Aplicação Final

Submeter no moodle

Sistema de Apoio a Votações do Parlamento (SAVOP)

Objectivos Gerais:

Analisar e Resolver Problemas Computacionalmente usando o paradigma procedimental estruturando o programa em módulos.

Codificar programas utilizando a linguagem Java na perspectiva essencialmente procedimental.

Assumir atitudes de aprendizagem activa, colaborativa e responsável, de trabalho persistente e de **aplicação de espírito crítico na análise e resolução de problemas**.

Objectivos Específicos

Mediante a especificação do problema o aluno deverá ser capaz de:

- Definir estruturas de dados adequadas (neste caso devem incluir *arrays unidimensionais e bidimensionais*);
- Utilizar ficheiros de texto como fontes de entrada de dados (leitura) e como destinos de informação (escrita);
- Analisar, conceber e descrever algoritmos estruturando-os em módulos;
- Codificar a aplicação em Java (na perspectiva procedimental) tendo em conta a utilização das normas adequadas de codificação e documentação;
- Elaborar um plano de testes adequado.

Maria da Conceição Neves

Especificação do problema

Sistema de Apoio a Votações do Parlamento (SAVOP)

- Pretende-se uma aplicação que apoie a gestão de votações dos deputados no parlamento e permita analisar cada uma das votações segundo várias perspetivas: distribuição dos tipos de votos por partido, por faixa etária, etc.
- A aplicação terá disponível a seguinte informação:
  - Um ficheiro de texto de nome **Deputados.txt** onde se encontre, em cada linha, a seguinte informação respeitante a um máximo de 230 deputados: *código identificação; nome completo; nome do partido e sigla; data de nascimento*
    O código de identificação é constituído por 5 caracteres. Os 3 primeiros caracteres são constituídos por letras que representam o círculo eleitoral e os restantes 2 por um número de ordem. Exemplo: "PRT03".
  - Existirão também vários ficheiros de texto, cada um com as informações referentes a cada votação ocorrida no parlamento. Estes ficheiros cujo nome será a designação do assunto votado, terão em cada linha a votação de cada deputado presente nessa votação (um '**S**' caso o deputado tenha votado a favor, um '**N**' caso o deputado tenha votado contra ou um '**A**' caso se tenha abstido), precedido do seu código de identificação. Exemplo de uma linha do ficheiro: "PRT03S". A ordem e a quantidade das linhas deste ficheiro pode não ser idêntica à ordem e à quantidade de linhas do ficheiro dos deputados. O nome do ficheiro tem a designação da lei votada.

Maria da Conceição Neves

Funcionalidades a disponibilizar:

1. Ler um ficheiro de texto, de nome "Deputados.txt", com a informação anteriormente indicada, respeitante aos deputados e armazene-a em memória central.
2. Visualizar toda a informação existente em memória, num determinado momento, sobre todos os deputados, usando paginação.
3. Alterar, em memória, qualquer parâmetro da informação de um deputado
4. Ler de um ficheiro de texto, com o nome que for indicado pelo utilizador, a informação referente a uma determinada votação ocorrida. Em cada linha deverá existir o código de identificação do deputado e o seu voto (por exemplo: "PRT03S"). O nome do ficheiro deverá indicar a votação a que diz respeito (por exemplo: "Lei_13_2015.txt").
5. Visualizar a informação dos deputados, ordenada alfabeticamente pelo código de identificação, em que em cada linha seja apresentado o código, o primeiro e o último nome, o seu partido e o seu voto na votação em análise, usando paginação.

Maria da Conceição Neves

Funcionalidades a disponibilizar (cont.):

6. Visualizar no ecrã os resultados da última votação introduzida e, também, guardá-los num ficheiro de texto cujo nome seja a palavra "Resultados" concatenada com o título da votação. Inicialmente será mostrado o título da votação (que será idêntico ao nome do ficheiro de texto lido com os dados da votação), seguido de uma listagem com os votos discriminados por cada partido representado no parlamento. Os partidos deverão ser visualizados por ordem decrescente do seu total de deputados. No final deverão ser visualizados os números totais de votos a favor, votos contra e abstenções. Exemplo:

Votação de: Lei_13_2015

PartidoC;	Votos a favor: 0;	Votos contra: 100;	Abstenções: 29.
PartidoD;	Votos a favor: 61;	Votos contra: 0;	Abstenções: 0.
PartidoA;	Votos a favor: 19;	Votos contra: 0;	Abstenções: 1.
PartidoB,	Votos a favor: 0;	Votos contra: 15;	Abstenções: 0.
Totais	Votos a favor: 80;	Votos contra: 115;	Abstenções: 30.

