TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN – ĐHQG-HCM
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

BÁO CÁO THỰC HÀNH

Bài thực hành số 01: Bài tập thực hành Java cơ bản

Môn học: Lập trình ứng dụng mạng

Lớp: NT109.O21.MMCL

THÀNH VIÊN THỰC HIỆN (Nhóm 11):

STT	Họ và tên	MSSV
1	Lê Hoàng Khánh	21522205

ĐÁNH GIÁ KHÁC:

Tổng thời gian thực hiện	1 tuần
Ý kiến *(nếu có)* + Khó khăn + Đề xuất, kiến nghị	Không có

MỤC LỤC

A. BÁO CÁO CHI TIẾT

Các Project trong bài Lab sẽ sử dụng Java 17 và build system là Gradle DSL.

1. Xây dựng thư viện Write Logs.

a. Xây dựng hàm Log.

```java
no usages
public class Log {

    2 usages
    private final String logName;


    no usages
    public Log(String configPath) { logName = readConfig(configPath); }

    1 usage
    private String readConfig(String configPath) {
        try {
            return Files.readString(Paths.get(configPath));
        } catch (Exception e) {
            // Handle the exception here (e.g., log an error or throw a custom exception)
            System.err.println("Error reading config file: " + e.getMessage());
            return "";
        }
    }

    1 usage
    public String createLogFile() {
        String logFilePath = "logs" + File.separator + logName + ".txt";
        Path logFile = Paths.get(logFilePath);
        try {
            if (Files.notExists(logFile)) {
                Files.createDirectories(logFile.getParent());
                Files.createFile(logFile);
            }
        } catch (Exception e) {
            System.err.println("Error creating log file: " + e.getMessage());
        }
        return logFilePath;
    }

    no usages
    public void writeLog(String message) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
        String logContent = formatter.format(LocalDateTime.now()) + " - " + message;
        try {
            Files.writeString(Paths.get(createLogFile()), csq: logContent + "\n",  StandardCharsets.UTF_8, StandardOpenOption.APPEND);
            System.out.printf("%s\n%n", logContent);
        } catch (Exception e) {
            System.err.println("Error writing to log file: " + e.getMessage());
        }
    }
}
```

Constructor Log. Tham số configPath là đường dẫn tới file cấu hình chứa tên file log. Hàm sẽ đọc tên file log từ file cấu hình và lưu vào thuộc tính logName.

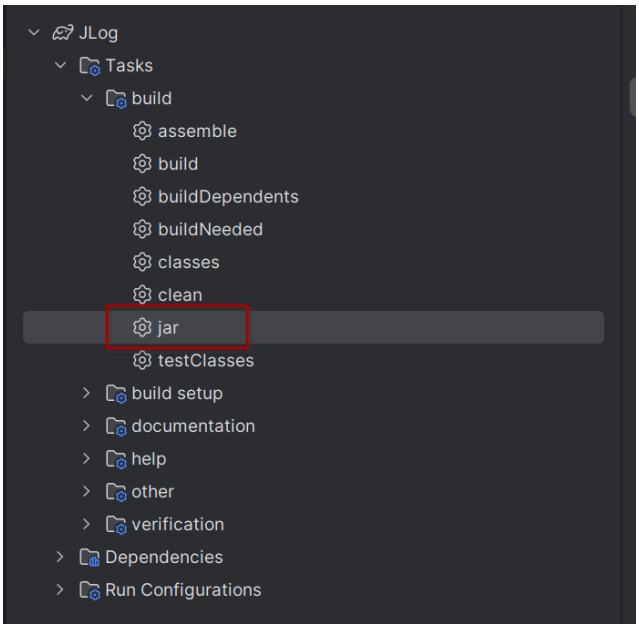
readConfig, phương thức này đọc nội dung từ file cấu hình theo đường dẫn configPath. Nó sẽ trả về tên file log đọc được, nếu gặp lỗi ngoại lệ trong quá trình đọc thì sẽ in ra thông báo lỗi và trả về một chuỗi trống.

createLogFile: Phương thức này tạo file log. Thứ nhất, nó sẽ tạo đường dẫn đến file log bằng cách nối thêm tên file log (đọc từ config) với thư mục "logs". Sau

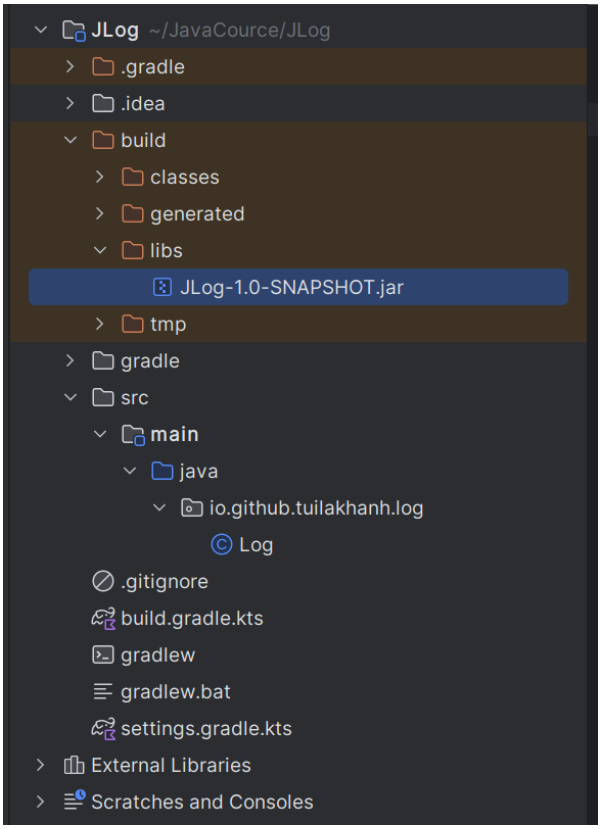
đó, kiểm tra nếu file log chưa tồn tại thì sẽ tạo các thư mục cha cần thiết và tạo file log. Cuối cùng, phương thức trả về đường dẫn đến file log vừa tạo.

writeLog: Phương thức này ghi nội dung message vào file log. Trước tiên, nó định dạng thời gian hiện tại theo định dạng dd-MM-yyyy HH:mm:ss. Sau đó, nội dung log được tạo bằng cách kết hợp thời gian đã định dạng với message. Phương thức sẽ ghi nội dung log này vào cuối file log được tạo bởi phương thức createLogFile(). Ngoài ra, nội dung log cũng được in ra console để tiện theo dõi.

b. Xuất thư viện Jar.



