Lab 6

```
// Base class for all accounts
class Account {
    protected int accountNumber;
    protected double balance;

    public Account(int accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited " + amount + " to account " + accountNumber);
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrew " + amount + " from account " + accountNumber);
        } else {
            System.out.println("Insufficient funds!");
        }
    }
}

// Derived class for savings accounts
class SavingsAccount extends Account {
    private double interestRate = 0.03; // 3% interest

    public SavingsAccount(int accountNumber, double balance) {
        super(accountNumber, balance);
    }

    public void calculateInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest added to savings account " + accountNumber + ": " + interest);
    }
}

// Derived class for checking accounts
class CheckingAccount extends Account {
```

```
    public CheckingAccount(int accountNumber, double balance) {
        super(accountNumber, balance);
    }
}

// Customer class
class Customer {
    private int customerId;
    private String name;
    private String address;
    private String phoneNumber;
    private SavingsAccount savingsAccount;
    private CheckingAccount checkingAccount;

    // Constructor and getters/setters omitted for brevity

    public void depositToSavings(double amount) {
        savingsAccount.deposit(amount);
    }

    public void withdrawFromSavings(double amount) {
        savingsAccount.withdraw(amount);
    }

    public void depositToCheckings(double amount) {
        checkingAccount.deposit(amount);
    }


    public void withdrawFromCheckings(double amount) {
        checkingAccount.withdraw(amount);
    }
}
```

Lab 07

```
interface Employee {
    String getName();
    double getBaseSalary();
    double calculateBonus();
    double getTotalSalary();
```

```
}

class Manager implements Employee {
    private String name;
    private double baseSalary;

    public Manager(String name, double baseSalary) {
        this.name = name;
        this.baseSalary = baseSalary;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public double getBaseSalary() {
        return baseSalary;
    }

    @Override
    public double calculateBonus() {
        return baseSalary * 0.2;
    }

    @Override
    public double getTotalSalary() {
        return baseSalary + calculateBonus();
    }
}

class Developer implements Employee {
    private String name;
    private double baseSalary;
    private int numberOfProjects;

    public Developer(String name, double baseSalary, int numberOfProjects) {
        this.name = name;
        this.baseSalary = baseSalary;
        this.numberOfProjects = numberOfProjects;
    }

    @Override
```

```
    public String getName() {
        return name;
    }

    @Override
    public double getBaseSalary() {
        return baseSalary;
    }

    @Override
    public double calculateBonus() {
        return baseSalary * 0.15 + numberOfProjects * (baseSalary * 0.15 * 0.05);
    }

    @Override
    public double getTotalSalary() {
        return baseSalary + calculateBonus();
    }
}

class Intern implements Employee {
    private String name;
    private double baseSalary;

    public Intern(String name, double baseSalary) {
        this.name = name;
        this.baseSalary = baseSalary;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public double getBaseSalary() {
        return baseSalary;
    }

    @Override
    public double calculateBonus() {
        return baseSalary * 0.05;
    }
```

```
    @Override
    public double getTotalSalary() {
        return baseSalary + calculateBonus();
    }
}

public class Main {
    public static void main(String[] args) {
        Employee manager = new Manager("Alice", 50000);
        Employee developer = new Developer("Bob", 60000, 5);
        Employee intern = new Intern("Charlie", 20000);

        System.out.println("Manager:");
        System.out.println("Name: " + manager.getName());
        System.out.println("Base Salary: " + manager.getBaseSalary());
        System.out.println("Bonus: " + manager.calculateBonus());
        System.out.println("Total Salary: " + manager.getTotalSalary());
        System.out.println();

        System.out.println("Developer:");
        System.out.println("Name: " + developer.getName());
        System.out.println("Base Salary: " + developer.getBaseSalary());
        System.out.println("Bonus: " + developer.calculateBonus());
        System.out.println("Total Salary: " + developer.getTotalSalary());
        System.out.println();

        System.out.println("Intern:");
        System.out.println("Name: " + intern.getName());
        System.out.println("Base Salary: " + intern.getBaseSalary());
        System.out.println("Bonus: " + intern.calculateBonus());
        System.out.println("Total Salary: " + intern.getTotalSalary());
    }
}
```

Lab 07

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
```

```
class Voter {
    String name;
    String birthCertificateNumber;
    int age;

    public Voter(String name, String birthCertificateNumber, int age) {
        this.name = name;
        this.birthCertificateNumber = birthCertificateNumber;
        this.age = age;
    }
}

public class EVotingSystem {
    public static void main(String[] args) {
        List<Voter> voterList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("1. Add to voter list");
            System.out.println("2. Check age and remove");
            System.out.println("3. Show all voters");
            System.out.println("4. Calculate monthly payment for elders");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    try {
                        System.out.print("Enter name: ");
                        String name = scanner.next();
                        System.out.print("Enter birth certificate number: ");
                        String birthCertificateNumber = scanner.next();
                        System.out.print("Enter age: ");
                        int age = scanner.nextInt();

                        if (age >= 18) {
                            voterList.add(new Voter(name, birthCertificateNumber, age));
                            System.out.println("Voter added successfully.");
                        } else {
                            throw new IllegalArgumentException("Age must be 18 or above.");
                        }
                    } catch (IllegalArgumentException e) {
```

```
                System.out.println("Error: " + e.getMessage());
            }
            break;

        case 2:
            try {
                System.out.print("Enter birth certificate number to remove: ");
                String birthCertificateNumber = scanner.next();

                Voter voterToRemove = null;
                for (Voter voter : voterList) {
                    if (voter.birthCertificateNumber.equals(birthCertificateNumber)) {
                        voterToRemove = voter;
                        break;
                    }
                }

                if (voterToRemove != null && voterToRemove.age < 60) {
                    voterList.remove(voterToRemove);
                    System.out.println("Voter removed successfully.");
                } else {
                    throw new IllegalArgumentException("Voter not found or age is 60 or above.");
                }
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            }
            break;

        case 3:
            for (Voter voter : voterList) {
                System.out.println("Name: " + voter.name);
                System.out.println("Birth Certificate Number: " + voter.birthCertificateNumber);
                System.out.println("Age: " + voter.age);
                System.out.println();
            }
            break;

        case 4:
            for (Voter voter : voterList) {
                if (voter.age >= 60) {
                    int monthlyPayment = voter.age * 200;
                    System.out.println(voter.name + " (Age " + voter.age + ") gets monthly
payment: " + monthlyPayment);
                }
```

```
                }
                break;

            case 5:
                System.out.println("Exiting...");
                System.exit(0);
             default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
  }
}
```