Sugestão: Deverá existir uma estrutura de dados que contenha todos os partidos representados, o respetivo número de deputados e a votação do partido. Esta estrutura deve estar ordenada por representatividade, isto é, primeiro os partidos com o maior número de deputados. Partidos com o mesmo número de deputados deverão estar ordenados alfabeticamente.

Maria da Conceição Neves

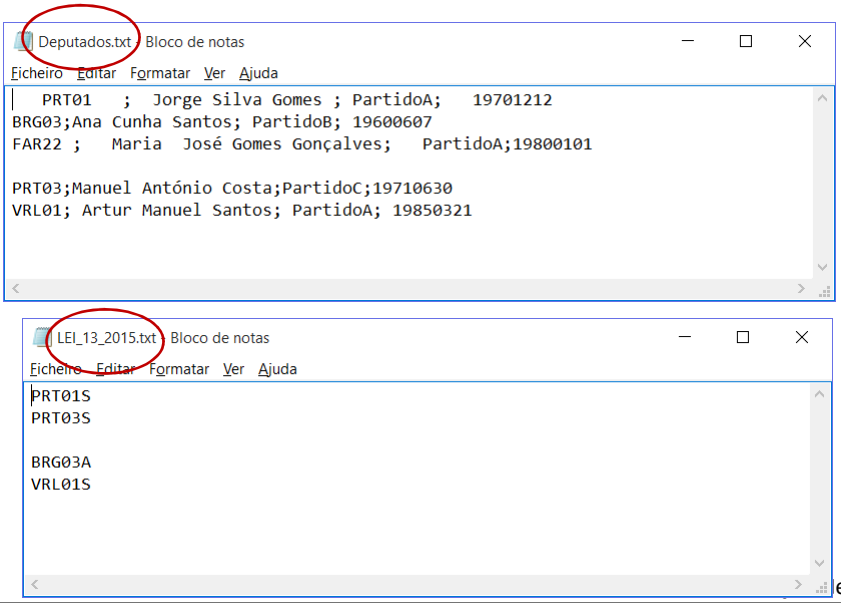
Funcionalidades a disponibilizar (cont.):

7. Visualizar para uma determinada votação os resultados obtidos em função da faixa etária. Inicialmente será mostrado o título da votação (que será idêntico ao nome do ficheiro de texto lido com os dados da votação), seguido da percentagem de votos a favor, votos contra e abstenções nas seguintes faixas etárias: menores de 35 anos; entre 35 e 60 e superiores a 60.

8. Criar uma página HTML com a informação obtida no ponto 6.

Maria da Conceição Neves

Exemplo de ficheiros disponíveis



eves

Processo de Desenvolvimento

PARTE 1: Análise dos Requisitos

- Requisitos :
  - A aplicação deve ser orientada por um menu de funcionalidades a ser apresentado ao utilizador e que em função da escolha do utilizador disponibilizará a funcionalidade selecionada.
    A aplicação só terminada quando a opção selecionada for FIM
  - Análise das funcionalidades de 1 a 8

Maria da Conceição Neves

PARTE 2: Conceção da solução

1. Estrutura de Dados principal (ED do main)

2. Algoritmo do módulo principal (main)

3. Para cada funcionalidade
   - ED local
   - Algoritmo

Maria da Conceição Neves

PARTE 3: Implementação e Teste

1. Após concebida a solução algorítmica de cada módulo passamos à fase de implementação e teste

- Usar as boas práticas de codificação:
  - Identificadores adequados
  - Evitar "valores literais" definindo constantes com esses valores e usar essas constantes
  - Evitar métodos com muitas linhas de código desdobrando-os
  - Desdobrar os métodos por várias classes em função dos seus objetivos
  - Usar comentários Javadoc

- À medida que forem sendo implementadas as funcionalidades deverão ser testadas

Maria da Conceição Neves

Estrutura de Dados (pode ser melhorada)

Assunto votado
String assuntoVotado Lei_01_2015

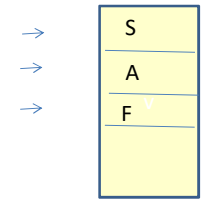
Matriz com informação dos deputados
String [ ][ ] deputados

deputados

PRT01	Jorge Silva Gomes	PartidoA	19701212
BRG03	Ana Cunha Santos	PartidoB	19600607
FAR03	Maria José Gomes	PartidoA	19800707

Vetor com votacoes
char [ ] votacoes

votacoes



int nDeputados

3

OBS: Sempre que haja a informação disponível a estrutura de dados deve estar atualizada

Maria da Conceição Neves

Criar / Utilizar outras classes como:

- Classe PaginaHtml - disponibiliza métodos (serviços) associados à construção de diferentes elementos a incluir numa página html
- Classe Utilitarios – disponibiliza métodos vários como:
  - Cálculo de idade
  - Mostrar a data de hoje no formato **05/12/2015**
  - Reduzir nome – 1ºnome e apelido
  - Ordenar matriz de Strings por coluna
- Classe LogErros – disponibiliza métodos associados à criação de um ficheiro de registo de situações de erro

Maria da Conceição Neves

Segue-se uma **proposta de uma solução parcial do problema que deve analisar com espírito crítico**.