Chạy gradle task jar, sau khi chạy xong sẽ thu được thư viện Log nằm ở folder **build/libs**



2. Thêm thư viện Log vào Project Lab01

```
plugins { this: PluginDependenciesSpecScope
    id("java")
}

group = "io.github.tuilakhanh.Lab01"
version = "1.0-SNAPSHOT"

repositories { this: RepositoryHandler
    mavenCentral()
}

dependencies { this: DependencyHandlerScope
    testImplementation(platform("org.junit:junit-bom:5.9.1"))
    testImplementation("org.junit.jupiter:junit-jupiter")
    implementation(files( ...paths: "libs/JLog-1.0-SNAPSHOT.jar"))
}


tasks.test { this: Test!
    useJUnitPlatform()
}
```

Sao chép thư viên JLog được build từ trước vào folder libs/ thêm lệnh ở dòng 15 như hình vào file build.gradle.kts. Sau khi thực hiện xong thư viện đã được thêm vào Project Lab01.

3. Bài 01

```
public class Exercise01 {
    1 usage   tuilakhanh
    public static void Exercise01(Log log) {
        log.writeLog(String.format("Bai 01: %s", findNum()));
    }

    1 usage   tuilakhanh
    static String findNum() {
        return IntStream.rangeClosed(10, 200) IntStream
                .filter(i -> i % 7 == 0 && i % 5 != 0)
                .mapToObj(String::valueOf) Stream<String>
                .collect(Collectors.joining( delimiter: ", "));
    }
}
```

Exercise01(), Nhận một đối tượng Log để ghi log. Gọi phương thức findNum() để tìm các số thỏa mãn điều kiện. Ghi kết quả ra file log bằng phương thức writeLog() của đối tượng Log.

findNum(), Tạo một IntStream chứa các số nguyên từ 10 đến 200 (bao gồm cả 10 và 200). Lọc các số chia hết cho 7 nhưng không chia hết cho 5 bằng phương thức filter(). Chuyển các số thành dạng chuỗi ký tự bằng phương thức mapToObj(). Nối các chuỗi thành một chuỗi duy nhất, ngăn cách nhau bởi dấu phẩy, bằng phương thức collect() và Collectors.joining(", "). Trả về chuỗi kết quả đó.

4. Bài 02

```java
public class Exercise02 {
    1 usage   tuilakhanh
    public static void Exercise02(Scanner scanner, Log log) {
        log.writeLog("Bai 2: Nhap so nguyen n: ");
        var n = scanner.nextLong();
        log.writeLog(String.format("Bai 02: So nguyen n = %s, Factorial = %s", n, Fractorial(n)));
    }
    2 usages   tuilakhanh
    static long Fractorial(long n) { return n < 0 ? Long.MIN_VALUE : (n == 0) ? 1 : n * Fractorial( n: n - 1); }
}
```

Exercise02(),

- Nhận hai đối tượng: scanner để đọc dữ liệu nhập từ bàn phím và log để ghi vào file log.
- Yêu cầu người dùng nhập một số nguyên n bằng phương thức log.writeLog().
- Đọc số nguyên nhập từ bàn phím bằng phương thức scanner.nextLong() và lưu vào biến n.
- Gọi phương thức Fractorial(n) để tính giai thừa của số n.
- Ghi kết quả ra file log cùng với số nhập ban đầu theo định dạng "Bai 02: So nguyen n = %s, Factorial = %s".

Fractorial(n), Tính giai thừa của số nguyên n. Xử lý các trường hợp:

- Nếu n nhỏ hơn 0, trả về Long.MIN_VALUE để báo hiệu lỗi.
- Nếu n bằng 0, trả về 1 vì 0! = 1.
- Nếu n lớn hơn 0, tính giai thừa đệ quy theo công thức: n! = n * (n-1)!.

5. Bài 03

```
public class Exercise03 {
    1 usage   tuilakhanh
    public static void Exercise03(Scanner scanner, Log log) {
        log.writeLog("Bai 03: Nhap so nguyen n: ");
        var n = scanner.nextInt();
        log.writeLog(String.format("Bai 03: So nguyen n = %s, map(i, i*i) = %s", n, MapIxI(n)));
    }

    1 usage   tuilakhanh
    static String MapIxI(int n) {
        var map = new HashMap<Integer, Integer>();
        for (int i = 1; i <= n; i++) {
            map.put(i, i * i);
        }
        return map.toString();
    }
}
```

Exercise03():

- Nhận hai đối tượng: scanner để đọc dữ liệu nhập từ bàn phím và log để ghi vào file log.
- Yêu cầu người dùng nhập một số nguyên n bằng phương thức log.writeLog().
- Đọc số nguyên nhập từ bàn phím bằng phương thức scanner.nextInt() và lưu vào biến n.
- Gọi phương thức MapIxI(n) để tạo HashMap và tính các giá trị i*i.
- Ghi kết quả HashMap ra file log cùng với số nhập ban đầu theo định dạng "Bai 03: So nguyen n = %s, map(i, i*i) = %s".

MapIxI(n):

- Tạo một đối tượng HashMap để lưu trữ các cặp khóa-giá trị.
- Duyệt từ 1 đến n bằng vòng lặp for.
- Trong mỗi vòng lặp, thêm cặp giá trị (i, i*i) vào HashMap sử dụng phương thức put().
- Trả về chuỗi thể hiện nội dung của HashMap bằng cách gọi phương thức toString().

6. Bài 04

```java
public class Exercise04 {
    1 usage  tuilakhanh
    public static void Exercise04(Scanner scanner, Log log) {
        log.writeLog("Bai 04: Nhap so thap phan: ");
        var num = scanner.nextInt();
        int base;
        do {
            System.out.print("Bai 04: Nhap he co so B (2 <= B <= 32): ");
            base = scanner.nextInt();
        } while (base < 2 || base > 32);
        log.writeLog(String.format("Bai 04: So thap phan = %s, Co so = %s, KQ = %s", num, base, toBase(num, base)));
    }

    1 usage  tuilakhanh
    static String toBase(int num, int base) {
        if (base < 2 || base > 32) {
            throw new IllegalArgumentException("Base khong hop le: " + base);
        }

        StringBuilder sb = new StringBuilder();
        while (num > 0) {
            int digit = num % base;
            char digitChar = digit < 10 ? (char) ('0' + digit) : (char) ('A' + digit - 10);
            sb.append(digitChar);
            num /= base;
        }
        return sb.reverse().toString();
    }
}
```

Exercise04():

Yêu cầu người dùng nhập một số thập phân bằng log.writeLog() và scanner.nextInt(). Lưu trữ số này vào biến num.

