30/10/2015

Universidade de Brasília

Programação Sistemática 2/2015 – Turma F

# Relatório do Trabalho Prático 2

## Introdução

Foi decidido entregar o relatório em um arquivo de formato PDF em razão de ele ser extenso. Acreditamos também no intuito de melhorar a leitura a ser feita pelo corretor de nosso trabalho.

gPalestra é um programa gerenciador de palestras da UnB, a partir de três arquivos de texto contendo informações sobre palestrantes, palestras e localidades. Inicialmente ele foi projetado para ser programado em C, porém devido a complicações por causa do tempo, decidimos refazê-lo em Java. Este relatório tem por seu objetivo detalhar a estrutura do projeto e prestar contas a respeito do processo envolvido até o produto final.

## Integrantes

No projeto, temos 4 membros. Cristiano Cardoso (15/0058349), Marcos Moura Gonçalves (15/0157479), Marcia Andréa Faria (13/0069671) e Lucas Lintomen (?). Este último não nos respondeu o e-mail e não nos procurou para contribuir para nosso trabalho. Foi nos relatado que ele tenha algum tipo de problema para trabalhar em grupo, mas a decisão de não participar foi dele.

A seguir, temos a tabela (também disponível nos arquivos de texto) de contribuição de cada participante do grupo.

Tabela 1.0

|  |  |  |  |
| --- | --- | --- | --- |
| Cristiano Cardoso | | | |
| **Data** | **Horas trabalhadas** | **Tipo de Tarefa** | **Descrição de Tarefa Realizada** |
| Oct 27 | 3 | Estudar Especificação | Tranquila o suficiente para fazê-la sem consultar manuais, porém especificação um pouco vaga a ponto de necessitar reuniões do grupo. |
| Oct 27-30 | 4 | Adaptação e codificação do projeto. | Realizar a adaptação da Versão 1.0 para a nova estrutura de componentes.  Codificar a base para módulo Localidades |
| 31 | 4 | Estudar especificação e conceitos de novas ferramentas;  Implementar novas ferramentas ao código. | Adicionar comentários no estilo *JavaDoc*;  Estudar o funcionamento do JavaDoc, JUnit e FindBugs.  Adicionar esqueleto do JUnit para o programa (componente teste).  Testar o módulo Localidades. |
| Marcos Moura | | | |
| **Data** | **Horas trabalhadas** | **Tipo de Tarefa** | **Descrição de Tarefa Realizada** |
| Oct 27 | 6 | Estudar Especificação  Revisar Código | Estudada a especificação do TP2, visto os diferenciais para o TP1;  Revisão do código feito pelos integrantes para encontrar falhas |
| Márcia Andréa | | | |
| **Data** | **Horas trabalhadas** | **Tipo de Tarefa** | **Descrição de Tarefa Realizada** |
| Oct 25 | 7 | Estudar Especificação  Estudar conceitos de JavaDoc e JUnit | Adicionado comentários JavaDoc e testes do JUnit. |

# Especificações Internas do Projeto

O programa foi inicialmente (TP1) projetado para ser programado em C. Como o tempo de todos os integrantes ativos do grupo é muito curto, decidimos mudar drasticamente para Java com finalidade de terminar o projeto a tempo. Isso porque esta linguagem de programação não requer manipulações de baixo nível e escrita de módulos referentes a escrita de arquivo, manipulação de estruturas de dados como *strings*, *vetor* ou *tabela* hashing.

A versão 2.0 trouxe as seguintes mudanças:

* Retirada da versão em C
* Introduzida uma nova estrutura de componentes, pois os módulos cresceram bastante a ponto de se tornarem super-módulos.
* Introdução do módulo localidade
* Implementação do módulo de testes estático (FindBugs) e dinâmico (JUnit)
* Documentação automática do código (JavaDoc)
* Reorganização dos arquivos

A seguir, temos a hierarquia do programa e sua descrição

Tabela 2.0

|  |  |
| --- | --- |
| **Nome** | **Descrição** |
| build.xml | Arquivo ANT Script para gerar o executável JAR |
| .settings  .classpath | Preferências da IDE Eclipse |
| /bin | Arquivos compilados | |
| /doc | Documentação gerado pelo JavaDoc | |
| /src | Arquivos código-fonte | |
| /src/módulos | Módulos menores | |
| /src/palestrante  /src/palestra  /src/localidades | Componentes principais do programa contendo rotinas de leitura, tratamento de dados e persistência de cada uma dessas entidades-chave. | |
| /src/teste | Módulo de teste dinâmico JUnit | |
| /src/ps\_trab1 | Rotina principal do programa | |

