* **F to Celsious**

class Calculation {

public double convert(double F) {

return (F - 32) / 1.8;

}

}

public class FtoC {

public static void main(String[] args) { // এখানে "Strring" -> "String" হবে

Calculation cal = new Calculation();

System.out.println("Temperature in celsius: " + cal.convert(45));

}

}

**Prime**   
class PrimeNumbers {

// Method to check prime

boolean isPrime(int n) {

if (n <= 1) return false; // 1 or less is not prime

for (int i = 2; i <= n / 2; i++) {

if (n % i == 0) return false; // divisible means not prime

}

return true;

}

public static void main(String[] args) {

PrimeNumbers obj = new PrimeNumbers(); // object create

int start = 10; // starting number

int end = 50; // ending number

System.out.println("Prime numbers between " + start + " and " + end + " are:");

for (int i = start; i <= end; i++) {

if (obj.isPrime(i)) {

System.out.print(i + " ");

}

}

}

}

Or,

import java.util.Scanner;

public class PrimeInterval {

// Method to check if a number is prime

public static boolean isPrime(int n) {

if (n <= 1)

return false;

if (n == 2)

return true;

if (n % 2 == 0)

return false;

for (int i = 3; i \* i <= n; i += 2) {

if (n % i == 0)

return false;

}

return true;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the starting interval: ");

int start = sc.nextInt();

System.out.print("Enter the ending interval: ");

int end = sc.nextInt();

System.out.println("Prime numbers between " + start + " and " + end + " are:");

for (int i = start; i <= end; i++) {

if (isPrime(i)) {

System.out.print(i + " ");

}

}

System.out.println();

sc.close();

}

}

\*\*\***CAr**

class Car {

private String brand;

private String model;

private int year;

private double price;

// Getter and Setter methods

public String getBrand() { return brand; }

public void setBrand(String brand) { this.brand = brand; }

public String getModel() { return model; }

public void setModel(String model) { this.model = model; }

public int getYear() { return year; }

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

public double getPrice() { return price; }

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

}

public class CarTest {

public static void main(String[] args) {

Car myCar = new Car();

// Set values

myCar.setBrand("Toyota");

myCar.setModel("Corolla");

myCar.setYear(2022);

myCar.setPrice(20000.50);

// Display values

System.out.println("Car Brand: " + myCar.getBrand());

System.out.println("Car Model: " + myCar.getModel());

System.out.println("Car Year: " + myCar.getYear());

System.out.println("Car Price: $" + myCar.getPrice());

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**3 abstract vehicle \*\*\***

**abstract class Vehicle {**

**public abstract void startEngine();**

**public void stopEngine() {**

**System.out.println("Engine stopped.");**

**}**

**}**

**class Car extends Vehicle {**

**public void startEngine() {**

**System.out.println("Car engine started.");**

**}**

**}**

**class Motorcycle extends Vehicle {**

**public void startEngine() {**

**System.out.println("Motorcycle engine started.");**

**}**

**}**

**public class VehicleTest {**

**public static void main(String[] args) {**

**Vehicle myCar = new Car();**

**Vehicle myBike = new Motorcycle();**

**myCar.startEngine();**

**myCar.stopEngine();**

**myBike.startEngine();**

**myBike.stopEngine();**

**}**

**}**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\* jog biog gun vag**

**import java.util.Scanner;**

**class Calculator {**

**// Methods for each operation**

**double add(double a, double b) {**

**return a + b;**

**}**

**double subtract(double a, double b) {**

**return a - b;**

**}**

**double multiply(double a, double b) {**

**return a \* b;**

**}**

**double divide(double a, double b) {**

**if (b == 0) {**

**System.out.println("Division by zero not allowed!");**

**return 0;**

**}**

**return a / b;**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**Scanner sc = new Scanner(System.in);**

**Calculator calc = new Calculator();**

**// Take inputs**

**System.out.print("Enter first number: ");**

**double num1 = sc.nextDouble();**

**System.out.print("Enter second number: ");**

**double num2 = sc.nextDouble();**

**// Menu**

**System.out.println("Choose Operation:");**

**System.out.println("1 - Addition");**

**System.out.println("2 - Subtraction");**

**System.out.println("3 - Multiplication");**

**System.out.println("4 - Division");**

**int choice = sc.nextInt();**

**// Perform operation**

**double result = 0;**

**switch (choice) {**

**case 1: result = calc.add(num1, num2); break;**

**case 2: result = calc.subtract(num1, num2); break;**

**case 3: result = calc.multiply(num1, num2); break;**

**case 4: result = calc.divide(num1, num2); break;**

**default: System.out.println("Invalid choice!");**

**}**

**// Show result**

**if (choice >= 1 && choice <= 4) {**

**System.out.println("Result: " + result);**

**}**

**sc.close();**

**}**

**}**

**Bank account**

class BankAccount{

    String name;

    int accNo;