Módulo menu

```
Definir menu() : int
   int op;
Inicio
    Escrever(“
        Ler Informação do ficheiro de texto …      1
        …                                          2
        ….                                         3
        …

        Terminar                                   0

        Qual a sua opção? “)
    Ler (op)
    Retornar op
Fim
```

Maria da Conceição Neves

```
public class SAVOP {

// Congiguração da aplicação
  private final static int MAX_DEPUTADOS = 230;
  private final static String FILE_DEPUTADOS = "Deputados.txt";
  private final static String PAGINA_HTML = "Pagina.html";
  private final static int MAX_LINHAS_PAGINA = 5;
  //……

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

    private static int menu() {
        String texto = "\nMENU:"
                + "\n Ler   …                …   1"
                + "\n                        …   2"
                //…
                + "\n TERMINAR               …   0"
                + "\nQual a sua opção?";
        System.out.printf("%n%s%n", texto);
        int op = in.nextInt();   in.nextLine();
        return op;
      }
    / / Continua o código de outros métodos
```

Maria da Conceição Neves

```
public static void main(String[] args)   throws FileNotFoundException {

        String[ ][ ] deputados= new String[MAX_DEPUTADOS][2];
        int nDeputados= 0;
        char[ ] votacoes= new char[MAX_DEPUTADOS];
        String  assuntoVotado=null;
        //…
        int op;
        do {
                op = menu();
                switch (op) {
                    case 1:
                        //
                        break;
                     case 2:
                            // ……..
                    case 0:
                        System.out.println("FIM");
                        break;
                    default:
                        System.out.println("Opção incorreta. Repita");
                        break;
                }
        } while (op != 0);
}
```

Maria da Conceição Neves

1

DEFINIR **lerInfoFicheiro(String nomeFich,**
String[][] deputados) : int

int nDeputados=0;
fIn=AbrirFichTextoParaLeitura(nomeFich)
ENQUANTO(**não fim fIn** AND **nDeputados< linhas matriz**)
aux=lerLinha(fIn);
nDeputados=**guardarDadosDeputado(aux,**
deputados, nDeputados)
FIM ENQUANTO
FecharFich(fIn)
retorna nDeputados
FIMDEFINIR

Maria da Conceição Neves

1

```
/**
 * Carrega informação dos deputados para memória a partir de ficheiro de
 * texto
 *
 * @param nomeFich - nome do ficheiro que contem info dos deputados
 * @param deputados - matriz de strings para guardar a info de deputados
 * @return o número de deputados inseridos na matriz
 * @throws FileNotFoundException
 */
private static int lerInfoFicheiro(String nomeFich, String[][] deputados)
                                                    throws  FileNotFoundException {
   Scanner fInput = new Scanner(new File(FILE_DEPUTADOS));
   int nDeputados = 0;

   while (fInput.hasNext() && nDeputados < MAX_DEPUTADOS) {
      String linha=Input.nextLine();

      // Verifica se linha não está em branco
      if(linha.length()  > 0){
            nDeputados = guardarDadosDeputado(fdeputados, nDeputados);
       }
   }
   fInput.close();
   return nDeputados;
}
```

Maria da Conceição Neves

1

```
/**
 * Acede à informação de uma linha do ficheiro e guarda na estrutura dados deputados se a linha tiver a
estrutura correta e o id com 5 carateres
 * @param linha - String com o conteúdo de uma linha do ficheiro com info de um deputado
 * @param deputados - matriz de strings com a informação dos deputados
 * @param nDeputados - número de deputados existentes na matriz deputados
 * @return  o novo número de deputados
 */
private static int guardarDadosDeputado(String linha, String[][] deputados,  int nDeputados {
  // separador de dados por linha
  String[] temp = linha.split(";");
  if (temp.length == 4) {
        String id = temp[0].trim();
        if (id.length() == 5) {
            deputados[nDeputados][0] = id;
            deputados[nDeputados][1] = temp[1].trim();
            deputados[nDeputados][2] = temp[2].trim();
            deputados[nDeputados][3] = temp[3].trim();
            nDeputados++;
       } else {
            System.out.println("Linha incorreta porque id incorreto");
       }
   } else {
          System.out.println("Linha incorreta porque id incorreto");
   }
   return nDeputados;
}
```

