QUESTION:01

```
public class BankAccount {
    protected String accountNumber;
    protected String accountHolder;
    protected double balance;


    public BankAccount(String accountNumber, String accountHolder, double balance) {
        this.accountNumber = accountNumber;
        this.accountHolder = accountHolder;
        this.balance = balance;
    }
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited TZS " + amount + ". New balance: TZS " + balance);
    }
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrew TZS " + amount + ". New balance: TZS " + balance);
```

```
        } else {
            System.out.println("Insufficient funds to withdraw " + amount);
        }
    }    public double getBalance() {
        return balance;
    }
    public void displayAccountInfo() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Account Holder: " + accountHolder);
        System.out.println("Balance: TZS " + balance);
    }
    public double calculateInterest() {
        return 0;
    }
}
public class SavingsAccount extends BankAccount {
    private double interestRate;
    public SavingsAccount(String accountNumber, String accountHolder, double balance, double
interestRate) {
        super(accountNumber, accountHolder, balance);
        this.interestRate = interestRate;
    }
    @Override
    public double calculateInterest() {
        return balance * interestRate;
    }
```

```
    @Override
    public void withdraw(double amount) {
        if (balance - amount >= 10000) {
            super.withdraw(amount);
        } else {
            System.out.println("Cannot withdraw. Minimum balance of TZS 10,000 must be maintained.");
        }
    }
    public void applyInterest() {
        double interest = calculateInterest();
        balance += interest;
        System.out.println("Interest of TZS " + interest + " applied. New balance: TZS " + balance);
    }
}
public class CurrentAccount extends BankAccount {
    private double overdraftLimit;
    public CurrentAccount(String accountNumber, String accountHolder, double balance, double
overdraftLimit) {
        super(accountNumber, accountHolder, balance);
        this.overdraftLimit = overdraftLimit;
    }

    @Override
    public double calculateInterest() {
        return 0;
    }
```

```
    @Override
    public void withdraw(double amount) {
        if (balance + overdraftLimit >= amount) {
            balance -= amount;
            System.out.println("Withdrew TZS " + amount + ". New balance: TZS " + balance);
        } else {
            System.out.println("Insufficient funds, including overdraft limit.");
        }
    }
    public boolean isOverdrawn() {
        return balance < 0;
    }
}
public class FixedDepositAccount extends BankAccount {
    private double interestRate;
    private int maturityMonths;
    private boolean isMatured;
    public FixedDepositAccount(String accountNumber, String accountHolder, double balance, double
interestRate, int maturityMonths) {
        super(accountNumber, accountHolder, balance);
        this.interestRate = interestRate;
        this.maturityMonths = maturityMonths;
        this.isMatured = false;
    }
    @Override
    public double calculateInterest() {
```

```
        return balance * interestRate * (maturityMonths / 12.0);
    }
    @Override
    public void withdraw(double amount) {
        if (isMatured) {
            super.withdraw(amount);
        } else {
            System.out.println("Account not matured yet. Cannot withdraw.");
        }
    }
    public void checkMaturity() {
        isMatured = true;
        System.out.println("Account is now matured.");
    }
    public double getMaturityAmount() {
        return balance + calculateInterest();
    }
}
import java.util.ArrayList;
public class Bank {
    private ArrayList<BankAccount> accounts = new ArrayList<>();
    public void addAccount(BankAccount account) {
        accounts.add(account);
    }
    public double getTotalDeposits() {
```

```
        double total = 0;
        for (BankAccount account : accounts) {
            total += account.getBalance();
        }
        return total;
    }
    public double getTotalInterest() {
        double totalInterest = 0;
        for (BankAccount account : accounts) {
            totalInterest += account.calculateInterest();