    String accountType;

    double balance;

    public BankAccount(String name, int accNo, String accountType, double balance) {

        this.name = name;

        this.accNo = accNo;

        this.accountType = accountType;

        this.balance = balance;

    }

    public void deposit(double amount){

        if(amount > 0){

            balance+=amount;

            System.out.println("Succesfully deposite " + amount);

        }else{

            System.out.println("Invalid");

        }

    }

    public void withdraw(double amount){

        if(amount>0)

        if(balance >= amount){

            balance -=amount;

            System.out.println("Successfully withdrew " + amount);

        }

        else{

            System.out.println("Invalid");

        }

    }

    public void displayAccountInfo() {

        System.out.println("Account Holder Name: " + name);

        //System.out.println("Account Number: " + accNo);

        //System.out.println("Account Type: " + accountType);

        System.out.println("Account Balance: " + balance);

    }

}

public class Bacc

{

    public static void main( String[] args){

        BankAccount acc=new BankAccount("John Doe", 12345, "Savings", 1000.0);

        acc.displayAccountInfo();

        acc.deposit(500);

        acc.withdraw(200);

        acc.displayAccountInfo();

    }

}

**Using abstract**

**abstract class BankAccount {**

**String depositorName;**

**String accountNumber;**

**String accountType;**

**double balance;**

**public BankAccount(String depositorName, String accountNumber, String accountType, double balance) {**

**this.depositorName = depositorName;**

**this.accountNumber = accountNumber;**

**this.accountType = accountType;**

**this.balance = balance;**

**}**

**abstract void deposit(double amount);**

**abstract void withdraw(double amount);**

**abstract void display();**

**}**

**class MyBankAccount extends BankAccount {**

**public MyBankAccount(String depositorName, String accountNumber, String accountType, double balance) {**

**super(depositorName, accountNumber, accountType, balance);**

**}**

**@Override**

**void deposit(double amount) {**

**balance += amount;**

**System.out.println(amount + " deposited successfully.");**

**}**

**@Override**

**void withdraw(double amount) {**

**if (amount <= balance) {**

**balance -= amount;**

**System.out.println(amount + " withdrawn successfully.");**

**} else {**

**System.out.println("Insufficient balance.");**

**}**

**}**

**@Override**

**void display() {**

**System.out.println("Depositor Name: " + depositorName);**

**System.out.println("Account Number: " + accountNumber);**

**System.out.println("Account Type: " + accountType);**

**System.out.println("Balance: " + balance);**

**}**

**}**

**class BankAccountTest {**

**public static void main(String[] args) {**

**MyBankAccount acc1 = new MyBankAccount("Rahim", "12345", "Savings", 5000);**

**acc1.display();**

**acc1.deposit(2000);**

**acc1.withdraw(3000);**

**acc1.display();**

**}**

**}**

**\*\*\*evaporetot**

**class Door {**

**boolean close = true;**

**void close() {**

**close = true;**

**System.out.println("Door closed.");**

**}**

**void open() {**

**close = false;**

**System.out.println("Door opened.");**

**}**

**}**

**class ElevatorController {**

**int floorId;**

**int position;**

**boolean direction;**

**Door door;**

**public ElevatorController() {**

**door = new Door();**

**}**

**}**

**class Elevator {**

**boolean direction;**

**int currentFloor;**

**ElevatorController controller;**

**public Elevator() {**

**controller = new ElevatorController();**

**}**

**void move() {**

**System.out.println("Elevator moving.");**

**}**

**void stop() {**

**System.out.println("Elevator stopped.");**

**}**

**void status() {**

**System.out.println("Current Floor: " + currentFloor + ", Direction: " + direction);**

**}**

**}**

**class Button {**

**boolean illuminate = false;**

**void illuminate() {**

**illuminate = true;**

**System.out.println("Button illuminated.");**

**}**

**void cancelIlluminate() {**

**illuminate = false;**

**System.out.println("Button light off.");**

**}**