Lặp lại việc yêu cầu người dùng nhập hệ cơ sở base cho đến khi base nằm trong phạm vi hợp lệ (2 đến 32) bằng vòng lặp do-while.

Gọi phương thức toBase(num, base) để thực hiện chuyển đổi số thập phân sang hệ cơ sở base.

Ghi kết quả ra file log theo định dạng "Bai 04: So thap phan = %s, Co so = %s, KQ = %s".

toBase():

- Kiểm tra tính hợp lệ của hệ cơ sở base. Nếu base không nằm trong khoảng từ 2 đến 32, nó ném ngoại lệ IllegalArgumentException với thông báo lỗi.
- Khởi tạo một đối tượng StringBuilder tên sb để xây dựng chuỗi kết quả theo từng chữ số.
- Lặp lại việc tính toán các chữ số cho đến khi num bằng 0:

  Tính toán chữ số cuối cùng (digit) bằng num % base.

- Xác định ký tự tương ứng với chữ số:
- Nếu digit nhỏ hơn 10, ký tự là '0' cộng với digit.
- Nếu digit lớn hơn hoặc bằng 10 (ký tự A-F), ký tự là 'A' cộng với digit trừ 10.
- Thêm ký tự digitChar vào đầu chuỗi sb bằng sb.append().
- Cập nhật num bằng num chia cho base để chuẩn bị tính toán chữ số tiếp theo.
- Đảo ngược chuỗi sb bằng sb.reverse() để có thứ tự các chữ số chính xác và chuyển đổi thành chuỗi String.
- Trả về chuỗi biểu diễn số thập phân num trong hệ cơ sở base.

7. Bài 05

```java
public class Exercise05 {
    3 usages
    static HashMap<Long, Long> memo = new HashMap<>();

    1 usage   tuilakhanh
    public static void Exercise05(Scanner scanner, Log log) {
        log.writeLog("Bai 05: Nhap so nguyen n: ");
        var n = scanner.nextInt();
        log.writeLog(String.format("Bai 05: so nguyen n = %s, N so Fibonacci dau tien: %s", n, findFirstNFibonacci(n)));
    }

    3 usages   tuilakhanh
    static long fibonacci(long n) {
        if (memo.containsKey(n)) {
            return memo.get(n);
        }
        if (n <= 1) {
            return n;
        } else {
            var result = fibonacci( n: n - 1) + fibonacci( n: n - 2);
            memo.put(n, result);
            return result;
        }
    }

    1 usage   tuilakhanh
    static String findFirstNFibonacci(int n) {
        return LongStream.rangeClosed(0, n) LongStream
                .map(Exercise05::fibonacci)
                .mapToObj(String::valueOf) Stream<String>
                .collect(Collectors.joining( delimiter: ", "));

    }
}
```

Biến toàn cục memo: Khởi tạo một HashMap tĩnh tên memo để lưu trữ các giá trị Fibonacci đã được tính toán trước đó theo cặp (chỉ số n, giá trị Fibonacci).

Exercise05():

- Yêu cầu người dùng nhập số nguyên n bằng log.writeLog() và scanner.nextInt().
- Gọi phương thức findFirstNFibonacci(n) để tính toán n số Fibonacci đầu tiên.
- Ghi kết quả ra file log theo định dạng "Bai 05: so nguyen n = %s, N so Fibonacci dau tien: %s".

findFirstNFibonacci():

- Sử dụng LongStream.rangeClosed(0, n) để tạo một luồng các số nguyên từ 0 đến n (bao gồm cả 0 và n).
- Ánh xạ từng số trong luồng này sang giá trị Fibonacci của nó bằng cách gọi map(Exercise05::fibonacci).
- Chuyển đổi các giá trị Fibonacci thành chuỗi bằng mapToObj(String::valueOf).
- Nối các chuỗi thành một chuỗi duy nhất, ngăn cách nhau bởi dấu phẩy, bằng collect(Collectors.joining(", ")).
- Trả về chuỗi kết quả chứa n số Fibonacci đầu tiên.

fibonacci(long n) (với memoization):

- Kiểm tra xem memo có chứa khóa n (chỉ số Fibonacci) hay không.
- Nếu có, giá trị Fibonacci tương ứng đã được tính toán trước đó và được trả về bằng memo.get(n).
- Nếu n nhỏ hơn hoặc bằng 1, đây là các trường hợp cơ sở của dãy Fibonacci và trả về chính n.
- Tính toán giá trị Fibonacci của n bằng cách cộng giá trị Fibonacci của n-1 và n-2 using recursive calls.
- Lưu trữ kết quả tính toán result vào memo với khóa n.
- Trả về giá trị Fibonacci đã tính toán result.

8. Bài 06.

```java
public class Exercise06 {
    1 usage   tuilakhanh
    public static void Exercise06(Scanner scanner, Log log) {
        log.writeLog("Bai 06: Nhap a: ");
        var a = scanner.nextInt();
        log.writeLog("Bai 06: Nhap b: ");
        var b = scanner.nextInt();
        var gcd = Exercise06.findGCD(a, b);
        var lcm = Exercise06.findLCM(a, b, gcd);
        log.writeLog(String.format("Bai 06: A = %s, B = %s, GCD = %s, LCM = %s", a, b, gcd, lcm));
    }

    1 usage   tuilakhanh
    static int findGCD(int a, int b) {
        while (b != 0) {
            var temp = a % b;
            a = b;
            b = temp;
        }
        return a;
    }

    1 usage   tuilakhanh
    static int findLCM(int a, int b, int gcd) { return (a * b) / gcd; }
}
```

Exercise06():

- Yêu cầu người dùng nhập hai số nguyên a và b lần lượt bằng log.writeLog() và scanner.nextInt().
- Gọi phương thức findGCD(a, b) để tính toán GCD của a và b. Lưu trữ kết quả trong biến gcd.
- Gọi phương thức findLCM(a, b, gcd) để tính toán LCM của a và b dựa trên GCD đã tính toán. Lưu trữ kết quả trong biến lcm.
- Ghi kết quả ra file log theo định dạng "Bai 06: A = %s, B = %s, GCD = %s, LCM = %s".

findGCD() (Thuật toán Euclid):

- Sử dụng thuật toán Euclid để tính toán GCD của a và b.
- Sử dụng vòng lặp while lặp lại cho đến khi b bằng 0.
- Trong mỗi vòng lặp:
- Tính toán temp bằng a % b (số dư của phép chia a cho b).
- Gán giá trị của b cho a.
- Gán giá trị của temp (số dư) cho b.
- Sau khi thoát vòng lặp, a sẽ chứa GCD của a và b ban đầu.