        }
        return totalInterest;
    }

    public void displayAllAccounts() {
        for (BankAccount account : accounts) {
            account.displayAccountInfo();
            System.out.println("----------------------------");
        }
    }
}
public class Exercise13_Polymorphism {
    public static void transferMoney(BankAccount from, BankAccount to, double amount) {
        if (from.getBalance() >= amount) {
            from.withdraw(amount);
```

```
            to.deposit(amount);
            System.out.println("Transferred TZS " + amount + " from " + from.accountNumber + " to " + to.accountNumber);
        } else {
            System.out.println("Insufficient balance for transfer.");
        }
    }

    public static void transferMoney(BankAccount from, BankAccount to, double amount, String description) {
        if (from.getBalance() >= amount) {
            from.withdraw(amount);
            to.deposit(amount);
            System.out.println("Transferred TZS " + amount + " from " + from.accountNumber + " to " + to.accountNumber + ". Description: " + description);
        } else {
            System.out.println("Insufficient balance for transfer.");
        }
    }

    public static void transferMoney(BankAccount from, String toAccountNumber, double amount, Bank bank) {
        BankAccount to = null;
        for (BankAccount account : bank.accounts) {
            if (account.accountNumber.equals(toAccountNumber)) {
                to = account;
                break;
```

```
        }
    }
    if (to != null) {
        transferMoney(from, to, amount);
    } else {
        System.out.println("Account not found: " + toAccountNumber);
    }
}

public static void main(String[] args) {
    System.out.println("=== BANKING SYSTEM TEST ===\n");
    SavingsAccount savings = new SavingsAccount("SAV001", "Ali Hassan", 500000, 0.05);
    CurrentAccount current = new CurrentAccount("CUR001", "Fatma Said", 1000000, 500000);
    FixedDepositAccount fixed = new FixedDepositAccount("FD001", "Omar Juma", 2000000, 0.08, 12);

    savings.displayAccountInfo();
    savings.deposit(100000);
    savings.withdraw(50000);
    savings.applyInterest();
    System.out.println("Interest earned: TZS " + savings.calculateInterest());

    Bank bank = new Bank();
    bank.addAccount(savings);
    bank.addAccount(current);
    bank.addAccount(fixed);
```

```
        transferMoney(savings, current, 50000);
        transferMoney(current, savings, 30000, "Rent payment");

        System.out.println("\n=== END OF TEST ===");
    }
}
```

QUESTION:02

```
/**
 * Write a description of class Circle here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Circle {
    private double radius;
    private String color;
    public Circle() {
        this.radius = 1.0;
        this.color = "red";
    }
    public Circle(double radius) {
        this.radius = radius;
        this.color = "red";
```

```
    }
    public Circle(double radius, String color) {
        this.radius = radius;
        this.color = color;
    }
    public double getRadius() {
        return radius;
    }
    public void setRadius(double radius) {
        this.radius = radius;
    }
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
    public double getArea() {
        return Math.PI * radius * radius;
    }
    @Override
    public String toString() {
        return "Circle[radius=" + radius + ", color=" + color + "]";
    }
}
```

```
public class Cylinder extends Circle {
    private double height;
    public Cylinder() {
        super();
        this.height = 1.0;
    }
    public Cylinder(double radius) {
        super(radius);
        this.height = 1.0;
    }
    public Cylinder(double radius, double height) {
        super(radius);
        this.height = height;
    }
    public Cylinder(double radius, double height, String color) {
        super(radius, color);
        this.height = height;
    }
    public double getHeight() {
        return height;
    }
    public void setHeight(double height) {
        this.height = height;
    }
    public double getVolume() {
```

```
        return getArea() * height;
    }
    @Override
    public String toString() {
        return "Cylinder[" + super.toString() + ", height=" + height + "]";
    }
}