**void status() {**

**System.out.println("Button light status: " + (illuminate ? "On" : "Off"));**

**}**

**}**

**class ElevatorButton extends Button {**

**int floorNum;**

**}**

**class FloorButton extends Button {**

**int floorNum;**

**boolean direction;**

**}**

**class ElevatorTest {**

**public static void main(String[] args) {**

**Elevator elevator = new Elevator();**

**elevator.move();**

**elevator.status();**

**elevator.stop();**

**elevator.controller.door.open();**

**elevator.controller.door.close();**

**ElevatorButton eb = new ElevatorButton();**

**eb.floorNum = 5;**

**eb.illuminate();**

**eb.cancelIlluminate();**

**}**

**}**

**import java.io.BufferedReader;**

**import java.io.FileReader;**

**import java.io.IOException;**

**class Convert {**

**public double add(double F) {**

**return (F - 32) / 1.8;**

**}**

**}**

**public class FtoC {**

**public static void main(String[] args) {**

**Convert cc = new Convert();**

**try {**

**BufferedReader br = new BufferedReader(new FileReader("input.txt"));**

**String line = br.readLine(); // ফাইল থেকে এক লাইন পড়া**

**if (line != null) {**

**double fahrenheit = Double.parseDouble(line.trim());**

**double celsius = cc.add(fahrenheit);**

**System.out.println("Converted value in Celsius: " + celsius);**

**}**

**br.close();**

**} catch (IOException e) {**

**System.out.println("Error reading file: " + e.getMessage());**

**} catch (NumberFormatException e) {**

**System.out.println("Invalid number format in file.");**

**}**

**}**

**}**

**Evaporator\*\*\*\*\*\***

**class Elevator {**

**private boolean direction;**

**private int current\_floor;**

**public Elevator(boolean direction, int current\_floor) {**

**this.direction = direction;**

**this.current\_floor = current\_floor;**

**}**

**public void move() {**

**System.out.println("Elevator is moving");**

**}**

**public void stop() {**

**System.out.println("Elevator has stopped");**

**}**

**public void status() {**

**System.out.println("Direction: " + direction);**

**System.out.println("Current floor: " + current\_floor);**

**}**

**// getters and setters**

**public boolean isDirection() {**

**return direction;**

**}**

**public void setDirection(boolean direction) {**

**this.direction = direction;**

**}**

**public int getCurrent\_floor() {**

**return current\_floor;**

**}**

**public void setCurrent\_floor(int current\_floor) {**

**this.current\_floor = current\_floor;**

**}**

**}**

**class Door {**

**private boolean close = true;**

**public void close() {**

**close = true;**

**System.out.println("Door closed");**

**}**

**public void open() {**

**close = false;**

**System.out.println("Door opened");**

**}**

**public boolean isClosed() {**

**return close;**

**}**

**}**

**class ElevatorController {**

**private int floor\_id;**

**private int position;**

**private boolean direction;**

**public ElevatorController(int floor\_id, int position, boolean direction) {**

**this.floor\_id = floor\_id;**

**this.position = position;**

**this.direction = direction;**

**}**

**// getters and setters**

**public int getFloor\_id() {**

**return floor\_id;**

**}**

**public void setFloor\_id(int floor\_id) {**

**this.floor\_id = floor\_id;**

**}**

**public int getPosition() {**

**return position;**

**}**

**public void setPosition(int position) {**

**this.position = position;**

**}**

**public boolean isDirection() {**

**return direction;**

**}**

**public void setDirection(boolean direction) {**

**this.direction = direction;**

**}**

**}**

**class Button {**

**private boolean illuminate = false;**

**public void illuminate() {**

**illuminate = true;**

**System.out.println("Button illuminated");**

**}**

**public void cancelIlluminate() {**

**illuminate = false;**

**System.out.println("Button illumination cancelled");**

**}**

**public void status() {**

**System.out.println("Button is " + (illuminate ? "illuminated" : "not illuminated"));**

**}**

**public boolean isIlluminated() {**

**return illuminate;**

**}**

**}**

**class Elevator\_Button extends Button {**

**private int floor\_num;**

**public Elevator\_Button(int floor\_num) {**

