLeases() { return activeLeases; }

    // Setters
    public void setName(String name) { this.name = name; }
    public void setContact(String contact) { this.contact = contact; }
    public void setDeposit(double deposit) { this.deposit = deposit; }

    // Lease management
    public void addLease(Lease lease) { activeLeases.add(lease); }
    public void removeLease(Lease lease) { activeLeases.remove(lease); }

    // Display tenant details
    public void display() {
        System.out.println("Tenant ID: " + tenantId + ", Name: " + name + ",
Contact: " + contact + ", Deposit: " + deposit);
        if (!activeLeases.isEmpty()) {
            System.out.println("Active Leases:");
            for (Lease lease : activeLeases) {
                lease.display();
            }
        }
    }
}
```

Lease.java

```
package rental;

import java.time.LocalDate;

public class Lease {
    private int leaseId;
    private Property property;
    private Tenant tenant;
    private LocalDate startDate;
    private LocalDate endDate;
    private String rentCycle;
    private boolean status; // true = active, false = terminated
```

```
    public Lease(int leaseId, Property property, Tenant tenant, LocalDate
startDate, LocalDate endDate, String rentCycle) {
        this.leaseId = leaseId;
        this.property = property;
        this.tenant = tenant;
        this.startDate = startDate;
        this.endDate = endDate;
        this.rentCycle = rentCycle;
        this.status = true;
    }

    public void display() {
        System.out.println("Lease ID: " + leaseId + ", Property ID: " +
property.getPropertyId() +
                ", Tenant: " + tenant.getName() + ", Start: " + startDate + ", End: " +
endDate +
                ", Rent Cycle: " + rentCycle + ", Status: " + (status ? "Active" :
"Terminated"));
    }

    public Tenant getTenant() { return tenant; }
    public Property getProperty() { return property; }
    public boolean isActive() { return status; }

    public void terminateLease() {
        status = false;
        property.setStatus(false);
        tenant.removeLease(this);
    }
}
```

RentalService.java

```
package rental;

import java.time.LocalDate;
import java.util.List;
import java.util.ArrayList;
```

```
public class RentalService {
    private List<Property> properties;
    private List<Tenant> tenants;
    private List<Lease> leases;

    public RentalService() {
        properties = new ArrayList<>();
        tenants = new ArrayList<>();
        leases = new ArrayList<>();
    }

    // Add properties & tenants
    public void addProperty(Property p) { properties.add(p); }
    public void addTenant(Tenant t) { tenants.add(t); }

    // List all properties
    public void listProperties() {
        for(Property p : properties) {
            p.display();
            System.out.println("Maintenance Charge: " + p.maintenanceCharge());
            System.out.println("-------------");
        }
    }

    // Lease property
    public Lease leaseProperty(int leaseId, int propertyId, int tenantId, LocalDate
start, LocalDate end, String cycle) {
        Property prop = properties.stream().filter(p -> p.getPropertyId() ==
propertyId && !p.isRented()).findFirst().orElse(null);
        Tenant ten = tenants.stream().filter(t -> t.getTenantId() ==
tenantId).findFirst().orElse(null);

        if(prop != null && ten != null) {
            Lease lease = new Lease(leaseId, prop, ten, start, end, cycle);
            prop.setStatus(true);
            ten.addLease(lease);
            leases.add(lease);
            return lease;
        }
```

```
        return null;
    }

    // Collect Rent (Method Overloading)
    public void collectRent(Lease lease) {
        System.out.println("Collected full rent of " + lease.getProperty().getRent() +
" from " + lease.getTenant().getName());
    }

    public void collectRent(Lease lease, double amount) {
        System.out.println("Collected partial rent of " + amount + " from " +
lease.getTenant().getName());
    }

    public void collectRent(Lease lease, double amount, double lateFee) {
        System.out.println("Collected " + amount + " with late fee " + lateFee + "
from " + lease.getTenant().getName());
    }

    // Generate Monthly Income Report
    public void generateIncomeReport() {
        double total = 0;
        for(Lease l : leases) {
            if(l.isActive()) total += l.getProperty().getRent();
        }
        System.out.println("Monthly Income: " + total);
    }