- Trả về giá trị GCD được lưu trữ trong a.

findLCM():

- Sử dụng công thức LCM(a, b) = (a * b) / GCD(a, b) để tính toán LCM dựa trên GCD đã được tính toán.
- Chia tích của a và b cho gcd để tìm LCM.
- Trả về giá trị LCM được tính toán.

9. Bài 07

```java
public class Exercise07 {
    1 usage  tuilakhanh
    public static void Exercise07(Scanner scanner, Log log) {
        log.writeLog("Bai 07: Nhap so nguyen n: ");
        var n = scanner.nextInt();
        log.writeLog(String.format("Bai 07: n = %s, Cac so nguyen to nho hon n: %s", n, findPrimeNumSmaller(n)));
    }

    5 usages  tuilakhanh
    public static boolean[] sieve(int n) {
        var isPrime = new boolean[n + 1];
        Arrays.fill(isPrime, val: true);
        isPrime[0] = isPrime[1] = false;
        for (int i = 2; i * i <= n; i++) {
            if (isPrime[i]) {
                for (int j = i * i; j <= n; j += i) {
                    isPrime[j] = false;
                }
            }
        }
        return isPrime;
    }

    1 usage  tuilakhanh
    static String findPrimeNumSmaller(int n) {
        var primes = sieve(n);
        return IntStream.rangeClosed(2, n) IntStream
                .filter(i -> primes[i])
                .mapToObj(String::valueOf) Stream<String>
                .collect(Collectors.joining( delimiter: ", "));
    }
}
```

Exercise07():

- Yêu cầu người dùng nhập một số nguyên n bằng log.writeLog() và scanner.nextInt().
- Gọi phương thức findPrimeNumSmaller(n) để tìm các số nguyên tố nhỏ hơn n. Lưu trữ kết quả trong biến primes.
- Ghi kết quả ra file log theo định dạng "Bai 07: n = %s, Cac so nguyen to nho hon n: %s".

sieve():

- Sử dụng thuật toán sàng số nguyên tô của Eratosthenes, tra về mảng boolean, nếu số n cần kiểm tra tại vị trí n của mảng boolean nếu là số nguyên tố thì giá trị sẽ là True nếu không thì ngược lại.

findPrimeNumSmaller():

- Gọi phương thức sieve(n) để tạo mảng primes chứa thông tin các số nguyên tố nhỏ hơn n.

- Sử dụng IntStream.rangeClosed(2, n) để tạo một luồng các số nguyên từ 2 đến n (bao gồm cả 2 và n).
- Lọc các số nguyên tố từ luồng bằng cách sử dụng filter(i -> primes[i]). Kiểm tra nếu phần tử tại index i của mảng primes là true (nguyên tố).
- Chuyển đổi các số nguyên tố còn lại thành chuỗi bằng mapToObj(String::valueOf).
- Nối các chuỗi số nguyên tố, ngăn cách nhau bởi dấu phẩy, thành một chuỗi duy nhất bằng collect(Collectors.joining(", ")).
- Trả về chuỗi chứa danh sách các số nguyên tố nhỏ hơn n.

10. Bài 08

```
public class Exercise08 {
    1 usage   tuilakhanh
    public static void Exercise08(Scanner scanner, Log log) {
        log.writeLog(String.format("Bai 08: So nguyen to co 5 chu so: %s", findPrimeWithFiveNum()));
    }

    1 usage   tuilakhanh
    static String findPrimeWithFiveNum() {
        var lowerLimit = 10000;
        var upperLimit = 99999;
        var primes = Exercise07.sieve(upperLimit);
        return IntStream.rangeClosed(lowerLimit, upperLimit) IntStream
                .filter(i -> primes[i])
                .mapToObj(String::valueOf) Stream<String>
                .collect(Collectors.joining( delimiter: ", "));
    }
}
```

Exercise08():

- Không yêu cầu người dùng nhập dữ liệu vì số nguyên tố cần tìm có phạm vi cố định (5 chữ số).
- Gọi phương thức findPrimeWithFiveNum() để tìm các số nguyên tố có 5 chữ số. Lưu trữ kết quả trong biến primes.
- Ghi kết quả ra file log theo định dạng "Bai 08: So nguyen to co 5 chu so: %s".

findPrimeWithFiveNum():

- Khởi tạo hai hằng số: lowerLimit: Giới hạn dưới (10000) - số nguyên tố có 5 chữ số nhỏ nhất. upperLimit: Giới hạn trên (99999) - số nguyên tố có 5 chữ số lớn nhất.
- Gọi phương thức Exercise07.sieve(upperLimit) để tạo mảng primes chứa thông tin các số nguyên tố nhỏ hơn hoặc bằng upperLimit (sử dụng kết quả từ Bài tập 7).
- Sử dụng IntStream.rangeClosed(lowerLimit, upperLimit) để tạo một luồng các số nguyên từ lowerLimit đến upperLimit (bao gồm cả hai giá trị).
- Lọc các số nguyên tố từ luồng bằng cách sử dụng filter(i -> primes[i]). Kiểm tra nếu phần tử tại index i của mảng primes là true (nguyên tố).
- Chuyển đổi các số nguyên tố còn lại thành chuỗi bằng mapToObj(String::valueOf).
- Nối các chuỗi số nguyên tố, ngăn cách nhau bởi dấu phẩy, thành một chuỗi duy nhất bằng collect(Collectors.joining(", ")).
- Trả về chuỗi chứa danh sách các số nguyên tố có 5 chữ số.

11. Bài 09

```java
public class Exercise09 {

    1 usage   tuilakhanh
    public static void Exercise09(Scanner scanner, Log log) {
        log.writeLog("Bai 09: Nhap so nguyen n: ");
        var n = scanner.nextInt();
        log.writeLog(String.format("Bai 09: So nguyen n = %s, Phan tich: %s", n, findPrimeFactorization(n)));
    }
    1 usage   tuilakhanh
    static String findPrimeFactorization(int n) {
        var upperLimit = n;
        var isPrime = Exercise07.sieve(upperLimit);

        var primeFactors = new ArrayList<Integer>();

        while (n > 1) {
            for (int i = 2; i <= upperLimit; i++) {
                if (isPrime[i] && n % i == 0) {
                    n /= i;
                    primeFactors.add(i);
                    break;
                }
            }
        }

        return primeFactors.stream() Stream<Integer>
                .map(String::valueOf) Stream<String>
                .collect(Collectors.joining( delimiter: "x"));
    }
}
```