**this.floor\_num = floor\_num;**

**}**

**public int getFloor\_num() {**

**return floor\_num;**

**}**

**public void setFloor\_num(int floor\_num) {**

**this.floor\_num = floor\_num;**

**}**

**}**

**class Floor\_Button extends Button {**

**private int floor\_num;**

**private boolean direction;**

**public Floor\_Button(int floor\_num, boolean direction) {**

**this.floor\_num = floor\_num;**

**this.direction = direction;**

**}**

**public int getFloor\_num() {**

**return floor\_num;**

**}**

**public void setFloor\_num(int floor\_num) {**

**this.floor\_num = floor\_num;**

**}**

**public boolean isDirection() {**

**return direction;**

**}**

**public void setDirection(boolean direction) {**

**this.direction = direction;**

**}**

**}**

**BANK ACCOUNT@2\*\*\***

**// BankAccounts.java**

**// Simple OOP design for Account, SavingsAccount and CurrentAccount**

**abstract class Account {**

**protected String customerName;**

**protected String accountNumber;**

**protected String accountType;**

**protected double balance;**

**// Constructor to initialize all instance variables**

**public Account(String customerName, String accountNumber, String accountType, double balance) {**

**this.customerName = customerName;**

**this.accountNumber = accountNumber;**

**this.accountType = accountType;**

**this.balance = balance;**

**}**

**// Deposit from a customer and update balance**

**public void deposit(double amount) {**

**if (amount <= 0) {**

**System.out.println("Deposit amount must be positive.");**

**return;**

**}**

**balance += amount;**

**System.out.printf("Deposited %.2f. New balance: %.2f\n", amount, balance);**

**}**

**// Display the balance**

**public void showBalance() {**

**System.out.printf("Account: %s (%s) | Balance: %.2f\n", customerName, accountNumber, balance);**

**}**

**// Compute and deposit interest (implementation differs by account type)**

**public abstract void computeInterest();**

**// Permit withdrawal and update the balance (some accounts may override)**

**public boolean withdraw(double amount) {**

**if (amount <= 0) {**

**System.out.println("Withdrawal amount must be positive.");**

**return false;**

**}**

**if (amount > balance) {**

**System.out.println("Insufficient funds. Withdrawal failed.");**

**return false;**

**}**

**balance -= amount;**

**System.out.printf("Withdrawn %.2f. New balance: %.2f\n", amount, balance);**

**return true;**

**}**

**}**

**/\* SavingsAccount:**

**- Provides compound/simple interest (here we use simple interest per call at a fixed rate)**

**- Allows withdrawals normally**

**\*/**

**class SavingsAccount extends Account {**

**private double annualInterestRate; // e.g., 0.05 for 5%**

**public SavingsAccount(String customerName, String accountNumber, double openingBalance, double annualInterestRate) {**

**super(customerName, accountNumber, "Savings", openingBalance);**

**this.annualInterestRate = annualInterestRate;**

**}**

**// Compute and deposit interest. For simplicity: compute interest on current balance as one periodic**

**// call (e.g., annual). This method adds interest to balance.**

**@Override**

**public void computeInterest() {**

**double interest = balance \* annualInterestRate;**

**balance += interest;**

**System.out.printf("Interest computed at %.2f%% -> Interest: %.2f added. New balance: %.2f\n",**

**annualInterestRate \* 100, interest, balance);**

**}**

**}**

**/\* CurrentAccount:**

**- No interest**

**- Must maintain a minimum balance (assume BDT 1000). If balance falls below minimum after any operation,**

**impose a penalty (we choose BDT 100 as example). If penalty would make balance negative, restrict withdrawal.