    // Generate Occupancy Report
    public void generateOccupancyReport() {
        long rented = properties.stream().filter(p -> p.isRented()).count();
        System.out.println("Total Properties: " + properties.size() + ", Occupied: " +
rented + ", Available: " + (properties.size() - rented));
    }
}
```

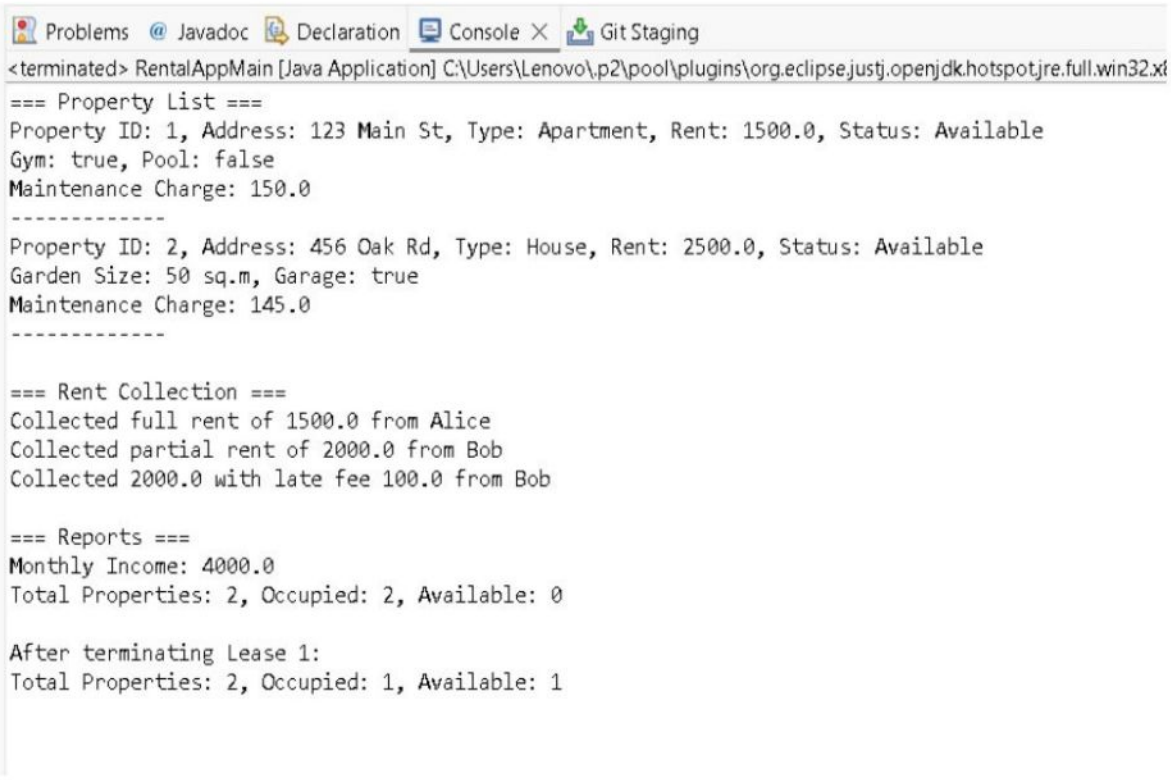
RentalAppMain.java

```
package rental;
```

```
import java.time.LocalDate;

public class RentalAppMain {
    public static void main(String[] args) {
        RentalService service = new RentalService();

        // Add properties
        service.addProperty(new Apartment(1, "123 Main St", 1500, true, false));
        service.addProperty(new House(2, "456 Oak Rd", 2500, 50, true));

        // Add tenants
        service.addTenant(new Tenant(1, "Alice", "1234567890", 500));
        service.addTenant(new Tenant(2, "Bob", "0987654321", 700));

        // List properties
        System.out.println("=== Property List ===");
        service.listProperties();

        // Create leases
        Lease lease1 = service.leaseProperty(1, 1, 1, LocalDate.of(2025, 9, 1),
LocalDate.of(2026, 8, 31), "Monthly");
        Lease lease2 = service.leaseProperty(2, 2, 2, LocalDate.of(2025, 9, 1),
LocalDate.of(2026, 8, 31), "Monthly");

        // Collect rents
        System.out.println("\n=== Rent Collection ===");
        service.collectRent(lease1);
        service.collectRent(lease2, 2000);
        service.collectRent(lease2, 2000, 100);

        // Generate Reports
        System.out.println("\n=== Reports ===");
        service.generateIncomeReport();
        service.generateOccupancyReport();

        // Terminate a lease
        lease1.terminateLease();
        System.out.println("\nAfter terminating Lease 1:");
```

```
        service.generateOccupancyReport();
    }
}
```

OUTPUT:



GITHUB REPOSITORY LINK:

https://github.com/Guna1213/PropertyRentalSystem