Exercise09():

- Yêu cầu người dùng nhập một số nguyên n bằng log.writeLog() và scanner.nextInt().
- Gọi phương thức findPrimeFactorization(n) để phân tích n thành các thừa số nguyên tố. Lưu trữ kết quả trong biến primeFactors.
- Ghi kết quả ra file log theo định dạng "Bai 09: So nguyen n = %s, Phan tich: %s".

findPrimeFactorization():

- Khởi tạo hằng upperLimit bằng n để giới hạn tìm kiếm các ước nguyên tố tiềm năng (tối đa bằng n).
- Gọi phương thức Exercise07.sieve(upperLimit) để tạo mảng isPrime chứa thông tin các số nguyên tố nhỏ hơn hoặc bằng upperLimit (sử dụng kết quả từ Bài tập 7).
- Khởi tạo danh sách primeFactors để lưu trữ các thừa số nguyên tố phân tích được.
- Sử dụng vòng lặp while cho đến khi n nhỏ hơn hoặc bằng 1 (phân tích xong).

- Sử dụng vòng lặp for để duyệt từ 2 đến upperLimit để tìm ước nguyên tố của n.
- Kiểm tra điều kiện:
- Phần tử i tại index i của mảng isPrime phải là true (nguyên tố).
- Số n phải chia hết cho i (n % i == 0).
- Nếu điều kiện thỏa mãn, thực hiện các bước sau:
- Cập nhật n bằng n chia cho i (loại bỏ ước nguyên tố i đã tìm thấy).
- Thêm i vào danh sách primeFactors (là một thừa số nguyên tố).
- Thoát khỏi vòng lặp for nội bằng break (chỉ cần tìm 1 ước nguyên tố tại mỗi bước lặp).
- Chuyển đổi các thừa số nguyên tố trong danh sách primeFactors thành chuỗi (mỗi phần tử) bằng stream().map(String::valueOf).
- Nối các chuỗi tạo thành, ngăn cách nhau bởi dấu nhân "x", thành một chuỗi duy nhất bằng collect(Collectors.joining("x")).
- Trả về chuỗi thể hiện phân tích thừa số nguyên tố của n.

12. Bài 10

```
public class Exercise10 {
    1 usage   tuilakhanh
    public static void Exercise10(Scanner scanner, Log log) {
        log.writeLog("Bai 10: Nhap chuoi: ");
        var s = scanner.nextLine();
        log.writeLog(String.format("Bai 10: Chuoi: %s, co %s ky tu", s, countWords(s)));
    }

    1 usage   tuilakhanh
    static long countWords(String s) {
        return Stream.of(s.split( regex: "\\s+"))
                .count();
    }
}
```

Exercise10():

- Yêu cầu người dùng nhập một chuỗi văn bản bằng log.writeLog() và scanner.nextLine(). Lưu trữ chuỗi nhập vào trong biến s.
- Gọi phương thức countWords(s) để đếm số từ trong chuỗi s. Lưu trữ kết quả đếm trong biến wordCount.
- Ghi kết quả ra file log theo định dạng "Bai 10: Chuoi: %s, co %s ky tu".

countWords():

- Sử dụng s.split("\\s+") để tách chuỗi s thành một mảng các từ dựa trên các khoảng trắng (\\s+ biểu thức regex cho một hoặc nhiều khoảng trắng).
- Tạo một luồng (Stream) từ mảng các từ bằng Stream.of(...).
- Đếm số lượng phần tử (từ) trong luồng bằng .count().
- Trả về kết quả đếm số từ (kiểu long).

13. Bài 11

```
public class Exercise11 {
    1 usage  tuilakhanh
    public static void Exercise11(Scanner scanner, Log log) {
        log.writeLog("Bai 11: Nhap chuoi: ");
        var s = scanner.nextLine();
        log.writeLog(String.format("Bai 11: Chuoi: %s, Co so lan xuat hien cac tu: %s", s, countWordOccurrences(s)));
    }

    1 usage  tuilakhanh
    static String countWordOccurrences(String s) {
        return Stream.of(s.split( regex: "\\s+")) Stream<String>
                .collect(Collectors.groupingBy(Function.identity(), Collectors.counting())) Map<String, Long>
                .entrySet() Set<Entry<...>>
                .stream() Stream<Entry<...>>
                .map(entry -> String.format("%s = %s", entry.getKey(), entry.getValue())) Stream<String>
                .collect(Collectors.joining( delimiter: ", ")); // Join entries with newline
    }
}
```

Exercise11():

- Yêu cầu người dùng nhập một chuỗi văn bản bằng log.writeLog() và scanner.nextLine(). Lưu trữ chuỗi nhập vào trong biến s.
- Gọi phương thức countWordOccurrences(s) để đếm số lần xuất hiện của mỗi từ trong chuỗi s. Lưu trữ kết quả đếm trong biến wordCounts.
- Chuyển đổi wordCounts thành chuỗi hiển thị số lần xuất hiện của mỗi từ và in kết quả ra file log theo định dạng "Bai 11: Chuoi: %s, Co so lan xuat hien cac tu: %s".

countWordOccurrences():

- Sử dụng s.split("\\s+") để tách chuỗi s thành một mảng các từ dựa trên các khoảng trắng (\\s+ biểu thức chính quy cho một hoặc nhiều khoảng trắng).
- Tạo một luồng (Stream) từ mảng các từ bằng Stream.of(...).
- Sử dụng Collectors.groupingBy(Function.identity(), Collectors.counting()) để nhóm các từ theo giá trị (tức là từ giống nhau) và đếm số lần xuất hiện của mỗi nhóm:
- Function.identity() giữ nguyên giá trị của mỗi từ (khóa của nhóm).
- Collectors.counting() đếm số phần tử trong mỗi nhóm (số lần xuất hiện của từ).
- Chuyển đổi kết quả thành một tập hợp các cặp key-value (từ và số lần xuất hiện) bằng .entrySet().
- Tạo một luồng mới từ tập hợp entrySet.
- Sử dụng map để chuyển đổi mỗi cặp key-value thành chuỗi theo định dạng "<từ> = <số lần xuất hiện>" (ví dụ: "apple = 3").

- Nối các chuỗi tạo thành, ngăn cách nhau bởi dấu phẩy (", "), thành một chuỗi duy nhất bằng Collectors.joining(", ").

