Algoritmia e Programação

TP8

ASSUNTO – Estruturas de Dados indexadas

Objetivos Específicos:

- Mediante a apresentação dum problema, os alunos deverão ser capazes de o analisar, conceber e descrever o algoritmo estruturado em módulos e utilizar *arrays monodimensionais*.
- Desenvolver métodos de manipulação de *arrays monodimensionais*.

Conteúdo da aula

Exercício 1

Analise o seguinte programa e diga qual a sua funcionalidade:

```
public class TP8_1 {
  static Scanner sc = new Scanner(System.in);

  public static void main(String[] args) {
    int[] arr1 = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int size = sc.nextInt();
    int[] arr2 = createAndFillArray(size);
    int sum1 = sumArray(arr1);
    int sum2 = sumArray(arr2);
    printArray(arr1);
    printArray(arr2);
    System.out.println("Greatest sum :" + ((sum1 > sum2) ? sum1 : sum2));
  }
//=============================================================

  public static int[] createAndFillArray(int size) {
    int[] result = new int[size];
    for (int i = 0; i < size; i++) {
      result[i] = sc.nextInt();
    }
    return result;
  }
//=============================================================

  public static int sumArray(int[] arr) {
    int sum = 0;
    for (int i = 0; i < arr.length; i++) {
      sum += arr[i];
    }
    return sum;
  }
//=============================================================

  public static void printArray(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
      System.out.print("[" + arr[i] + "]");
    }
    System.out.println("");
  }
}
```

Input:

```
3                   (size  )
10                  (arr[0])
11                  (arr[1])
12                  (arr[2])
```

Output:

```
[1][2][3][4][5][6][7][8][9]
[10][11][12]
Greatest sum :45
```

Exercício 2

Elabore um programa modular para gerar e guardar números inteiros aleatórios (todos diferentes) pertencentes ao intervalo [5, 100]). A geração dos números termina quando surgir um número repetido ou não for possível inserir um novo número. No final, todos os números devem ser visualizados no ecrã.

O programa deve incluir os seguintes módulos:

generate	Recebe um array de inteiros e dois números inteiros que definem um intervalo fechado. Gera aleatoriamente números entre o intervalo definido por parâmetro e retorna o número de elementos armazenados no array.
search	Recebe um array de inteiros, o número de elementos armazenados no array e o número a procurar nesse vector. No caso de encontrar o número procurado, retorna a posição do respectivo elemento, senão retorna -1.
insert	Recebe um array de inteiros, o número de elementos armazenados no array e o número inteiro a armazenar na primeira posição disponível desse array. Retorna o novo número de elementos armazenados no array ou -1 se não foi possível inserir.
print	Recebe um array de inteiros e o número de elementos armazenados no array. Deve visualizar todos os elementos na mesma linha, sem espaços e entre [ ].

TP8

Uma proposta de resolução

```
public class TP8_2 {

  static final int LOWER_LIMIT = 5;
  static final int UPPER_LIMIT = 100;
  static final int MAX_ELEMENTS = UPPER_LIMIT - LOWER_LIMIT + 1;

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

  public static void main(String[] args) {
    int[] arr = new int[MAX_ELEMENTS];
    int numberOfElements = generate(arr, LOWER_LIMIT, UPPER_LIMIT);
    print(arr, numberOfElements);
  }
//=============================================================

  public static int generate(int[] arr, int lowerLimit, int upperLimit) {
    int numberOfElements = 0;
    int resultSearch = 0;
    int resultInsert = 0;
    do {
      int number = (int) (Math.random() * (upperLimit - lowerLimit) + lowerLimit);
      resultSearch = search(arr, numberOfElements, number);
      if (resultSearch == -1) {
        resultInsert = insert(arr, numberOfElements, number);
        if (resultInsert != -1) {
          numberOfElements++;
        }
      }
    } while (resultSearch == -1 && resultInsert != -1);
    return numberOfElements;
  }
//=============================================================

  public static int search(int[] arr, int numberOfElements, int number) {
    for (int pos = 0; pos < numberOfElements; pos++) {
      if (arr[pos] == number) {
        return pos;
      }
    }
    return -1;
  }
//=============================================================

  public static int insert(int[] arr, int numberOfElements, int number) {
    if (numberOfElements < arr.length) {
      arr[numberOfElements++] = number;
      return numberOfElements;
    }
    return -1;
  }
//=============================================================

  public static void print(int[] arr, int numberOfElements) {
    for (int pos = 0; pos < numberOfElements; pos++) {
      System.out.print("[" + arr[pos] + "]");
    }
    System.out.println("");
  }
}
```

TP8

Exercício 3

Pretende-se um programa para ler uma sequência de nomes de alunos de uma turma e respetivas notas da disciplina de APROG. O número de alunos é fornecido pelo utilizador. O programa deve mostrar no final os nomes dos alunos com a melhor nota.

Uma proposta de resolução

```
public class TP8_3 {
  static final int MIN_NUMBER_OF_STUDENTS = 0;
  static final int MAX_NUMBER_OF_STUDENTS = 50;
  static final float MIN_GRADE = 0.0f;
  static final float MAX_GRADE = 20.0f;
  static Scanner sc = new Scanner(System.in);

  public static void main(String[] args) {

    int numberOfStudents = readNumberBetween(MIN_NUMBER_OF_STUDENTS, MAX_NUMBER_OF_STUDENTS);
    String[] arrNames = new String[numberOfStudents];
    float[] arrGrades = new float[numberOfStudents];

    readNameAndGradeOfStudents(arrNames, arrGrades);

    float bestGrade = getBestGrade(arrGrades);

    printNamesOfStudentsWithBestGrade(bestGrade, arrGrades, arrNames);
  }
//=============================================================
  public static int readNumberBetween(int lowerLimit, int upperLimit) {
    int number = sc.nextInt();
    while (number < lowerLimit || number > upperLimit) {
      System.out.println("Invalid number! try [" + lowerLimit + "," + upperLimit + "]");
      number = sc.nextInt();
    }
    sc.nextLine();
    return number;
  }
//=============================================================
  public static float readNumberBetween(float lowerLimit, float upperLimit) {
    float number = Float.parseFloat(sc.nextLine());
    while (number < lowerLimit || number > upperLimit) {
      System.out.println("Invalid number! try [" + lowerLimit + "," + upperLimit + "]");
      number = Float.parseFloat(sc.nextLine());
    }
    return number;
  }
//=============================================================
  public static void readNameAndGradeOfStudents(String[] arrNames, float[] arrGrades) {
    System.out.println("Reading students' name and grade");
    for (int i = 0; i < arrNames.length; i++) {
      System.out.println("#" + (i + 1) + " name :");
      arrNames[i] = sc.nextLine();
      System.out.println("#" + (i + 1) + " grade:");
      arrGrades[i] = readNumberBetween(MIN_GRADE, MAX_GRADE);
    }
  }
//=============================================================
  public static float getBestGrade(float[] arrGrades) {
    float maxGrade = arrGrades[0];
    for (int i = 1; i < arrGrades.length; i++) {
      if (arrGrades[i] > maxGrade) {
        maxGrade = arrGrades[i];
      }
    }
    return maxGrade;
  }
//=============================================================
  public static void printNamesOfStudentsWithBestGrade(float grade, float arrGrades[], String[] arrNames) {
    System.out.printf("Students with best grade (%f)%n", grade);
    for (int i = 0; i < arrGrades.length; i++) {
      if (arrGrades[i] == grade) {
        System.out.println(arrNames[i]);
      }
    }
  }
}
```