Questão 01

```
package questao01;

import java.util.Scanner;

public class App {
    public static void main(String[] args) {
        Scanner leitorDeEntrada = new Scanner(System.in);

        while (true) {
            System.out.println("Digite, respectivamente, a altura e
largura do retângulo a ser impresso.");

            int altura = leitorDeEntrada.nextInt();
            int largura = leitorDeEntrada.nextInt();

            imprimeRetangulo(altura, largura);

            leitorDeEntrada.nextLine();
            System.out.println("Digite '-1' para sair ou qualquer
tecla para continuar.");
            String entrada = leitorDeEntrada.nextLine();

            if ("-1".equals(entrada)) {
                break;
            }
        }

        leitorDeEntrada.close();
    }

    private static void imprimeRetangulo(int altura, int largura) {
        if (altura <= 0 || largura <= 0) {
            System.out.println("Dimensões inválidas.");
            return;
        }

        imprimeLargura(largura);
        imprimeAltura(altura, largura);

        if (altura > 1) {
            imprimeLargura(largura);
        }
    }

    private static void imprimeLargura(int largura) {
        for (int i = 0; i < largura; i++) {
            System.out.print("X");
        }
        System.out.println();
    }

    private static void imprimeAltura(int altura, int largura) {
        for (int i = 0; i < altura - 2; i++) {
```

```
            imprimeLinhaMeio(largura);
        }
    }

    private static void imprimeLinhaMeio(int largura) {
        if (largura > 1) {
            System.out.print("X");
            for (int i = 0; i < largura - 2; i++) {
                System.out.print(" ");
            }
            System.out.println("X");
        } else {
            System.out.println("X");
        }
    }
}
```

Questão 02

```
package questao02;

import java.util.ArrayList;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.Scanner;
import java.util.stream.Collectors;

public class App {
    public static void main(String[] args) {
        menu();
    }

    public static void menu() {
        Scanner leitorDeEntrada = new Scanner(System.in);

        while (true) {

System.out.println("==========================================");
            System.out.println("1 - Somar dois números");
            System.out.println("2 - O maior de dois números");
            System.out.println("3 - Somar N números");
            System.out.println("4 - Contador de pares de uma
sequência");
            System.out.println("0 - Sair");

System.out.println("==========================================");
            System.out.print("Opção: ");

            int opcaoSelecionada = lerInteiroValido(leitorDeEntrada);

            if (opcaoSelecionada == 0) {
                System.out.println("Saindo...");
                leitorDeEntrada.close();
                break;
```

```
            }

            if (opcaoSelecionada < 0 || opcaoSelecionada > 4) {
                System.out.println("Opção inválida. Tente
novamente.");
                continue;
            }

            selecionaOpcao(opcaoSelecionada, leitorDeEntrada);
        }
    }

    public static void selecionaOpcao(int opcaoSelecionada, Scanner
leitorDeEntrada) {
        List<Integer> numeros = obtemListaDeNumeros(leitorDeEntrada);

        switch (opcaoSelecionada) {
            case 1 -> System.out.println("Soma dos números: " +
somaNumeros(numeros));
            case 2 -> System.out.println("Maior número: " +
verificaOMaior(numeros));
            case 3 -> System.out.println("Soma dos números: " +
somaNumeros(numeros));
            case 4 -> System.out.printf("Números pares: %s (%s)%n",
contaOsPares(numeros), exibeOsPares(numeros));
        }
    }

    public static List<Integer> obtemListaDeNumeros(int quantidade,
Scanner leitorDeEntrada) {
        List<Integer> inputs = new ArrayList<>();

        System.out.println("Digite " + quantidade + " número(s):");
        while (inputs.size() < quantidade) {
            inputs.add(lerInteiroValido(leitorDeEntrada));
        }
        return inputs;
    }

