TRABALHO PARA A DISCIPLINA DE TÉCNICAS DE PROGRAMAÇÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DA UTFPR: *AQUI NOME DO JOGO – MODELO & ESPECIFICAÇÃO DO TRABALHO*

Primeiro Autor, Segundo Autor
primeiro.autor@utfpr.edu.br, segundo.autor@utfpr.edu.br

Disciplina: **Técnicas de Programação – CSE20** / S1? – Prof. Dr. Jean M. Simão
Departamento Acadêmico de Informática – DAINF - Campus de Curitiba
Curso Bacharelado em: Engenharia da Computação / Sistemas de Informação
Universidade Tecnológica Federal do Paraná - UTFPR
Avenida Sete de Setembro, 3165 - Curitiba/PR, Brasil - CEP 80230-901

Resumo - *Este documento apresenta um modelo para o texto do trabalho de Técnicas de Programação, além de instruções/especificações para o trabalho ele mesmo e detalhes sobre sua avaliação. Quanto ao resumo em si, ele deve trazer uma visão geral do trabalho. Mais precisamente, o resumo deve contemplar sucintamente a motivação e o contexto do trabalho, o seu objeto de estudo (um jogo de plataforma), o seu processo de desenvolvimento e os resultados obtidos. Um exemplo de resumo seria*: A disciplina de Técnicas de Programação exige o desenvolvimento de um *software* de plataforma, no formato de um jogo, para fins de aprendizado de técnicas de engenharia de *software*, particularmente de programação orientada a objetos em C++. Para tal, neste trabalho, escolheu-se o jogo Brasileirinho++, no qual o jogador enfrenta inimigos em um dado cenário. O jogo tem três fases que se diferenciam por dificuldades para o jogador. Para o desenvolvimento do jogo foram considerados os requisitos textualmente propostos e elaborado modelagem (análise e projeto) via Diagrama de Classes em Linguagem de Modelagem Unificada (*Unified Modeling Language* - *UML*) usando como base um diagrama genérico e prévio proposto. Subsequentemente, em linguagem de programação C++, realizou-se o desenvolvimento que contemplou os conceitos usuais de Orientação a Objetos como Classe, Objeto e Relacionamento, bem como alguns conceitos avançados como Classe Abstrata, Polimorfismo, Gabaritos, Persistências de Objetos por Arquivos, Sobrecarga de Operadores e Biblioteca Padrão de Gabaritos (*Standard Template Library* - *STL*). Depois da implementação, os testes e uso do jogo feitos pelos próprios desenvolvedores demonstraram sua funcionalidade conforme os requisitos e a modelagem elaborada. Por fim, salienta-se que o desenvolvimento em questão permitiu cumprir o objetivo de aprendizado visado.

Palavras-chave ou Expressões-chave (máximo quatro, não excedendo três linhas)**:** Artigo-Relatório Modelo para o Trabalho em Fundamentos de Programação 2, Trabalho Acadêmico Voltado a Implementação em C++, Normas Internas para Elaboração de Trabalho, Exemplo de Conteúdos de Trabalho de Fundamentos de Prog. 2.

Abstract - *This document presents a model for the manuscript to the academic work of Programming Techniques (Técnicas de Programação) as well as it presents general instructions/specifications about this academic work and details about its evaluation process. With respect to the abstract contents, it must provide a general explanation about the work. Precisely, the abstract must shortly present the work motivation and context, its study object (a platform game), its development process, and the obtained results. An instance of abstract would be*:

• • •

Key-words or Key-expressions (maximum four, not exceeding three lines)**:** *Paper Model to the Academic Work of Programming Course, Academic Work Related to C++ Implementation, Internal Rules for Work Elaboration, Examples of Elements for the Work of a Programming Course.*

INTRODUÇÃO

Este documento apresenta um modelo para o texto do trabalho de Técnicas de Programação no qual se estabelece como idioma oficial o português. Na verdade, este modelo é baseado em um dado modelo de artigos de Anais do Seminário de Iniciação Científica e Tecnológica da UTFPR. Em todo caso, este presente modelo mostra a configuração básica do trabalho e do texto, bem como os detalhes sobre o uso de figuras, tabelas, equações e

referências. Ademais e sobretudo, este modelo traz instruções/especificações para o trabalho ele mesmo, bem como da forma de sua avaliação pelo professor. Isto tudo considerado, passa-se para as demais explicações.