## Manual do gPalestra

*Presente também no “LEIAME.txt”*

1. Como compilar

*Não se esqueça de:*

a)possuir o JDK 8. Para instalar utilize os seguintes comandos:

|  |
| --- |
| sudo add-apt-repository ppa:webupd8team/java  sudo apt-get update  sudo apt-get install oracle-java8-installer |

b) possuir a IDE Eclipse Mars 1. Faça o download do instalador em:

|  |
| --- |
| http://www.eclipse.org/downloads/index.php?show\_instructions=TRUE |

c) importar o projeto no Eclipse:

|  |
| --- |
| * Abra o Eclipse * Vá em File > Import > General > Archive File * Selecione o .zip que foi baixado do Moodle   Ele deve importar todas as preferências do projeto, caso não o façacontate o responsável pelo envio.  Obs: Geralmente a dependência de nosso projeto, JUnit 4, já vem instalada no Eclipse. |

*Agora, com o projeto devidamente importado, você está pronto para compila-lo na IDE.*

|  |
| --- |
| * Vá em File > Export > Runnable JAR file * Em "Launch configuration" você poderá escolher o programa para rodar no:   + Modo normal (Principal - gPalestra)   + Modo JUnit (test\_runner - gPalestra) |

*Clique em Finish e tudo estará certo*

1. Como Usar

*Nosso arquivo comprimido já com o com o .jar se houver algum inconveniente na hora de compilar.*

a) Abra o terminal

b) Vá até a pasta onde você importou o .zip com o Eclipse usando o comando "cd"

c) Entre com o comando:

|  |
| --- |
| java -jar gPalestra.jar |

Obs: Se no terminal aparecer algumas letras estranhas, é porque você está executando este programa no Windows, provavelmente. Ele foi desenvolvido em Unix, então peçamos que o use de preferência. Até porque eu (desenvolvedor) já enfrentei alguns problemas com o mesmo enquanto programava, porém não tenho nada contra - só inexperiência.

*O programa irá ler do arquivo Palestrantes.txt, Palestras.txt e Localidades.txt na raiz do a fim de gerar um arquivo saída ao usuário contendo uma lista, em ordem cronológica, das palestras confirmadas com sucesso.*

*Caso alguma palestra não possa ser alocada, o programa automaticamente interrompe o processamento e envia uma exceção (erro) detalhando qual palestra não pôde ser alocada.*

1. Usando o FindBugs

*O Eclipse é uma ferramenta bastante poderosa. É simples usa-la pra praticamente qualquer coisa. Para instalar o FindBugs e gerar um laudo proceda com o seguinte:*

|  |
| --- |
| * Vá em Help > Install New Software * *Coloque o link a seguir no campo “Work with: “:* [*http://findbugs.cs.umd.edu/eclipse/*](http://findbugs.cs.umd.edu/eclipse/) * *Siga o restante dos passos (é bastante fácil). No final o programa irá pedir para reiniciar.* |

1. Usando o JUnit

*Após o programa ter sido compilado abra o terminal e o rode usando o seguinte comando:*

|  |
| --- |
| java -cp gPalestra.jar teste.test\_runner |

1. Versões

|  |  |
| --- | --- |
| 0.1 | Primeira versão do gPalesta! Implementa os módulos básicos de leitura, tratamento e persistência de dados referentes a palestrantes e palestra. Versão em C cancelada (vide relatório em pdf), porém com implementação de módulo de testes CUnit para funcionalidade de leitura e tratamento de dados. E versão Java com todos os módulos implementados (programa funcional), todavia sem módulo JUnit. |
| 0.2 | Nesta implementamos testes básicos do JUnit, documentação em JavaDoc, rodamos o FindBugs para coletar erros estáticos no projeto (laudo no .zip) e incluímos o componente localidade, bem como reformulamos toda a estrutura do mesmo. |

1. Informações Extras

*Considera-se que o 'TP2' de Programação Sistemática foi um sucesso.*