    public static List<Integer> obtemListaDeNumeros(Scanner
leitorDeEntrada) {
        System.out.print("Quantos números deseja inserir? ");
        int quantidade = lerInteiroValido(leitorDeEntrada);
        return obtemListaDeNumeros(quantidade, leitorDeEntrada);
    }

    public static int somaNumeros(List<Integer> numeros) {
        return numeros.stream().mapToInt(Integer::intValue).sum();
    }

    public static int verificaOMaior(List<Integer> numeros) {
        return numeros.stream()
                .max(Integer::compareTo)
                .orElseThrow(() -> new NoSuchElementException("Não há
```

```
números na lista."));
    }

    public static int contaOsPares(List<Integer> numeros) {
        return (int) numeros.stream()
                .filter(n -> n % 2 == 0)
                .count();
    }

    public static String exibeOsPares(List<Integer> numeros) {
        List<Integer> pares = numeros.stream()
                .filter(n -> n % 2 == 0)
                .toList();
        return pares.isEmpty() ? "Nenhum número par encontrado." :
                pares.stream()
                        .map(String::valueOf)
                        .collect(Collectors.joining(", "));
    }

    public static int lerInteiroValido(Scanner scanner) {
        while (!scanner.hasNextInt()) {
            System.out.println("Entrada inválida. Digite um número
inteiro.");
            scanner.next();
        }
        return scanner.nextInt();
    }
}
```

Questão 03

```
package questao03;

import java.math.BigDecimal;
import java.util.*;

public class App {

    public static void main(String[] args) {
        recebeNotas();
    }

    public static void recebeNotas() {
        Scanner leitorDeEntrada = new Scanner(System.in);

        System.out.println("Digite as notas dos 4 exercícios
realizados:");
        List<BigDecimal> notasExercicios = (List<BigDecimal>)
obtemNotas(4, leitorDeEntrada);

        System.out.println("Digite a nota das 2 provas realizadas:");
        List<BigDecimal> notasProvas = (List<BigDecimal>)
obtemNotas(2, leitorDeEntrada);
```

```
        System.out.println("Digite a nota do trabalho prático
realizado:");
        BigDecimal notaTrabalho = (BigDecimal) obtemNotas(1,
leitorDeEntrada);

        Map<String, Object> mapaGeralDeNotas = new HashMap<>();
        mapaGeralDeNotas.put("exercicios", notasExercicios);
        mapaGeralDeNotas.put("provas", notasProvas);
        mapaGeralDeNotas.put("trabalho", notaTrabalho);

        BigDecimal notaFinal = calculaNotaFinal(mapaGeralDeNotas);

        System.out.println("A nota final do aluno é: " + notaFinal);
    }

    public static Object obtemNotas(Integer quantidade, Scanner
leitorDeEntrada) {
        if (quantidade < 2) {
            return BigDecimal.valueOf(leitorDeEntrada.nextInt());
        }

        List<BigDecimal> inputs = new ArrayList<>();

        while (inputs.size() < quantidade) {

inputs.add(BigDecimal.valueOf(leitorDeEntrada.nextInt()));
            leitorDeEntrada.nextLine();
        }

        return inputs;
    }

    public static BigDecimal calculaNotaFinal(Map<String, Object>
mapaGeralDeNotas) {
        List<BigDecimal> exercicios = (List<BigDecimal>)
mapaGeralDeNotas.get("exercicios");
        List<BigDecimal> provas = (List<BigDecimal>)
mapaGeralDeNotas.get("provas");
        BigDecimal trabalho = (BigDecimal)
mapaGeralDeNotas.get("trabalho");

        BigDecimal somaExercicios = exercicios.stream()
                .reduce(BigDecimal.ZERO, BigDecimal::add);
        BigDecimal mediaExercicios =
somaExercicios.divide(BigDecimal.valueOf(exercicios.size()),
BigDecimal.ROUND_HALF_UP);

        BigDecimal somaProvas = provas.stream()
                .reduce(BigDecimal.ZERO, BigDecimal::add);
        BigDecimal mediaProvas =
somaProvas.divide(BigDecimal.valueOf(provas.size()),
BigDecimal.ROUND_HALF_UP);

        BigDecimal notaFinal =
mediaExercicios.multiply(BigDecimal.valueOf(0.4))
```

```
                .add(mediaProvas.multiply(BigDecimal.valueOf(0.5)))
                .add(trabalho.multiply(BigDecimal.valueOf(0.1)));

        return notaFinal.setScale(2, BigDecimal.ROUND_HALF_UP);