public class Lab1_CircleCylinder {
    public static void main(String[] args) {
        System.out.println("========================================");
        System.out.println("  Lab 1: Circle and Cylinder Hierarchy");
        System.out.println("========================================\n");
        System.out.println("--- Section 1: Basic Object Creation ---");
        Circle c1 = new Circle(5.0, "blue");
        System.out.println("Circle: " + c1);
        System.out.println("Area: " + c1.getArea());
        Cylinder cy1 = new Cylinder(5.0, 10.0, "green");
        System.out.println("\nCylinder: " + cy1);
        System.out.println("Base Area: " + cy1.getArea());
        System.out.println("Volume: " + cy1.getVolume());
        System.out.println("\n--- Section 2: Upcasting ---");
        Circle c2 = new Cylinder(3.0, 7.0, "yellow");
        System.out.println("c2 is a: " + c2.getClass().getSimpleName());
        System.out.println("c2.toString(): " + c2);
        System.out.println("c2.getArea(): " + c2.getArea());
```

```
System.out.println("c2.getRadius(): " + c2.getRadius());
System.out.println("\n--- Section 3: Downcasting ---");
Circle c3 = new Cylinder(4.0, 8.0, "purple");
Cylinder cy2 = (Cylinder) c3;
System.out.println("After downcast: " + cy2);
System.out.println("Now we can call getVolume(): " + cy2.getVolume());

System.out.println("\n--- Section 4: instanceof Operator ---");

Circle[] shapes = {
    new Circle(2.0, "red"),
    new Cylinder(3.0, 5.0, "blue"),
    new Circle(4.0, "green"),
    new Cylinder(1.0, 10.0, "orange")
};

for (Circle shape : shapes) {
    System.out.println(shape);
    if (shape instanceof Cylinder) {
        Cylinder temp = (Cylinder) shape;
        System.out.println("  -> This is a Cylinder! Volume = " + temp.getVolume());
    } else {
        System.out.println("  -> This is just a Circle. Area = " + shape.getArea());
    }
}
```

```
        System.out.println("\n=======================================");
        System.out.println("  End of Lab 1");
        System.out.println("=======================================");
    }
}
```

QUESTION:03

```
public class Person {
    private String name;
    private String address;
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
    }
    public String getName() {
        return name;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    @Override
```

```
    public String toString() {
        return "Person[name=" + name + ", address=" + address + "]";
    }
}
public class Student extends Person {
    private String program;
    private int year;
    private double fee;
    public Student(String name, String address, String program, int year, double fee) {
        super(name, address);
        this.program = program;
        this.year = year;
        this.fee = fee;
    }
    public String getProgram() {
        return program;
    }
    public void setProgram(String program) {
        this.program = program;
    }

    public int getYear() {
        return year;
    }
```

```
    public void setYear(int year) {
        this.year = year;
    }

    public double getFee() {
        return fee;
    }

    public void setFee(double fee) {
        this.fee = fee;
    }

    @Override
    public String toString() {
        return "Student[Person[name=" + getName() + ", address=" + getAddress() + "], program=" +
program + ", year=" + year + ", fee=" + fee + "]";
    }
}
public class Staff extends Person {
    private String department;
    private double salary;
    public Staff(String name, String address, String department, double salary) {
        super(name, address);
        this.department = department;
        this.salary = salary;
    }
```

```
    public String getDepartment() {
        return department;
    }
    public void setDepartment(String department) {
        this.department = department;
    }
    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Staff[Person[name=" + getName() + ", address=" + getAddress() + "], department=" +
department + ", salary=" + salary + "]";
    }
}
public class Lab2_PersonStudentStaff {
    public static void main(String[] args) {
        System.out.println("=============================================");
        System.out.println("  Lab 2: Person, Student, and Staff Hierarchy");
