Programando em Java

Gestor de agenda de campanha eleitoral

Mestrado Integrado em Engenharia Informática FCT UNL

<http://ctp.di.fct.unl.pt/miei/ip/>

Corpo Docente 2020/2021

António Ravara, Artur Miguel Dias, Bernardo Toninho,
Ema Vieira, Inês Fernandes, Margarida Mamede,
Miguel Monteiro, Rui Nóbrega

Gestor de agenda de campanha eleitoral

Carregamento e gravação para ficheiro

Pode ser útil gravar em ficheiro o estado da agenda da campanha ao sair da aplicação

Para isso basta alterar o método `executeOption`, no caso da opção `EXIT`

```
private static void executeOption
(Scanner in, String option, Campaign camp) {
    switch (option) {
        ...
        case EXIT: processEXIT(camp);
                    break;
        default: System.out.println("Comando inexistente");
    }
}
```

Método processEXIT

Deve (tentar) criar um ficheiro onde grava todo o “estado” da aplicação

O “estado” é o conteúdo das variáveis de instância (os atributos) do “sistema” (a entidade - classe - Campaign)

O estado contém o número de candidatos, todos os candidatos, e para cada um, o seu nome e eventos (e o seu número).

Vai-se usar de novo os iteradores desenvolvidos (de candidatos e de eventos dos candidatos)

Método processEXIT

Para trabalhar com ficheiros, é preciso lidar com FileNotFoundException

```
private static void processEXIT(Campaign camp) {
    try {
        if (camp.hasCandidates()) {
            PrintWriter out = new PrintWriter("log.txt");
            saveState(out, camp);
            out.close();
            System.out.println("Estado da aplicação gravado para o
ficheiro log.txt");
        }
    }
    catch (FileNotFoundException e) {
        System.out.println("Falhou a criação do ficheiro log.txt");
    }
    System.out.println("Terminou o uso da aplicação");
}
```

Método saveState

Primeiro escreve-se o número total de candidatos.

Depois percorre-se um iterador de candidatos - para cada candidato, escreve-se o seu nome e os eventos na sua agenda (o método printEvsCand trata disto)

```
private static void saveState(PrintWriter out, Campaign camp) {
    out.println(camp.numCandidates());
    IteratorCandidates itCands = camp.iteratorCandidates();
    while (itCands.hasNext()) {
        Candidate cand = itCands.next();
        out.println(cand.getName());
        printEvsCand(out, cand);
    }
}
```

Método printEvsCand

Primeiro escreve-se o número total de eventos do candidato.

Depois percorre-se um iterador de eventos - para cada evento, escreve-se o dia, hora e local (semelhante ao método processLEC)

```
private static void printEvsCand(PrintWriter out, Candidate cand) {
    out.println(cand.numEvents());
    IteratorEventsCand itEvents = cand.iteratorEventsCand();
    while (itEvents.hasNext()) {
        Event ev = itEvsCand.next();
        out.println(ev.getDay());
        out.println(ev.getPeriod());
        out.println(ev.getPlace());
    }
}
```

Será possível unificar este método e o processLEC? Já o fizemos antes...

Gestor de agenda de campanha eleitoral

Carregamento e gravação para ficheiro

Pode ser útil ao lançar a aplicação carregar de ficheiro o estado da agenda da campanha gravado ao sair da anterior utilização da aplicação

Para isso basta alterar o método main, colocando antes do ciclo de processamento de ficheiros o carregamento

```
public static void main(String[] args) {
    Scanner in = new Scanner(System.in); String option;
    Campaign camp = new Campaign();
    try { loadData(camp); }
    catch (FileNotFoundException e)
        { System.out.println("Erro na leitura de log.txt"); }
    do {
        printMenu(); option = in.nextLine();
        executeOption(in,option,camp);
    } while (!option.equals(EXIT));
    in.close();
}
```

Método loadData

Supõe-se que os dados estão no ficheiro `log.txt`, com o formato que usámos agora:

- primeiro está o número de candidatos
- depois, para cada candidato, o seu nome, número de eventos e dia, hora e local de cada evento

```
public static void loadData(Campaign camp) throws
FileNotFoundException {
    Scanner in = new Scanner(new FileReader("log.txt"));
    int numCands = in.nextInt();in.nextLine();
    for(int i = 0; i < numCands; i++) {
        String candName = in.nextLine();
        camp.addCandidate(candName);
        loadEvsCand(in,camp,candName);
    }
    in.close();
}
```

Método loadEvsCand

Para cada candidato, cria todos os seus eventos

```
public static void loadEvsCand
(Scanner in, Campaign camp, String cand) {
    int numEvs = in.nextInt();in.nextLine();
    for (int j = 0; j < numEvs; j++) {
        int day = in.nextInt();in.nextLine();
        int period = in.nextInt();in.nextLine();
        camp.addEvent(day,period,cand,in.nextLine());
    }
}
```