    }
}
```

Questão 04

```
package questao04;

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

public class App {
    public static void main(String[] args) {
        recebeAMensagem();
    }

    public static void recebeAMensagem() {
        Scanner leitorDeEntrada = new Scanner(System.in);

        while (true) {
            System.out.println("Digite a mensagem:");

            String mensagem = leitorDeEntrada.nextLine();

            if ("-1".equals(mensagem)) {
                break;
            }

            System.out.println(criptografaAMensagem(mensagem));
        }
    }

    public static String criptografaAMensagem(String mensagem) {
        List<String> colunas = preparaAsColunas(mensagem);
        return adicionaOsAsteriscos(colunas);
    }

    public static List<String> preparaAsColunas(String mensagem) {
        int numeroDeColunas = 5;
        int numeroDeLinhas = (int) Math.ceil((double)
mensagem.length() / numeroDeColunas);

        char[][] matriz = new char[numeroDeLinhas][numeroDeColunas];

        int index = 0;
        for (int i = 0; i < numeroDeLinhas; i++) {
            for (int j = 0; j < numeroDeColunas; j++) {
                if (index < mensagem.length()) {
                    matriz[i][j] = mensagem.charAt(index);
                    index++;
                } else {
```

```
                    matriz[i][j] = ' ';
                }
            }
        }

        List<String> palavrasResultantes = new ArrayList<>();
        for (int coluna = 0; coluna < numeroDeColunas; coluna++) {
            StringBuilder novaPalavra = new StringBuilder();
            for (int linha = 0; linha < numeroDeLinhas; linha++) {
                novaPalavra.append(matriz[linha][coluna]);
            }
            palavrasResultantes.add(novaPalavra.toString());
        }

        return palavrasResultantes;
    }

    public static String adicionaOsAsteriscos(List<String> palavras)
{
        return String.join("*", palavras);
    }
}
```

Questão 05

```
package questao05;

import java.util.Scanner;

public class App {
    public static void main(String[] args) {
        recebeAMensagem();
    }

    public static void recebeAMensagem() {
        Scanner leitorDeEntrada = new Scanner(System.in);

        while (true) {
            System.out.println("Digite a mensagem criptografada:");
            String mensagemCriptografada =
leitorDeEntrada.nextLine();

            if ("-1".equals(mensagemCriptografada)) {
                break;
            }

            if (!mensagemEhValida(mensagemCriptografada)) {
                System.out.println("Mensagem inválida! Envie uma
mensagem dentro do padrão.");
                continue;
            }


System.out.println(descriptografaAMensagem(mensagemCriptografada));
        }
```

```
    }

    public static boolean mensagemEhValida(String mensagem) {
        if (!mensagem.contains("*")) {
            return false;
        }

        String[] colunas = mensagem.split("\\*");
        int tamanhoEsperado = colunas[0].length();

        for (String coluna : colunas) {
            if (coluna.length() != tamanhoEsperado) {
                return false;
            }
        }

        return true;
    }

    public static String descriptografaAMensagem(String
mensagemCriptografada) {
        String[] colunas = mensagemCriptografada.split("\\*");
        int numeroDeLinhas = colunas[0].length();
        int numeroDeColunas = colunas.length;

        char[][] matriz = new char[numeroDeLinhas][numeroDeColunas];

        for (int coluna = 0; coluna < numeroDeColunas; coluna++) {
            for (int linha = 0; linha < numeroDeLinhas; linha++) {
                matriz[linha][coluna] =
colunas[coluna].charAt(linha);
            }
        }

        StringBuilder mensagemDescriptografada = new StringBuilder();
        for (int linha = 0; linha < numeroDeLinhas; linha++) {
            for (int coluna = 0; coluna < numeroDeColunas; coluna++)
{
                mensagemDescriptografada.append(matriz[linha]
[coluna]);
            }
        }

        return mensagemDescriptografada.toString().trim();
    }
}
```