        System.out.println("=============================================\n");
        System.out.println("--- Section 1: Creating Objects ---");
```

```
Person p1 = new Person("Amina Hassan", "Stonetown, Zanzibar");
System.out.println(p1);
Student s1 = new Student("Juma Ali", "Chwaka, Zanzibar", "BITA", 2, 1500000);
Student s2 = new Student("Fatma Omar", "Mbweni, Zanzibar", "BCS", 1, 1800000);
System.out.println(s1);
System.out.println(s2);
Staff staff1 = new Staff("Dr. Khalid Salum", "Vuga, Zanzibar", "SCCMS", 3500000);
System.out.println(staff1);
System.out.println("\n--- Section 2: Inheritance in Action ---");
System.out.println("Student name: " + s1.getName());
System.out.println("Student address: " + s1.getAddress());
System.out.println("Student program: " + s1.getProgram());

System.out.println("\nStaff name: " + staff1.getName());
System.out.println("Staff department: " + staff1.getDepartment());
s1.setAddress("Fumba, Zanzibar");
System.out.println("\nAfter address change: " + s1);
System.out.println("\n--- Section 3: Polymorphism ---");
Person[] people = {
   new Person("Bakari Juma", "Mwanakwerekwe, Zanzibar"),
   new Student("Zainab Moh'd", "Kiembe Samaki, Zanzibar", "BITA", 3, 1500000),
   new Student("Hassan Said", "Amani, Zanzibar", "BCS", 1, 1800000),
   new Staff("Prof. Mwanaisha Ali", "Mazizini, Zanzibar", "SCCMS", 4500000)
};
```

```
System.out.println("All people at SUZA:");
for (Person p : people) {
   System.out.println("  " + p);
System.out.println("\n--- Section 4: instanceof and Type Checking ---");
int studentCount = 0;
int staffCount = 0;

for (Person p : people) {
   if (p instanceof Student) {
      Student s = (Student) p;
      System.out.println(s.getName() + " is a Student in " + s.getProgram() + " Year " + s.getYear());
      studentCount++;
   } else if (p instanceof Staff) {
      Staff st = (Staff) p;
      System.out.println(st.getName() + " is Staff in " + st.getDepartment());
      staffCount++;
   } else {
      System.out.println(p.getName() + " is a Person (visitor/other)");
   }
}

System.out.println("\nSummary: " + studentCount + " students, " + staffCount + " staff members");
System.out.println("\n==========================================");
System.out.println("  End of Lab 2");
System.out.println("==========================================");
```

```
    }
}
```

QUESTION:04

```
/**
 * Write a description of class Point here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Point {
    private double x = 0.0;
    private double y = 0.0;
    public Point() {
    }
    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
    public double getX() {
        return x;
    }
    public void setX(double x) {
        this.x = x;
```

```
    }

    public double getY() {
        return y;
    }

    public void setY(double y) {
        this.y = y;
    }
    public void setXY(double x, double y) {
        this.x = x;
        this.y = y;
    }
    public double[] getXY() {
        return new double[]{x, y};
    }
    @Override
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}
public class MovablePoint extends Point {

    private double xSpeed = 0.0;
    private double ySpeed = 0.0;
```

```
public MovablePoint() {
}
public MovablePoint(double xSpeed, double ySpeed) {
   this.xSpeed = xSpeed;
   this.ySpeed = ySpeed;
}
public MovablePoint(double x, double y, double xSpeed, double ySpeed) {
   super(x, y);
   this.xSpeed = xSpeed;
   this.ySpeed = ySpeed;
}
public double getXSpeed() {
   return xSpeed;
}

public void setXSpeed(double xSpeed) {
   this.xSpeed = xSpeed;
}

public double getYSpeed() {
   return ySpeed;
}

public void setYSpeed(double ySpeed) {
```

```
        this.ySpeed = ySpeed;
    }
    public void setSpeed(double xSpeed, double ySpeed) {
        this.xSpeed = xSpeed;
        this.ySpeed = ySpeed;
    }
    public double[] getSpeed() {
        return new double[]{xSpeed, ySpeed};
    }
    public MovablePoint move() {
        setX(getX() + xSpeed);
        setY(getY() + ySpeed);
        return this;
    }
    @Override
    public String toString() {
        return super.toString() + " speed=(" + xSpeed + ", " + ySpeed + ")";
    }
}
public class Lab3_PointMovablePoint {
    public static void main(String[] args) {
        System.out.println("===============================================");