14. Bài 12

```
public class Exercise12 {
    1 usage   tuilakhanh
    public static void Exercise12(Scanner scanner, Log log) {
        log.writeLog("Bai 12: Nhap chuoi s1: ");
        var s1 = scanner.nextLine();
        log.writeLog("Bai 12: Nhap chuoi s2: ");
        var s2 = scanner.nextLine();
        log.writeLog(String.format("Bai 11: Chuoi s1: %s, Chuoi s2: %s, %s", s1, s2, isStringContain(s1
    }

    1 usage   tuilakhanh
    public static String isStringContain(String s1, String s2) {
        return s1.contains(s2) ? "s1 co chua s2" : "s1 khong chua s2";
    }
}
```

Exercise12():

- Yêu cầu người dùng nhập hai chuỗi s1 và s2 lần lượt bằng log.writeLog() và scanner.nextLine().
- Gọi phương thức isStringContain(s1, s2) để kiểm tra xem s1 có chứa s2 hay không. Lưu trữ kết quả kiểm tra trong biến result.
- Ghi kết quả ra file log theo định dạng "Bai 11: Chuoi s1: %s, Chuoi s2: %s, %s".

isStringContain():

- Sử dụng phương thức contains(String s2) của lớp String để kiểm tra xem chuỗi s1 có chứa chuỗi s2 hay không.
- Sử dụng toán tử điều kiện ba ngôi (?:) để trả về chuỗi kết quả:
- Nếu s1.contains(s2) là true (nghĩa là s1 chứa s2), trả về "s1 co chua s2".
- Ngược lại, trả về "s1 khong chua s2".

15. Bài 13

```java
public class Exercise13 {
    2 usages  tuilakhanh
    record TwoLargest(int firstElement, int firstIndex, int secondElement, int secondIndex) {}

    1 usage  tuilakhanh
    public static void Exercise13(Scanner scanner, Log log) {
        log.writeLog("Bai 13: Nhap so phan tu cua mang: ");
        var n = scanner.nextInt();
        var A = new int[n];

        log.writeLog("Bai 13: Nhap " + n + " phan tu (so nguyen lon hon 0 va nho hon 100):");
        for (int i = 0; i < n; i++) {
            do {
                log.writeLog("Nhap phan tu thu " + i + ": ");
                A[i] = scanner.nextInt();
            } while (A[i] < 1 || A[i] > 99);
        }

        log.writeLog(String.format("Bai 13: Mang A: %s", printArray(A)));

        var largest = findTwoLargest(A);
        log.writeLog(String.format("Bai 13: Phan tu lon nhat la %s tai vi tri %s", largest.firstElement, largest.firstIndex));
        log.writeLog(String.format("Bai 13: Phan tu lon thu hai la %s tai vi tri %s", largest.secondElement, largest.secondIndex()));

        Arrays.sort(A);
        log.writeLog(String.format("Bai 13: Mang A sau khi sap xep: %s", printArray(A)));

        log.writeLog("Bai 13: Nhap so nguyen x muon them vao mang: ");
        var x = scanner.nextInt();
        log.writeLog(String.format("Bai 13: Mang A sau khi them x: %s", printArray(insertElement(A, x))));
    }

    3 usages  tuilakhanh
    static String printArray(int[] A) {
        return Arrays.stream(A) IntStream
                .mapToObj(String::valueOf) Stream<String>
                .collect(Collectors.joining( delimiter: ", "));
    }
```

```java
static TwoLargest findTwoLargest(int[] A) {
    int firstIndex = 0, secondIndex = -1;
    int first = A[0], second = Integer.MIN_VALUE;

    for (int i = 1; i < A.length; i++) {
        if (A[i] > first) {
            second = first;
            secondIndex = firstIndex;
            first = A[i];
            firstIndex = i;
        } else if (A[i] > second && A[i] != first) {
            second = A[i];
            secondIndex = i;
        }
    }

    return new TwoLargest(first, firstIndex, second, secondIndex);
}

1 usage  tuilakhanh
static int[] insertElement(int[] A, int x) {
    int insertIndex = Arrays.binarySearch(A, x);
    if (insertIndex < 0) {
        insertIndex = -(insertIndex + 1);
    }

    int[] inserted = new int[A.length + 1];
    System.arraycopy(A, i: 0, inserted, i1: 0, insertIndex);
    inserted[insertIndex] = x;
    System.arraycopy(A, insertIndex, inserted, i1: insertIndex + 1, i2: A.length - insertIndex);
    return inserted;
}
```

Record TwoLargest:

- Giữ thông tin về hai phần tử lớn nhất trong mảng: firstElement (phần tử lớn nhất), firstIndex (vị trí của phần tử lớn nhất), secondElement (phần tử lớn thứ hai), secondIndex (vị trí của phần tử lớn thứ hai).

Exercise13():

- Yêu cầu người dùng nhập số phần tử của mảng n và tạo mảng A kích thước n.
- Nhập từng phần tử của mảng A, đảm bảo nằm trong phạm vi 1 đến 99, sử dụng vòng lặp do-while để lặp lại yêu cầu nhập cho đến khi nhập hợp lệ.
- In ra mảng A đã nhập.
- Gọi phương thức findTwoLargest(A) để tìm hai phần tử lớn nhất và vị trí của chúng, lưu trữ kết quả trong biến largest (record TwoLargest).
- In ra thông tin phần tử lớn nhất và lớn thứ hai cùng vị trí tương ứng.
- Sắp xếp mảng A theo thứ tự tăng dần bằng Arrays.sort(A).
- In ra mảng A sau khi sắp xếp.
- Yêu cầu người dùng nhập một số nguyên x để thêm vào mảng.
- Gọi phương thức insertElement(A, x) để thêm x vào mảng A tại vị trí thích hợp, lưu trữ kết quả trong biến inserted.
- In ra mảng A sau khi thêm phần tử x.

printArray():

- Sử dụng Stream API để chuyển đổi từng phần tử của mảng A thành chuỗi, sau đó nối các chuỗi tạo thành, ngăn cách nhau bởi dấu phẩy (", "), thành một chuỗi duy nhất.

findTwoLargest():

- Khởi tạo các biến:
- firstIndex và secondIndex để lưu trữ vị trí ban đầu của phần tử lớn nhất và lớn thứ hai (gán giá trị mặc định).
- first và second để lưu trữ giá trị của phần tử lớn nhất và lớn thứ hai (gán giá trị mặc định).
- Duyệt qua mảng A từ phần tử thứ hai (vị trí 1) đến phần tử cuối cùng.
- Kiểm tra điều kiện:
- Nếu phần tử A[i] lớn hơn first:
- Cập nhật second bằng first, secondIndex bằng firstIndex.