**

**\*/**

**class CurrentAccount extends Account {**

**private double minimumBalance;**

**private double penaltyAmount;**

**public CurrentAccount(String customerName, String accountNumber, double openingBalance) {**

**super(customerName, accountNumber, "Current", openingBalance);**

**this.minimumBalance = 1000.0; // as per problem statement**

**this.penaltyAmount = 100.0; // chosen penalty (exam-style; can be changed)**

**}**

**// No interest for current account**

**@Override**

**public void computeInterest() {**

**System.out.println("Current account does not provide interest.");**

**}**

**// Override withdraw to check minimum balance and impose penalty if necessary**

**@Override**

**public boolean withdraw(double amount) {**

**if (amount <= 0) {**

**System.out.println("Withdrawal amount must be positive.");**

**return false;**

**}**

**if (amount > balance) {**

**System.out.println("Insufficient funds. Withdrawal failed.");**

**return false;**

**}**

**// Tentatively perform withdrawal**

**double newBalance = balance - amount;**

**// If after withdrawal the balance would be below minimum, impose penalty**

**if (newBalance < minimumBalance) {**

**System.out.printf("Warning: Balance after withdrawal (%.2f) will be below minimum (%.2f).\n",**

**newBalance, minimumBalance);**

**// Check if penalty can be applied**

**if (newBalance - penaltyAmount < 0) {**

**// Reject withdrawal because penalty would make balance negative**

**System.out.println("Withdrawal restricted: penalty would make the account overdrawn. Withdrawal denied.");**

**return false;**

**} else {**

**// Apply withdrawal and penalty**

**balance = newBalance - penaltyAmount;**

**System.out.printf("Withdrawn %.2f. Penalty %.2f imposed for falling below minimum balance.\n",**

**amount, penaltyAmount);**

**System.out.printf("New balance after penalty: %.2f\n", balance);**

**return true;**

**}**

**} else {**

**// No penalty needed**

**balance = newBalance;**

**System.out.printf("Withdrawn %.2f. New balance: %.2f\n", amount, balance);**

**return true;**

**}**

**}**

**// Method to check minimum balance and impose penalty if current balance is below minimum (standalone)**

**public void checkMinimumBalanceAndPenalize() {**

**if (balance < minimumBalance) {**

**if (balance - penaltyAmount < 0) {**

**System.out.println("Balance is below minimum and penalty cannot be applied without overdraft. Please deposit.");**

**} else {**

**balance -= penaltyAmount;**

**System.out.printf("Minimum balance violation: penalty %.2f imposed. New balance: %.2f\n", penaltyAmount, balance);**

**}**

**} else {**

**System.out.println("Minimum balance satisfied. No penalty.");**

**}**

**}**

**}**

**// Demonstration main class**

**public class BankDemo {**

**public static void main(String[] args) {**

**System.out.println("=== Savings Account Demo ===");**

**SavingsAccount sAcc = new SavingsAccount("Rafi", "SAV001", 5000.0, 0.05); // 5% interest**

**sAcc.showBalance();**

**sAcc.deposit(2000.0);**

**sAcc.computeInterest(); // adds 5% of current balance**

**sAcc.withdraw(1000.0);**

**sAcc.showBalance();**

**System.out.println("\n=== Current Account Demo ===");**

**CurrentAccount cAcc = new CurrentAccount("Mita", "CUR001", 1500.0);**

**cAcc.showBalance();**

**cAcc.withdraw(600.0); // will cause balance drop to 900 -> below 1000 -> penalty applies**

**cAcc.showBalance();**

**cAcc.deposit(300.0);**

**cAcc.checkMinimumBalanceAndPenalize(); // standalone check (if still below)**

**cAcc.withdraw(50.0); // normal small withdrawal**

**cAcc.showBalance();**

**System.out.println("\n=== Edge Case: Withdrawal Denied Due to Penalty Overdraft ===");**

**CurrentAccount c2 = new CurrentAccount("Salma", "CUR002", 1050.0);**

**c2.showBalance();**

**// Try withdrawing 60: newBalance = 990, penalty 100 -> would cause balance -10 => deny**

**c2.withdraw(60.0); // should be denied by our rule**

**c2.showBalance();}**

**}**

**SUM\*\*\*\*\*\*\*\*\*\***

**import java.util.Scanner;**

**public class SumN {**

**public static void main(String[] args) {**

**Scanner sc = new Scanner(System.in);**

**System.out.print("n er man din: ");**

**int n = sc.nextInt();**

**int sum = 0;**

**for (int i = 1; i <= n; i++) {**

**sum += i;**

**}**

**System.out.println("1 theke " + n + " porjonto jogfol: " + sum);**

**sc.close();**

**}**

**}**

**Or\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**public class SumOneToHundred {**

**public static void main(String[] args) {**

**int sum = 0;**

**for (int i = 1; i <= 100; i++) {**

**sum += i;**

**}**

**System.out.println("Sum of numbers from 1 to 100 is: " + sum);**

**}**

**}**