        System.out.println("  Lab 3: Point and MovablePoint");
        System.out.println("===============================================\n");
```

```
System.out.println("--- Section 1: Point Objects ---");

Point p1 = new Point();
System.out.println("Default point: " + p1);

Point p2 = new Point(3.0, 4.0);
System.out.println("Point at (3, 4): " + p2);

p2.setX(5.0);
p2.setY(6.0);
System.out.println("After setX(5), setY(6): " + p2);

double[] coords = p2.getXY();
System.out.println("getXY() = [" + coords[0] + ", " + coords[1] + "]");
System.out.println("\n--- Section 2: MovablePoint Objects ---");
MovablePoint mp1 = new MovablePoint(0.0, 0.0, 2.0, 3.0);
System.out.println("Initial position: " + mp1);
System.out.println("X coordinate: " + mp1.getX());
System.out.println("Y coordinate: " + mp1.getY());
System.out.println("\n--- Section 3: Movement ---");
System.out.println("Before move: " + mp1);
mp1.move();
System.out.println("After 1st move: " + mp1);
mp1.move();
System.out.println("After 2nd move: " + mp1);
```

```
mp1.move();
System.out.println("After 3rd move: " + mp1);
mp1.setSpeed(1.0, -1.0);
System.out.println("\nSpeed changed to (1.0, -1.0)");
mp1.move();
System.out.println("After move: " + mp1);
mp1.move();
System.out.println("After move: " + mp1);
System.out.println("\n--- Section 4: Polymorphism ---");
Point p3 = new MovablePoint(1.0, 1.0, 0.5, 0.5);
System.out.println("p3 (Point ref): " + p3);
System.out.println("p3 class: " + p3.getClass().getSimpleName());

   MovablePoint mp2 = (MovablePoint) p3;
   mp2.move();
   System.out.println("After downcast and move: " + mp2);
   System.out.println("p3 also changed: " + p3);
}
System.out.println("\n--- Section 5: Simple Movement Simulation ---");

MovablePoint[] points = {
   new MovablePoint(0.0, 0.0, 1.0, 1.0),
   new MovablePoint(10.0, 0.0, -1.0, 0.5),
   new MovablePoint(5.0, 5.0, 0.0, -2.0)
};
```

```
        System.out.println("Starting positions:");
        for (int i = 0; i < points.length; i++) {
            System.out.println("  Point " + (i + 1) + ": " + points[i]);
        }



        for (int step = 1; step <= 5; step++) {
            System.out.println("\nStep " + step + ":");
            for (int i = 0; i < points.length; i++) {
                points[i].move();
                System.out.println("  Point " + (i + 1) + ": " + points[i]);
            }
        }

        System.out.println("\n================================================");
        System.out.println("  End of Lab 3");
        System.out.println("================================================");
    }
}
```

QUESTION:05

```
/**
 * Write a description of class Shape here.
```

```
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Shape {
    private String color;
    private boolean filled;
    public Shape() {
        this.color = "red";
        this.filled = true;
    }
    public Shape(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public boolean isFilled() {
```

```
        return filled;
    }

    public void setFilled(boolean filled) {
        this.filled = filled;
    }
    @Override
    public String toString() {
        return "Shape[color=" + color + ", filled=" + filled + "]";
    }
}
public class Circle extends Shape {
    private double radius;
    public Circle() {
        this.radius = 1.0;
    }
    public Circle(double radius) {
        this.radius = radius;
    }

    public Circle(double radius, String color, boolean filled) {
        super(color, filled);
        this.radius = radius;
    }
    public double getRadius() {
```

```
        return radius;
    }
    public void setRadius(double radius) {
        this.radius = radius;
    }
    public double getArea() {
        return Math.PI * radius * radius;
    }

    public double getPerimeter() {
        return 2 * Math.PI * radius;
    }
    @Override
    public String toString() {
        return "Circle[Shape[color=" + getColor() + ", filled=" + isFilled() + "], radius=" + radius + "]";
    }
}
public class Rectangle extends Shape {
    private double width;
    private double length;
    public Rectangle() {
        this.width = 1.0;
        this.length = 1.0;
    }
    public Rectangle(double width, double length) {
```

```
    this.width = width;
    this.length = length;
}

