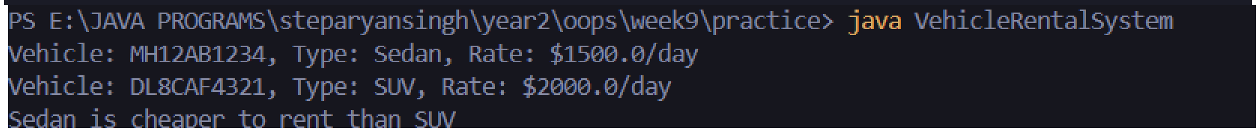
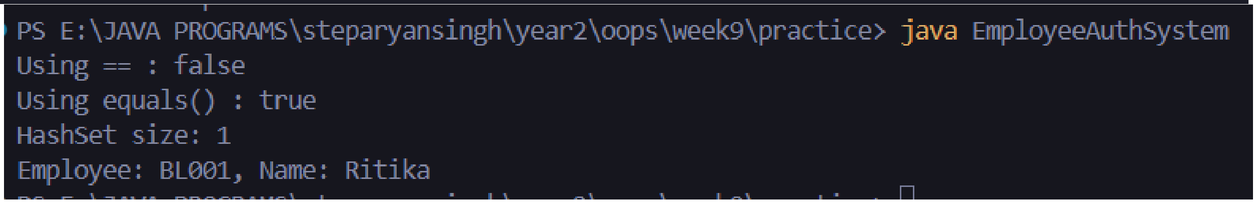
```java
// File: VehicleRentalSystem.java

class Vehicle {
    private String registrationNo;
    private String type;
    private double ratePerDay;

    // Constructor initializing all fields
    public Vehicle(String registrationNo, String type, double ratePerDay)
{
        this.registrationNo = registrationNo;
        this.type = type;
        this.ratePerDay = ratePerDay;
    }

    // Getters
    public String getRegistrationNo() {
        return registrationNo;
    }

    public String getType() {
        return type;
    }

    public double getRatePerDay() {
        return ratePerDay;
    }

    // Override toString()
    @Override
    public String toString() {
        return "Vehicle: " + registrationNo + ", Type: " + type + ", Rate:
$" + ratePerDay + "/day";
    }
}

public class VehicleRentalSystem {
    public static void main(String[] args) {
        // 1. Create Vehicle("MH12AB1234", "Sedan", 1500)
        Vehicle v1 = new Vehicle("MH12AB1234", "Sedan", 1500);
        // 2. Print the Vehicle object and observe output
        System.out.println(v1);

        // 3. Create another vehicle and compare
        Vehicle v2 = new Vehicle("DL8CAF4321", "SUV", 2000);
        System.out.println(v2);

        // Optional comparison logic
        if (v1.getRatePerDay() < v2.getRatePerDay()) {
            System.out.println(v1.getType() + " is cheaper to rent than "
+ v2.getType());
        } else {
            System.out.println(v2.getType() + " is cheaper to rent than "
+ v1.getType());
        }
    }
}
```



```
PS E:\JAVA PROGRAMS\stepharyansingh\year2\oops\week9\practice> java VehicleRentalSystem
Vehicle: MH12AB1234, Type: Sedan, Rate: $1500.0/day
Vehicle: DL8CAF4321, Type: SUV, Rate: $2000.0/day
Sedan is cheaper to rent than SUV
```

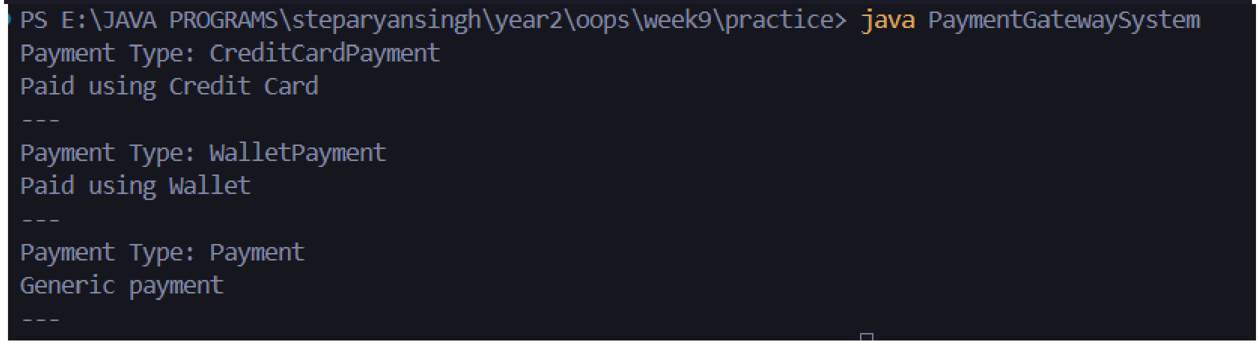
```java
// File: EmployeeAuthSystem.java

import java.util.HashSet;

class Employee {
    private String empCode;
    private String name;

    public Employee(String empCode, String name) {
        this.empCode = empCode;
        this.name = name;
    }

    // Override equals(): same empCode means same employee
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Employee other = (Employee) obj;
        return empCode != null && empCode.equals(other.empCode);
    }

    // Override hashCode() based on empCode
    @Override
    public int hashCode() {
        return empCode != null ? empCode.hashCode() : 0;
    }

    // toString() showing both fields
    @Override
    public String toString() {