Só valida se o id tem 5 caracteres, não valida se são 3 letras e 2 digitos nem valida se já existe um deputado com o mesmo número

Maria da Conceição Neves

2

```
/**
 * Visualizar toda a informação dos deputados existente em memória
 paginada
 * @param vec - matriz
 * @param nEl
 */
private static void listagemPaginada(String[][] matriz, int nEl) {
   int contPaginas = 0;
   for (int i = 0; i < nEl; i++) {
      if (i % MAX_LINHAS_PAGINA == 0) {
           if(contPaginas>0){
                pausa();
           }
          contPaginas++;
          System.out.println("\nPÁGINA: " + contPaginas);
          cabecalho();
      }
      System.out.printf("%-6s||%-30s||%-10s||%-12s%n", matriz[i][0],
 matriz[i][1], matriz[i][2], matriz[i][3]);
   }
}
```


2

Esta é uma solução para a **listagem paginada,**

- Informação é organizada em páginas
- Cada página será constituída, no máximo, por MAX_LINHAS_PAGINA (constante definida na classe).
- Sempre que muda de página o programa para até o utilizador carregar na tecla ENTER
- A nova página inicia sempre com o cabeçalho.

```
private static void cabecalho() {
    System.out.printf("%-6s||%-30s||%-10s||%-12s%n", "ID", "NOME",
 "PARTIDO", "DATA NASC");
    System.out.println(
     "=========================…======================");
}

private static void pausa( ) {
    System.out.println("\n\nPara continuar digite ENTER\n");
    in.nextLine();
}
```

```
/**
   * Atualiza informação alterável de um deputado
   * @param idDeputado - identificação do deputado
   * @param deputados - matriz com toda a informação dos deputados
   * @param nDeputados - número de deputados
   * @return false se o deputado não foi encontrado ou true se foi encontrado
   * e atualizadao provavelmente atualizado
   */
  private static boolean actualizaInfoDeputado(String idDeputado, String[][] deputados, int nDeputados) {
     int pos, nc, np, ntp;
     // Pesquisa do deputado
     if ((pos = pesquisarDeputadoPorID(idDeputado, nDeputados, deputados)) > -1) {
        System.out.printf("");
        int op;
        do {
           System.out.printf("%6s-%30s-%7s-%12s%n", deputados[pos][0], deputados[pos][1], deputados[pos][2], deputados[pos][3]);
           op = menuDadosDeputado();
           switch (op) {
              case 1:
                 System.out.println("Novo nome:");
                 deputados[pos][1] = in.nextLine();
                 break;
              case 2:
                 System.out.println("Nova data:");
                 deputados[pos][3] = in.nextLine();
                 break;
              default:
                 System.out.println("Opção incorreta");
                 break;
           }
        } while (op != 0);
     } else {
        System.out.printf("O deputado %s não foi encontrado!", idDeputado);
        return false;
     }
     return true;
  }
```

3

O algoritmo desta funcionalidade foi previamente esboçado, mas não está aqui apresentado

Maria da Conceição Neves

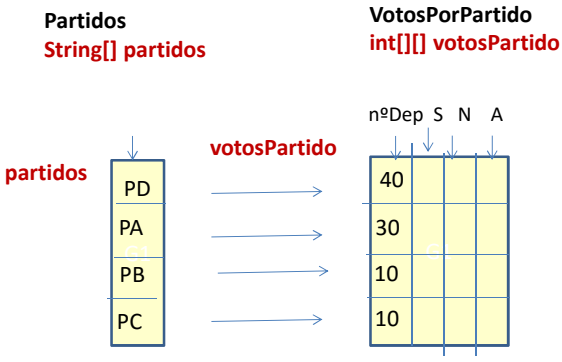
```
private static int menuDadosDeputado() {

    String texto = "ATUALIZAR"
          + "\n NOME                    ...  1"
          + "\n DATA NASCIMENTO ...  2"
          + "\n TERMINAR              ...  0"
          + "\n\nQUAL A SUA OPÇÃO?";

    System.out.printf("%n%s%n", texto);
    int op = in.nextInt();
    in.nextLine();
    return op;
  }
```

3

Maria da Conceição Neves

6

Uma sugestão para a Estrutura de Dados local associada à funcionalidade 6-)



Maria da Conceição Neves