Conforme dito em classe e especificado no plano de aulas, cinquenta por cento (50%) da nota da disciplina advém de um trabalho (excetuando o exame de recuperação). A parte prática deste trabalho consiste primeiramente em compreender os requisitos definidos textualmente, modelar (analisar e projetar) o *software* visado, o qual deve ser um jogo em estilo de plataforma. A necessária modelagem deve utilizar Diagrama de Classes em Linguagem Unificada de Modelagem - *Unified Modeling Language* (*UML*). Esta parte prática consiste, particularmente e principalmente, em desenvolver/implementar este *software* de jogo em linguagem C++ respeitando os princípios da orientação objetos doutrinados em classe, salientando aqui coesão e desacoplamento.

Este trabalho é proposto visando principalmente ampliar a aplicação dos conceitos aprendidos em classe ou mesmo eventualmente a aplicação de novos conceitos aprendidos extraclasse. Assim sendo, o jogo escolhido para ser implementado deve ter complexidade tal que permita utilizar diversos recursos da linguagem, sobretudo os ensinados em classe. Portanto, deve-se conversar com o Professor da disciplina para verificar se o jogo escolhido se faz apropriado (caso alguém ainda não o tenha feito...). Não obstante, no decorrer deste documento são apresentados os requisitos solicitados para cada jogo.

Uma vez escolhido e implementado o jogo, que deve ser um estilo plataforma, este será expresso em um documento escrito. O documento será entregue no dia da apresentação do trabalho ou antes, conforme combinado em classe e especificado no planejamento da disciplina. Tanto a apresentação do desenvolvimento (levantamento de requisitos, modelagem e implementação) do jogo, quanto o documento escrito e sua apresentação serão avaliados, permitindo compor uma nota para o trabalho. Bem entendido que o desenvolvimento será avaliado inclusive por meio do acompanhamento do desenvolvimento que se dá por interações/reuniões para com o Professor, as quais devem ser solicitadas pelos discentes (i.e., alunos).

Quanto ao trabalho escrito, este deve conter um conjunto de elementos segundo um modelo dado, o qual é detalhado nas seções subsequentes deste presente documento. Justamente esse modelo tem por finalidade inclusive padronizar o trabalho escrito a ser apresentado na disciplina de Técnicas de Programação do DAINF/UTFPR lecionada pelo Prof. J. M. Simão. Os trabalhos apresentados que não sigam o padrão aqui apresentado poderão, a critério do Professor, ser penalizados e (no limite) até rejeitados. Idem para trabalhos não escritos corretamente.

Os trabalhos **não** poderão ser entregues ao Professor de maneira impressa (nem mesmo se usar frente e verso). Assim sendo, apenas a versão digital será aceita. Na verdade, é necessário enviar o trabalho escrito em formato digital ***.doc(x)*** **e** ***.pdf*** para o *e-mail* do Professor (*jeansimao* ‘arroba’ *utfpr.edu.br*). Também é necessário enviar as implementações respectivas, diagrama(s) de projeto e demais materiais de suporte, como apresentação .ppt (e respectivos pdf) usados para apresentar o trabalho em classe. Para tal, podem-se utilizar sítios seguros de compartilhamento de arquivos, preferencialmente o google-drive institucional.

Quanto à introdução em si do trabalho, mais precisamente, ela deve apresentar quatro parágrafos (cada qual com algo como quatro frases) contendo:

(1) em que contexto (*i.e.*, disciplina de Técnicas de Programação) este trabalho se dá e qual é o objetivo de tal realização;

(2) qual é o objeto de estudo e da implementação do trabalho (*i.e.*, jogo de plataforma previamente acordado com o Professor);

(3) o método utilizado que em suma é ciclo de Engenharia de *Software* de forma simplificada, *i.e.*, compreensão dos requisitos, modelagem (análise e projeto) via

diagrama(s) de classes em *UML*, implementação em C++ orientado a objetos e testes pelo uso do *software*; e

(4) introdução às seções subsequentes.

Uma vez explicado o necessário à introdução em si, este presente documento-modelo de artigo-relatório apresenta demais seções necessárias (mas não limitantes) que o trabalho deve conter, bem como seus conteúdos. Estas indicações de conteúdos mostram o que se faz necessário contemplar em cada seção, além de explicar alguns itens de formatação de elementos contemplados.