- Cập nhật first bằng A[i], firstIndex bằng i.
- Ngược lại, nếu A[i] lớn hơn second và khác first:
- Cập nhật second bằng A[i], secondIndex bằng i.
- Trả về một record TwoLargest chứa thông tin hai phần tử lớn nhất và vị trí tương ứng.

insertElement():

- Chèn x vào vị trí insertIndex trong mảng inserted.
- Sao chép các phần tử còn lại từ A sang inserted sau vị trí chèn (insertIndex).
- Trả về mảng inserted sau khi thêm x và duy trì tính sắp xếp.

16. Bài 14

```java
public class Exercise14 {
    3 usages  tuilakhanh
    record Largest(int largest, int rowIndex, int colIndex) {}
    1 usage  tuilakhanh
    public static void Exercise14(Scanner scanner, Log log) {
        log.writeLog("Bai 14: Nhap so dong: ");
        var n = scanner.nextInt();
        log.writeLog("Bai 14: Nhap so cot: ");
        var m = scanner.nextInt();

        var matrix = new int[n][m];
        log.writeLog("Bai 14: Nhap phan tu cua ma tran: ");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                do {
                    log.writeLog(String.format("Nhap phan tu tai %sx%s: ", i, j));
                    matrix[i][j] = scanner.nextInt();
                } while (matrix[i][j]  < 1 || matrix[i][j]  > 99);
            }
        }
        log.writeLog("Bai 14: Ma tran sau khi nhap: ");
        log.writeLog(printMatrix(matrix));

        Largest resultA = findLargest(matrix);
        log.writeLog(String.format("Bai 14: Phan tu lon nhat: %s, tai vi tri %sx%s", resultA.largest, resultA.rowIndex, resultA.colIndex));

        var primeMatrix = primeMatrix(matrix);
        log.writeLog("Bai 14: Ma tran voi phan nguyen to:");
        log.writeLog(printMatrix(primeMatrix));

        sortColumns(matrix);
        log.writeLog("Bai 14: Ma tran da duoc sap xep theo thu tu tang dan cua cot:");
        log.writeLog(printMatrix(matrix));

        log.writeLog(String.format("Bai 14: Cot co nhieu so nguyen to nhat: %s", findMostPrimeCol(matrix)));
    }

    3 usages  tuilakhanh
    static String printMatrix(int[][] matrix) {
        return "\n" + Arrays.stream(matrix) Stream<int[]>
                .map(row -> String.join( delimiter: " ", Arrays.stream(row) IntStream
                                .mapToObj(String::valueOf) Stream<String>
                                .collect(Collectors.toList())
                        )
                ) Stream<String>
                .collect(Collectors.joining( delimiter: "\n"));
    }
```

```
static Largest findLargest(int[][] matrix) {
    var largest = Integer.MIN_VALUE;
    int rowIndex = -1, colIndex = -1;
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix[i].length; j++) {
            if (matrix[i][j] > largest) {
                largest = matrix[i][j];
                rowIndex = i;
                colIndex = j;
            }
        }
    }
    return new Largest(largest, rowIndex, colIndex);
}

1 usage  tuilakhanh
static int[][] primeMatrix(int[][] matrix) {
    var primeMatrix = new int[matrix.length][matrix[0].length];
    var isPrime = Exercise07.sieve( n: 100);
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix[i].length; j++) {
            if (!isPrime[matrix[i][j]]) {
                primeMatrix[i][j] = 0;
            }
            else {
                primeMatrix[i][j] = matrix[i][j];
            }
        }
    }
    return primeMatrix;
}

1 usage  tuilakhanh
static void sortColumns(int[][] matrix) {
    for (int col = 0; col < matrix[0].length; col++) {
        var column = new ArrayList<Integer>();
        for (int row = 0; row < matrix.length; row++) {
            column.add(matrix[row][col]);
        }
        column.sort(Comparator.naturalOrder());
        for (int row = 0; row < matrix.length; row++) {
            matrix[row][col] = column.get(row);
        }
    }
}
```

```java
static int findMostPrimeCol(int[][] matrix) {
    var isPrime = Exercise07.sieve( n: 100);
    var primeCount = new int[matrix[0].length];
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix[i].length; j++) {
            if (isPrime[matrix[i][j]]) {
                primeCount[j]++;
            }
        }
    }
    return Arrays.stream(primeCount) IntStream
            .max() OptionalInt
            .getAsInt();
}
```

Record Largest:

- Giữ thông tin về phần tử lớn nhất trong ma trận: largest (giá trị), rowIndex (vị trí dòng), colIndex (vị trí cột).

Exercise14():

- Yêu cầu người dùng nhập kích thước (số dòng, số cột) của ma trận n và m.
- Tạo ma trận matrix kích thước n x m.
- Yêu cầu người dùng nhập từng phần tử của ma trận matrix, đảm bảo nằm trong phạm vi 1 đến 99, sử dụng vòng lặp do-while để lặp lại yêu cầu nhập cho đến khi nhập hợp lệ.
- In ra ma trận đã nhập.
- Gọi phương thức findLargest(matrix) để tìm phần tử lớn nhất và vị trí của nó, lưu trữ kết quả trong biến resultA (record Largest).
- In ra thông tin phần tử lớn nhất và vị trí tương ứng.
- Gọi phương thức primeMatrix(matrix) để tạo ma trận mới với các số nguyên tố, lưu trữ kết quả trong biến primeMatrix.
- In ra ma trận mới với các số nguyên tố.
- Gọi phương thức sortColumns(matrix) để sắp xếp các phần tử trong mỗi cột theo thứ tự tăng dần.
- In ra ma trận sau khi sắp xếp các cột.
- Gọi phương thức findMostPrimeCol(matrix) để tìm cột có chứa nhiều số nguyên tố nhất, lưu trữ kết quả trong biến mostPrimeCol.
- In ra cột có chứa nhiều số nguyên tố nhất.

printMatrix():

- Sử dụng Stream API để chuyển đổi từng dòng của ma trận thành chuỗi, mỗi phần tử trong dòng được ngăn cách nhau bởi dấu cách.
- Nối các chuỗi đại diện cho các dòng, tạo thành chuỗi hiển thị toàn bộ ma trận.

findLargest():

- Khởi tạo các biến:
- largest để lưu trữ giá trị phần tử lớn nhất (gán giá trị mặc định).
- rowIndex và colIndex để lưu trữ vị trí dòng và cột của phần tử lớn nhất (gán giá trị mặc định).
- Duyệt qua các phần tử của ma trận matrix.