        return "Employee: " + empCode + ", Name: " + name;
    }
}

public class EmployeeAuthSystem {
    public static void main(String[] args) {
        // 1. Create two employees with same empCode
        Employee e1 = new Employee("BL001", "Ritika");
        Employee e2 = new Employee("BL001", "Ritika S.");

        // 2. Compare using == and equals()
        System.out.println("Using == : " + (e1 == e2)); // false
        System.out.println("Using equals() : " + e1.equals(e2)); // true

        // 3. Test using HashSet
        HashSet<Employee> set = new HashSet<>();
        set.add(e1);
        set.add(e2); // Should not be added if equals/hashCode are correct

        System.out.println("HashSet size: " + set.size()); // Should be 1
        for (Employee emp : set) {
            System.out.println(emp);
        }
    }
}
```



```
PS E:\JAVA PROGRAMS\stepAryansingh\year2\oops\week9\practice> java EmployeeAuthSystem
Using == : false
Using equals() : true
HashSet size: 1
Employee: BL001, Name: Ritika
```

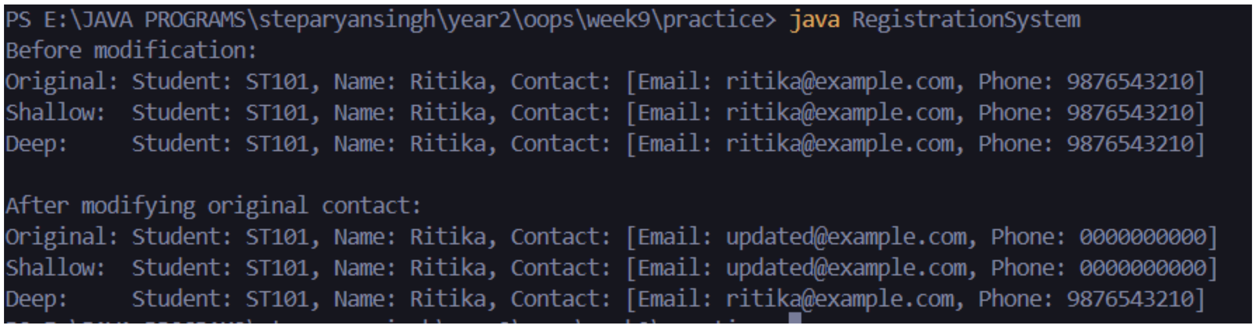
```java
// File: PaymentGatewaySystem.java

class Payment {
    public void pay() {
        System.out.println("Generic payment");
    }
}

class CreditCardPayment extends Payment {
    @Override
    public void pay() {
        System.out.println("Paid using Credit Card");
    }
}

class WalletPayment extends Payment {
    @Override
    public void pay() {
        System.out.println("Paid using Wallet");
    }
}
```

```java
public class PaymentGatewaySystem {
    public static void main(String[] args) {
        // 1. Create array of Payment references
        Payment[] payments = {
            new CreditCardPayment(),
            new WalletPayment(),
            new Payment()
        };

        // 2. Loop, call getClass().getSimpleName(), and pay()
        for (Payment p : payments) {
            System.out.println("Payment Type: " +
p.getClass().getSimpleName());
            p.pay();
            System.out.println("---");
        }
    }
}
```



```
PS E:\JAVA PROGRAMS\stepAryansingh\year2\oops\week9\practice> java PaymentGatewaySystem
Payment Type: CreditCardPayment
Paid using Credit Card
---
Payment Type: WalletPayment
Paid using Wallet
---
Payment Type: Payment
Generic payment
---
```

```java
// File: RegistrationSystem.java

class ContactInfo implements Cloneable {
    String email;
    String phone;

    public ContactInfo(String email, String phone) {
        this.email = email;
        this.phone = phone;
    }

    @Override
    protected Object clone() throws CloneNotSupportedException {
        return super.clone(); // shallow copy is fine here