EXPLICAÇÃO DO JOGO EM SI

Nesta seção se deve discorrer a explicação do jogo em si. **Portanto, salienta-se que esta seção NÃO é (em absoluto) para explicar a modelagem (análise e projeto) ou a implementação do jogo.**

Isto dito, esta seção assim como as demais, devem seguir as regras de formatação dadas as quais foram seguidas para compor este próprio presente modelo. Justamente, quanto à formatação, o texto do trabalho deve seguir as seguintes regras:

- O trabalho deve ser totalmente digitado em fonte Times New Roman. Esta diretriz inclui, portanto, o título do trabalho, autores, filiação e endereços, títulos de seções e legendas de figuras e tabelas, além do texto normal do trabalho. O texto deve ser digitado com alinhamento 'justificado' ou ajustado.
- O trabalho completo, incluindo figuras e tabelas, deve ser limitado a doze (12) páginas (no máximo) em papel de tamanho padrão A4 (21 cm x 29,7 cm). Não reduzir figuras e tabelas a tamanhos que sacrifiquem o entendimento dos seus conteúdos.
- Cada página, no tamanho A4, deve ser formatada de modo a apresentar 2,5 cm de margem em todos os lados do documento. Dentro desta área o texto deve ser formatado em uma única coluna, **sem** incluir moldura no texto**.**
- O título deve ser digitado em **destaque**, em letras maiúsculas, centralizado e em tamanho 14 pt, não excedendo três linhas, seguido de uma linha em branco (12 pt) e pelas linhas que conterão o(s) nome(s) do(s) autor(es), em tamanho 12 pt. Em seguida, deverá vir a filiação e o(s) endereço(s) para correspondência do(s) autor(es) (tamanho 10 pt) separada por uma linha em branco. Deve-se deixar 3 linhas de espaço antes do resumo e uma linha entre os itens subsequentes (palavras-chave, *abstract* e *Key-words*).
- Digitar o título **Resumo** em **destaque**, alinhado à esquerda, tamanho 10 pt, seguido de um traço. Sem trocar de linha, digitar o resumo, em tamanho 10 pt com alinhamento justificado. Pular uma linha e digite o título **Palavras-chave:** em **destaque**, alinhado à esquerda, tamanho 10 pt. Digitar então no máximo quatro (4) palavras-chave, separadas por vírgulas, com somente a primeira letra de cada palavra-chave em maiúscula. Na sequência devem vir o ***Abstract*** e ***Key-words*****:** em inglês, seguindo o mesmo padrão de formatação do resumo e das palavras-chave. Tanto o resumo quanto o *abstract* devem conter no máximo 200 palavras.
- A seguir, separado por 2 linhas (12 pt), o texto deve ser iniciado pela **Introdução**. Os títulos das seções (Introdução etc.) devem ser escritos em **destaque**, sem numeração, em maiúsculo e alinhados à esquerda, sendo que o conteúdo, propriamente dito, deve ser iniciado após espaçamento de uma linha e tabulação (1 cm).
- Ao final de cada seção deve-se deixar uma linha vazia. Todo o texto deverá ser escrito em espaço simples. Para as eventuais subseções, somente a primeira letra do subtítulo deve ser maiúscula, sendo todas em **destaque**, sem numeração, com o título alinhado à esquerda. Uma vez escrito o subtítulo, pular uma linha. Após esta linha (vazia), iniciar o texto da subseção.

- As ilustrações e gráficos podem ser em escalas cinza ou coloridos, mas sempre centralizados. As notas de rodapé[1] devem ser colocadas na parte inferior da página correspondente separadas por um traço conforme modelo. Usar o tamanho de 8 pt.
- As referências bibliográficas devem ser listadas no fim do artigo, na ordem de citação, conforme formato da Associação Brasileira de Normas Técnicas (ABNT) ou de forma organizada enfim. No texto, as citações devem ser referenciadas por seu número colocado entre colchetes, por exemplo, [1] e [2].

Em tempo, para melhor explicar o Jogo em Si, aconselha-se utilizar de recursos como gráficos, telas e figuras do próprio jogo. **Na verdade, telas e figuras do jogo DEVEM ser usadas, inclusive para a explicação do jogo em si com as devidas imagens do jogo executando. Isto facilita por demais a compreensão do jogo contemplado no trabalho.**

A propósito, as figuras, tabelas etc., devidamente referenciadas (citadas explicitamente) no texto, podem ser colocadas da maneira mais conveniente para o autor em uma ou duas colunas, desde que o texto permaneça em apenas uma coluna. Antes e após os elementos não textuais e suas respectivas legendas, deve-se deixar uma linha de espaçamento.

Os autores não devem se esquecer da colocação de legendas nas figuras, tabelas e outros elementos gráficos. As figuras devem ser numeradas sequencialmente com algarismos arábicos conforme o exemplo da figura 1. Outrossim, as figuras devem ser apropriadamente intercaladas pelo texto e explicitamente citados nele.

