TRABALHO 1 - PROGRAMAÇÃO ORIENTADA A OBJETOS

Fernando Albuquerque
fernandoalbuquerque@yahoo.com.br

1. INTRODUÇÃO

- Desenvolva o sistema de software seguindo os requisitos especificados.
- Atente para as instruções quanto à arquitetura e as responsabilidades dos módulos.
- Deve ser usada a linguagem *C++*.
- Devem ser usadas as ferramentas: *Dev-C++*, *Doxygen*, *CppUnit* e *gcov*.
- Os documentos devem ser preenchidos com clareza e cuidados devem ser tomados com a ortografia.
- Adote uma convenção de codificação (http://www.doc.ic.ac.uk/lab/cplus/c++.rules/).
- Forneça os artefatos produzidos em um CD adequadamente organizado (diretórios).
- Na raiz do CD deve haver um arquivo LEIAME.PDF onde é explicado o objetivo de cada diretório.
- Identifique o CD com as matrículas dos membros do grupo.
- Textos e diagramas devem ser fornecidos no formato PDF.

1.1. REQUISITOS FUNCIONAIS

O sistema de software a ser desenvolvido visa facilitar o uso do PSP (Personal Software Process) nos projetos realizados por alunos. O sistema possibilitará o acesso a dados variados acerca de tais projetos. Para cada projeto realizado, serão armazenados os seguintes dados: matrícula do aluno, código identificador do projeto, datas (início e término) e nota. Cada projeto poderá ser composto por módulos. Para cada módulo, serão armazenados os seguintes dados: código identificador do módulo, nome do arquivo com o módulo e tamanho do módulo (número de linhas de código). O desenvolvimento de cada módulo será dividido nas seguintes fases: planejamento, projeto, codificação, compilação, teste e avaliação. Para cada fase, de cada módulo, serão armazenados os seguintes dados: tempo estimado (minutos) e tempo efetivamente gasto (minutos). A partir de uma matrícula e um código identificador de projeto, deverá ser possível obter as métricas: número de linhas de código no projeto do aluno, tempo gasto (minutos) pelo aluno para a realização do projeto, tempo gasto (minutos) pelo aluno em cada módulo do projeto, produtividade média do aluno no projeto (número de linhas de código por hora), produtividade do aluno em cada módulo do projeto (número de linhas de código por hora). A partir do código identificador de um projeto, deverá ser possível obter as seguintes métricas: nota média dos alunos, tamanho médio (número de linhas de código), produtividade média (número de linhas de código por hora). As seguintes regras devem ser observadas: cada aluno pode participar de até cinco projetos e cada projeto pode conter até 10 módulos.

1.2. DOMÍNIOS

CÓDIGO DE PROJETO	NÚMERO DECIMAL COMPOSTO POR 5 DÍGITOS
CÓDIGO DE MÓDULO	NÚMERO DECIMAL COMPOSTO POR 5 DÍGITOS
CÓDIGO DE FASE	NÚMERO DECIMAL COMPOSTO POR 1 DÍGITO
DATA	DD/MM/AAAA
MATRÍCULA	NÚMERO DECIMAL COMPOSTO POR 5 DÍGITOS
NOME DE ARQUIVO	10 CARACTERES QUAISQUER
NOTA	VALOR DE 0 A 10
TAMANHO EM LINHAS DE CÓDIGO	VALOR DE 0 A 1000
TEMPO	VALOR DE 0 A 2000

- As fases têm os códigos: planejamento (1), projeto (2), codificação (3), compilação (4), teste (5) e *postmortem*(6).

1.3. REQUISITOS NÃO FUNCIONAIS

- O sistema deve ser organizado em camadas de apresentação, negócio e persistência.
- A camada de apresentação é responsável pela interface com o usuário e pela validação de domínios.
- A camada de negócio deve possibilitar o cálculo das métricas.
- Os dados devem ser armazenados em um banco de dados relacional.
- O sistema gerenciador de banco de dados deve ser o *SQLite*.
- A camada de persistência deve ser responsável pela conexão ao banco de dados e execução de comandos SQL.

- O sistema deve ser composto por um programa decomposto em módulos.
- Os módulos devem ser decompostos em classes.
- Cada módulo da interface com o usuário deve ter uma classe controladora de interação.
- Cada módulo da lógica de negócio deve ter uma classe controladora de negócio.

2. QUESTÕES

Resolva as seguintes questões atentando para o que é pedido.

QUESTÃO 1 – 2,5 PONTOS

- Projete o sistema no nível de módulos de software.
- Distribua os módulos entre as camadas de apresentação, negócio e persistência.
- Documente a responsabilidade de cada módulo via cartões CRC.
- Documente as interfaces (protótipos das funções e descrições) entre os módulos.
- Crie um diagrama com os relacionamentos entre os módulos.
- Considere que o módulo de persistência recebe instâncias de classes *Command* para execução.

QUESTÃO 2 – 2,5 PONTOS

- Crie uma unidade para conter as classes que representam os domínios.
- Identifique e codifique as classes que representam os domínios.
- Comente as classes e produza a documentação com o *Doxygen*.
- Cada classe deve ser responsável pelo armazenamento dos atributos necessários.
- Cada classe deve conter código para validar o domínio que a classe representa.
- Em caso de falha na validação, deve ser lançada uma exceção.
- Use uma classe existente para a exceção, ou codifique uma classe que considere apropriada.

QUESTÃO 3 – 2,5 PONTOS

- Crie uma unidade para conter as classes que representam as entidades.
- Identifique e codifique as classes que representam as entidades.
- Comente as classes e produza a documentação com o *Doxygen*.
- Cada classe deve conter atributos que sejam instâncias das classes que modelam os domínios.
- As classes que modelam as entidades não são responsáveis por validar os formatos dos seus atributos.

QUESTÃO 4 – 2,5 PONTOS

- Codifique um teste de unidade para cada tipo básico usando as ferramentas *CppUnit* e *gcov*.
- Cada teste de unidade deve atingir, no mínimo, a cobertura de 90% das linhas do código sendo testadas.
- Comente as classes e produza a documentação com o *Doxygen*.
- Forneça cópias dos relatórios de cobertura produzidos pela ferramenta *gcov*.