    }

    @Override
    public String toString() {
        return "Email: " + email + ", Phone: " + phone;
    }
}

class Student implements Cloneable {
    String id;
    String name;
    ContactInfo contact;

    public Student(String id, String name, ContactInfo contact) {
        this.id = id;
        this.name = name;
        this.contact = contact;
    }

    // Shallow copy: contact is shared
    public Student shallowClone() throws CloneNotSupportedException {
        return (Student) super.clone();
    }

    // Deep copy: contact is cloned separately
    public Student deepClone() throws CloneNotSupportedException {
        Student cloned = (Student) super.clone();
        cloned.contact = (ContactInfo) contact.clone();
        return cloned;
    }

    @Override
    public String toString() {
        return "Student: " + id + ", Name: " + name + ", Contact: [" +
contact + "]";
    }
}

public class RegistrationSystem {
    public static void main(String[] args) throws
CloneNotSupportedException {
        ContactInfo contact = new ContactInfo("ritika@example.com",
"9876543210");
        Student original = new Student("ST101", "Ritika", contact);

        Student shallow = original.shallowClone();
        Student deep = original.deepClone();

        System.out.println("Before modification:");
        System.out.println("Original: " + original);
        System.out.println("Shallow:  " + shallow);
        System.out.println("Deep:     " + deep);

        // Modify contact info
        contact.email = "updated@example.com";
        contact.phone = "0000000000";

        System.out.println("\nAfter modifying original contact:");
        System.out.println("Original: " + original);
        System.out.println("Shallow:  " + shallow); // reflects changes
        System.out.println("Deep:     " + deep);    // remains unchanged
    }
}
```



```
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week9\practice> java RegistrationSystem
Before modification:
Original: Student: ST101, Name: Ritika, Contact: [Email: ritika@example.com, Phone: 9876543210]
Shallow:  Student: ST101, Name: Ritika, Contact: [Email: ritika@example.com, Phone: 9876543210]
Deep:     Student: ST101, Name: Ritika, Contact: [Email: ritika@example.com, Phone: 9876543210]

After modifying original contact:
Original: Student: ST101, Name: Ritika, Contact: [Email: updated@example.com, Phone: 0000000000]
Shallow:  Student: ST101, Name: Ritika, Contact: [Email: updated@example.com, Phone: 0000000000]
Deep:     Student: ST101, Name: Ritika, Contact: [Email: ritika@example.com, Phone: 9876543210]
```