- Kiểm tra điều kiện: Nếu phần tử hiện tại lớn hơn largest, cập nhật largest, rowIndex, và colIndex tương ứng.
- Trả về một record Largest chứa thông tin phần tử lớn nhất và vị trí tương ứng.

primeMatrix():

- Tạo ma trận mới primeMatrix có cùng kích thước với matrix.
- Sử dụng kết quả từ phương thức Exercise07.sieve(100) (giả sử đây là phương thức kiểm tra số nguyên tố) để xác định các số nguyên tố.
- Duyệt qua các phần tử của matrix. Nếu phần tử không phải số nguyên tố, gán 0 vào vị trí tương ứng trong primeMatrix. Ngược lại, sao chép giá trị của phần tử từ matrix sang primeMatrix.
- Trả về ma trận primeMatrix mới.

sortColumns():

- Duyệt qua từng cột của ma trận matrix.
- Tạo một danh sách column để lưu trữ tạm thời các phần tử trong cột hiện tại.
- Thêm các phần tử trong cột hiện tại của matrix vào danh sách column.
- Sắp xếp danh sách column theo thứ tự tăng dần.
- Gán các phần tử được sắp xếp trong danh sách column trở lại vị trí tương ứng trong cột của ma trận matrix.

findMostPrimeCol():

- Sử dụng kết quả từ phương thức Exercise07.sieve(100) (giả sử đây là phương thức kiểm tra số nguyên tố) để xác định các số nguyên tố.
- Tạo mảng primeCount có kích thước bằng số cột của ma trận matrix.
- Duyệt qua các phần tử của matrix. Nếu phần tử là số nguyên tố, tăng giá trị tương ứng trong mảng primeCount lên 1.
- Tìm giá trị lớn nhất trong mảng primeCount.
- Trả về vị trí (chỉ số) của giá trị lớn nhất trong mảng primeCount, đại diện cho cột có chứa nhiều số nguyên tố nhất.

17. Bài 15

```
public class Exercise15 {

    1 usage   tuilakhanh
    public static void Exercise15(Scanner scanner, Log log) {
        var codes = new String[5];

        for (int i = 0; i < 5; i++) {
            do {
                log.writeLog(String.format("Bai 15: Nhap code thu %s: ", i));
                codes[i] = scanner.nextLine();
            } while (!isValidCode(codes[i]));
        }

        log.writeLog(String.format("Bai 15: Code da nhap: %s", String.join( delimiter: ", ", codes)));
    }

    1 usage   tuilakhanh
    static boolean isValidCode(String code) { return Pattern.matches( regex: "^00[2-5]L[0-9]{4}$", code); }
}
```

Exercise15():

- Tạo một mảng codes kiểu String để lưu trữ 5 mã code.
- Sử dụng vòng lặp for để lặp lại 5 lần (i = 0 đến i < 5).
- Trong mỗi lần lặp:
- Yêu cầu người dùng nhập mã code thứ i + 1 bằng log.writeLog với nội dung được định dạng.
- Gọi phương thức isValidCode(codes[i]) để kiểm tra tính hợp lệ của mã code vừa nhập.
- Sử dụng vòng lặp do-while để lặp lại yêu cầu nhập cho đến khi nhập mã code hợp lệ.
- Nếu mã code hợp lệ, thoát khỏi vòng lặp do-while.
- Sau vòng lặp for:
- Sử dụng log.writeLog để in ra danh sách các mã code đã nhập, nối các mã code với nhau bằng dấu phẩy (", ").

isValidCode():

- Sử dụng biểu thức chính quy (Pattern.matches) để kiểm tra xem chuỗi code có khớp với mẫu định sẵn hay không.
- Biểu thức regex được sử dụng trong phương thức này là: ^00[2-5]L[0-9]{4}$

  ^: Bắt đầu chuỗi.

  00: Chuỗi phải bắt đầu bằng "00".

[2-5]: Kí tự tiếp theo phải nằm trong khoảng từ 2 đến 5.

L: Kí tự tiếp theo phải là "L".

[0-9]{4}: Bốn ký tự tiếp theo phải là các chữ số (0-9).

$: Kết thúc chuỗi.

- Trả về true nếu code khớp với mẫu, false nếu không khớp.

18. Menu dành cho Lab01

```
public class Menu {
    17 usages
    private Scanner scanner;

    19 usages
    private Log log;

    1 usage  tuilakhanh
    public Menu(Scanner scanner, Log log) {
        this.scanner = scanner;
        this.log = log;
    }

    1 usage  tuilakhanh
    public int displayMenu() {
        var menu = new StringBuffer();
        menu.append("\nMenu:\n");
        menu.append("01. Bai 01: Tim tat ca cac so chia het cho 7 nhung khong phai boi so cua 5 tu 10 den 200\n");
        menu.append("02. Bai 02: Tinh Giai thua cua so n\n");
        menu.append("03. Bai 03: Tinh map(i, i*i)\n");
        menu.append("04. Bai 04: Chuyen doi so thap phan sang he co so B\n");
        menu.append("05. Bai 05: In ra N so Fibonacci dau tien\n");
        menu.append("06. Bai 06: Tim UCLN va BCLN cua a va b\n");
        menu.append("07. Bai 07: In ra cac so nguyen to nho hon n\n");
        menu.append("08. Bai 08: Tim so nguyen to co 5 chu so\n");
        menu.append("09. Bai 09: Phan tich so nguyen n thanh cac thua so nguyen to\n");
        menu.append("10. Bai 10: Dem so ky tu trong chuoi\n");
        menu.append("11. Bai 11: Dem so lan xuat hien cac tu trong chuoi\n");
        menu.append("12. Bai 12: Kiem tra s1 co phai la xau con cua s2 khong\n");
        menu.append("13. Bai 13: Xu ly mang\n");
        menu.append("14. Bai 14: Xu ly ma tran\n");
        menu.append("15. Bai 15: Nhap code\n");
        menu.append("0 Thoat\n");
        menu.append("Nhap lua chon: ");
        log.writeLog(menu.toString());
        int choice = scanner.nextInt();
        scanner.nextLine();
        return choice;
    }

    1 usage  tuilakhanh
    public void handleChoice(int choice) {
        switch (choice) {
            case 1 -> Exercise01.Exercise01(log);
            case 2 -> Exercise02.Exercise02(scanner, log);
            case 3 -> Exercise03.Exercise03(scanner, log);
            case 4 -> Exercise04.Exercise04(scanner, log);
            case 5 -> Exercise05.Exercise05(scanner, log);
            case 6 -> Exercise06.Exercise06(scanner, log);
            case 7 -> Exercise07.Exercise07(scanner, log);
```

B. TÀI LIỆU THAM KHẢO