public Rectangle(double width, double length, String color, boolean filled) {
    super(color, filled);
    this.width = width;
    this.length = length;
}
public double getWidth() {
    return width;
}

public void setWidth(double width) {
    this.width = width;
}

public double getLength() {
    return length;
}

public void setLength(double length) {
    this.length = length;
}
public double getArea() {
```

```
        return width * length;
    }

    public double getPerimeter() {
        return 2 * (width + length);
    }
    @Override
    public String toString() {
        return "Rectangle[Shape[color=" + getColor() + ", filled=" + isFilled() + "], width=" + width + ", length=" + length + "]";
    }
}
public class Square extends Rectangle {
    public Square() {
        super(1.0, 1.0);
    }
    public Square(double side) {
        super(side, side);
    }
    public Square(double side, String color, boolean filled) {
        super(side, side, color, filled);
    }
    public double getSide() {
        return getWidth();
    }
    public void setSide(double side) {
```

```
        setWidth(side);
        setLength(side);
    }

    @Override
    public void setWidth(double side) {
        super.setWidth(side);
        super.setLength(side);
    }
    @Override
    public void setLength(double side) {
        super.setWidth(side);
        super.setLength(side);
    }



    @Override
    public String toString() {
        return "Square[Rectangle[Shape[color=" + getColor() + ", filled=" + isFilled() + "], width=" +
getWidth() + ", length=" + getLength() + "]]";
    }
}
public class Lab4_ShapeHierarchy {
    public static void main(String[] args) {
        System.out.println("============================================");
        System.out.println("  Lab 4: Shape Hierarchy");
```

```
System.out.println("==========================================\n");



System.out.println("--- Section 1: Creating Objects ---");

Shape s1 = new Shape("yellow", false);
System.out.println(s1);

Circle c1 = new Circle(5.0, "blue", true);
System.out.println(c1);
System.out.println("  Area: " + c1.getArea());
System.out.println("  Perimeter: " + c1.getPerimeter());

Rectangle r1 = new Rectangle(4.0, 6.0, "green", true);
System.out.println(r1);
System.out.println("  Area: " + r1.getArea());
System.out.println("  Perimeter: " + r1.getPerimeter());

Square sq1 = new Square(5.0, "orange", false);
System.out.println(sq1);
System.out.println("  Area: " + sq1.getArea());
System.out.println("  Perimeter: " + sq1.getPerimeter());



System.out.println("\n--- Section 2: Square Invariant ---");
```

```
Square sq2 = new Square(3.0);
System.out.println("Initial: " + sq2);
System.out.println("Width: " + sq2.getWidth() + ", Length: " + sq2.getLength());

sq2.setWidth(7.0);
System.out.println("\nAfter setWidth(7.0):");
System.out.println("Width: " + sq2.getWidth() + ", Length: " + sq2.getLength());

sq2.setLength(9.0);
System.out.println("\nAfter setLength(9.0):");
System.out.println("Width: " + sq2.getWidth() + ", Length: " + sq2.getLength());



System.out.println("\n--- Section 3: Polymorphism ---");

Shape[] shapes = {
   new Circle(3.0, "red", true),
   new Rectangle(4.0, 5.0, "blue", false),
   new Square(6.0, "green", true)
};

double totalArea = 0;
for (Shape shape : shapes) {
   System.out.println(shape);
```

```
        if (shape instanceof Circle) {
            Circle c = (Circle) shape;
            System.out.println("  -> Circle area: " + c.getArea());
            totalArea
```

QUESTION:06

```
/**
 * Write a description of class Author here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Author {

   private String name;
   private String email;
   private char gender;
   public Author(String name, String email, char gender) {
      this.name = name;
      this.email = email;
      this.gender = gender;
   }
   public String getName() {
      return name;
```

```
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public char getGender() {
        return gender;
    }
    @Override
    public String toString() {
        return String.format("Author[name=%s, email=%s, gender=%c]", name, email, gender);
    }
}
public class Book {
    private String name;
    private Author author;
    private double price;
    private int qty;
