// Main.java

package assignment\_3;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);

        /\*\*\*\*\* TASK 01 \*\*\*\*\*/

        System.out.println("----- Task 01 -----");

        System.out.print("Enter 1st integer: ");

        var num1 = keyboard.nextInt();

        System.out.print("Enter 2nd integer: ");

        var num2 = keyboard.nextInt();

        Task01 runs1 = new Task01(num1, num2, "add");

        Task01 runs2 = new Task01(num1, num2, "sub");

        Task01 runs3 = new Task01(num1, num2, "mul");

        Task01 runs4 = new Task01(num1, num2, "div");

        Task01 runs5 = new Task01(num1, num2, "oth");

        Thread add = new Thread(runs1, "Addition Thread");

        Thread sub = new Thread(runs2, "Subtraction Thread");

        Thread mul = new Thread(runs3, "Multiplication Thread");

        Thread div = new Thread(runs4, "Division Thread");

        Thread oth = new Thread(runs5, "Other Threads");

        add.start();

        sub.start();

        mul.start();

        div.start();

        oth.start();

        try {

            add.join();

            sub.join();

            mul.join();

            div.join();

            oth.join();

        } catch (InterruptedException e) {

            // e.printStackTrace();

        }

        System.out.println("----- Task 01 Completed -----\n");

        /\*\*\*\*\* TASK 02 \*\*\*\*\*/

        System.out.println("----- Task 02 -----");

        Thread houseStark = new Thread(new Task02("House Stark"));

        Thread houseTargaryen = new Thread(new Task02("House Targaryen"));

        Thread houseBolton = new Thread(new Task02("House Bolton"));

        Thread houseLannister = new Thread(new Task02("House Lannister"));

        Thread houseTyrell = new Thread(new Task02("House Tyrell"));

        houseStark.setPriority(Thread.MAX\_PRIORITY);

        houseBolton.setPriority(Thread.MIN\_PRIORITY);

        System.out.println("~ Single Threaded Mode ~");

        houseStark.run();

        houseTargaryen.run();

        houseLannister.run();

        houseBolton.run();

        System.out.println("~ Multi Threaded Mode ~");

        houseStark.start();

        houseLannister.start();

        houseBolton.start();

        houseTyrell.start();

        try {

            houseStark.join();

            houseTargaryen.join();

            houseLannister.join();

            houseBolton.join();

            houseTyrell.join();

        } catch (InterruptedException e) {

            // e.printStackTrace();

        }

        System.out.println("----- Task 02 Completed -----\n");

        /\*\*\*\*\* TASK 03 \*\*\*\*\*/

        System.out.println("----- Task 03 -----");

        long[] fibonacci = new long[51];

        fibonacci[0] = 1;

        fibonacci[1] = 1;

        for (int i = 2; i < fibonacci.length; i++) {

            fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];

        }

        Task03[] runs = new Task03[4];

        runs[0] = new Task03(fibonacci, "odd", 1);

        runs[1] = new Task03(fibonacci, "even", 1);

        runs[2] = new Task03(fibonacci, "odd", 2);

        runs[3] = new Task03(fibonacci, "even", 2);

        Thread[] th = new Thread[runs.length];

        for (int i = 0; i < th.length; i++) {

            th[i] = new Thread(runs[i], "Thread " + i);

        }

        th[0].start();

        th[1].start();

        th[2].start();

        th[3].start();

        try {

            th[0].join();

            th[1].join();

            th[2].join();

            th[3].join();

        } catch (InterruptedException e) {

            // e.printStackTrace();

        }

        long[] mean = new long[runs.length];

        for (int i = 0; i < mean.length; i++) {

            mean[i] = runs[i].getMean();

        }

        Task03 run = new Task03(mean);

        Thread th5 = new Thread(run, "Thread 4");

        th5.start();

        try {

            th5.join();

        } catch (InterruptedException e) {

            // e.printStackTrace();

        }

        System.out.println("PIN: " + run.getMean());

        System.out.println("----- Task 03 Completed -----\n");

        keyboard.close();

    }

}

// Task01.java

package assignment\_3;

public class Task01 implements Runnable {

    private int a, b;

    private String mode;

    public Task01(int a, int b, String mode) {

        this.a = a;

        this.b = b;

        this.mode = mode;

    }

    @Override

    public void run() {

        String output = null;

        switch (mode) {

        case "add":

            output = String.format("Sum: %s", a + b);

            break;

        case "sub":

            output = String.format("Sub: %s", a - b);

            break;

        case "mul":

            output = String.format("Mul: %s", a \* b);

            break;

        case "div":

            output = String.format("Div: %.5f", 1.0 \* a / b);

            break;

        case "oth":

            output = "No valid operation";

            break;

        }

        String threadName = Thread.currentThread().getName();

        System.out.printf("%s\n> %s\n", threadName, output);

    }

}

// Task02.java

package assignment\_3;

public class Task02 implements Runnable {

    private String houseName;

    public Task02(String name) {

        this.houseName = name;

    }

    @Override

    public void run() {

        System.out.printf("The house is : %s\n", houseName);

        if (houseName == "House Stark" || houseName == "House Targaryen") {

            sleep(1000);

        } else if (houseName == "House Lannister" || houseName == "House Bolton") {

            sleep(3000);

        } else {

            sleep(5000);

        }

    }

    private void sleep(int millis) {

        try {

            Thread.sleep(millis);

        } catch (Exception e) {

            // e.printStackTrace();

        }

    }

}

// Task03.java

package assignment\_3;

public class Task03 implements Runnable {

    private long[] fibonacci;

    private String type;

    private int part;

    private long average;

    private long[] mean;

    private boolean skipAverage = false;

    public Task03(long[] fibonacci, String type, int part) {

        this.fibonacci = fibonacci;

        this.type = type;

        this.part = part;

    }

    public Task03(long[] mean) {

        this.mean = mean;

        this.skipAverage = true;

    }

    @Override

    public void run() {

        if (skipAverage) {

            calculateMean(mean, 0, mean.length, 1);

        } else {

            if (part == 1 && type == "odd") {

                calculateMean(fibonacci, 1, 1 + fibonacci.length / 2, 2);

            } else if (part == 1 && type == "even") {

                calculateMean(fibonacci, 2, 1 + fibonacci.length / 2, 2);

            } else if (part == 2 && type == "odd") {

                calculateMean(fibonacci, 26, fibonacci.length, 2);

            } else if (part == 2 && type == "even") {

                calculateMean(fibonacci, 27, fibonacci.length, 2);

            }

        }

    }

    private void calculateMean(long[] array, int ini, int len, int step) {

        long sum = 0;

        int count = 0;

        for (int i = ini; i < len; i += step, count++) {

            sum += array[i];

        }

        this.average = sum / count;

    }

    public long getMean() {

        return this.average;

    }

}