    public Book(String name, Author author, double price) {
        this.name = name;
```

```
    this.author = author;
    this.price = price;
    this.qty = 0;
}

public Book(String name, Author author, double price, int qty) {
    this.name = name;
    this.author = author;
    this.price = price;
    this.qty = qty;
}
public String getName() {
    return name;
}

public Author getAuthor() {
    return author;
}

public String getAuthorName() {
    return author.getName();
}

public String getAuthorEmail() {
    return author.getEmail();
```

```
}

public char getAuthorGender() {
   return author.getGender();
}

public double getPrice() {
   return price;
}

public void setPrice(double price) {
   this.price = price;
}

public int getQty() {
   return qty;
}

public void setQty(int qty) {
   this.qty = qty;
}

@Override
public String toString() {
```

```
        return String.format("Book[name=%s, Author[%s], price=%.2f, qty=%d]", name, author, price, qty);
    }
}
public class Lab5_AuthorBookComposition {
    public static void main(String[] args) {
        System.out.println("============================================");
        System.out.println("  Lab 5: Author and Book (Composition)");
        System.out.println("============================================\n");



        System.out.println("--- Section 1: Creating Authors ---");

        Author author1 = new Author("Ali Sultan", "ali.sultan@suza.ac.tz", 'm');
        Author author2 = new Author("Mwanaisha Bakari", "mwanaisha.b@suza.ac.tz", 'f');
        Author author3 = new Author("Hamad Khamis", "hamad.k@gmail.com", 'm');

        System.out.println(author1);
        System.out.println(author2);
        System.out.println(author3);



        System.out.println("\n--- Section 2: Creating Books ---");

        Book book1 = new Book("Introduction to Java", author1, 35000, 50);
        Book book2 = new Book("Data Structures in Java", author2, 42000, 30);
```

```
System.out.println(book1);
System.out.println(book2);


System.out.println("\n--- Section 3: Accessing Through Composition ---");

System.out.println("Book: " + book1.getName());
System.out.println("Author name: " + book1.getAuthorName());
System.out.println("Author email: " + book1.getAuthorEmail());


Author bookAuthor = book1.getAuthor();
System.out.println("Author object: " + bookAuthor);


System.out.println("\n--- Section 4: Shared Author References ---");


Book book3 = new Book("Advanced Java Programming", author1, 55000, 20);
System.out.println("Book 1 author: " + book1.getAuthorName());
System.out.println("Book 3 author: " + book3.getAuthorName());
System.out.println("Same author? " + (book1.getAuthor() == book3.getAuthor()));
```

```
author1.setEmail("ali.sultan.new@suza.ac.tz");
System.out.println("\nAfter changing author1's email:");
System.out.println("Book 1 author email: " + book1.getAuthorEmail());
System.out.println("Book 3 author email: " + book3.getAuthorEmail());
System.out.println("Both changed! Because they share the same Author object.");



System.out.println("\n--- Section 5: Creating Book with Anonymous Author ---");

Book book4 = new Book(
   "Python for Beginners",
   new Author("Salma Haji", "salma.h@suza.ac.tz", 'f'),
   28000,
   100
);
System.out.println(book4);
System.out.println("Author: " + book4.getAuthorName());



System.out.println("\n--- Section 6: Book Inventory ---");

Book[] inventory = {book1, book2, book3, book4};

System.out.println("SUZA Bookshop Inventory:");
System.out.println(String.format("%-30s %-25s %10s %5s",
```

```
            "Title", "Author", "Price(TZS)", "Qty"));
        System.out.println("-".repeat(75));

        double totalValue = 0;
        for (Book book : inventory) {
            System.out.println(String.format("%-30s %-25s %,10.0f %5d",
                book.getName(), book.getAuthorName(),
                book.getPrice(), book.getQty()));
            totalValue += book.getPrice() * book.getQty();
        }
        System.out.println("-".repeat(75));
        System.out.println(String.format("Total inventory value: TZS %,.0f", totalValue));

        System.out.println("\n==========================================");
        System.out.println("  End of Lab 5");
        System.out.println("==========================================");
    }
}
```