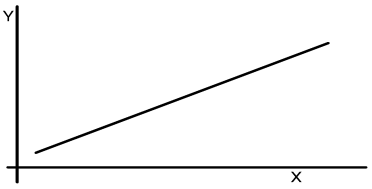

Figura 1. Centralizada na coluna e com legenda abaixo da figura.

Aproveitando o ensejo, talvez os autores façam uso de equações em alguma parte do texto. Neste âmbito, todas as equações deverão ser tabuladas a 1 cm da margem esquerda e numeradas sequencialmente, com os números entre parênteses, conforme o exemplo abaixo:

$$e(t) = \sum_{n=1}^{5} \frac{1}{2+n} \cos\left(2\pi n t\right) \tag{1}$$

As equações devem ser referenciadas no texto da seguinte forma: "Substituindo a equação (1) na equação (3), obtém-se ..."

DESENVOLVIMENTO DO JOGO NA VERSÃO ORIENTADA A OBJETOS

Nesta seção se deve discorrer a explicação do desenvolvimento do jogo/*software* utilizando orientação a objeto**, começando pelos requisitos e avançando para a modelagem, salientado o projeto em diagrama(s) de classes em UML,** e culminando na programação em C++. A explicação deve ser feita de maneira tal a **não** ser um relatório técnico repleto de detalhes, mas que seja capaz de sintetizar e valorizar os recursos técnicos utilizados (*i.e.*, sucinto e suficiente).

[1] Exemplo de nota de rodapé – Apenas os diagramas em UML que devem ser impressos na entrega do trabalho ao professor da disciplina na data acordada, impressão esta preferencialmente em papel reciclado.

Nesta explicação, deve-se primeiramente ter texto introdutório que leva a listar textualmente em tabela os **requisitos funcionais** definidos para o jogo/*software* em questão, os quais são justamente os requisitos definidos na Tabela 1 neste presente documento. Estes requisitos não podem ser alterados, mas eventualmente interpretados em reuniões com o professor, **reuniões estas obrigatórias**. Isto disto, os requisitos devem estar enquadrados em uma tabela de duas colunas na qual a primeira coluna traz os requisitos e a segunda coluna a sua situação (*status*) que pode ser 'realizado', 'semi-realizado', 'abandonado' etc.

Ainda, quando o requisito estiver como 'realizado', 'semi-realizado' ou similares faz-se absolutamente necessário indicar sucintamente quais classes ou conjuntos de classes (*e.g.*, via pacotes) que realizaram cada requisito no preenchimento tabela, no tocante ao campo 'Implementação' da Tabela 1. Eventualmente, pode-se também nomear objetos que se julguem pertinentes, se não for suficiente apenas nomear suas classes. **A Tabela 1 exemplifica o exposto definindo, sobretudo, os requisitos mínimos que cada jogo deve ter**. Em tempo, *à priori*, cada requisito será contabilizado no âmbito de avaliação apenas quando estiver completamente 'realizado'.

Tabela 1. Lista de Requisitos do Jogo e exemplos de Situações.

N.	Requisitos Funcionais	Situação	Implementação
1	Apresentar graficamente menu de opções aos usuários do Jogo, no qual pode se escolher fases, ver colocação (*ranking*) de jogadores e demais opções pertinentes.	Requisito previsto inicialmente e realizado.	Requisito cumprido via classe Menu e seu respectivo objeto, com suporte da SFML.
2	Permitir um ou dois jogadores com representação gráfica aos usuários do Jogo, sendo que no último caso seria para que os dois joguem de maneira concomitante.	Requisito previsto inicialmente, mas realizado apenas **PARCIALMENTE** – faltou ainda o segundo jogador.	Requisito cumprido inclusive via classe Jogador cujos objetos são agregados em jogo, podendo ser apenas um jogador, entretanto.
3	Disponibilizar ao menos duas fases que podem ser jogadas sequencialmente ou selecionadas, via menu, nas quais jogadores tentam neutralizar inimigos por meio de algum artifício e vice-versa.	Requisito previsto inicialmente e realizado.	Requisitos foi realizado completamente porque a classe XYZ, no pacote W, sendo que permitem . . .
4	Ter pelo menos três tipos distintos de inimigos, cada qual com sua representação gráfica, sendo que ao menos um dos inimigos deve ser capaz de lançar projetil contra o(s) jogador(es) e um dos inimigos dever ser um 'Chefão'.	Requisito previsto inicialmente, mas realizado apenas **PARCIALMENTE**.	Tal requisito foi realizado PARCIALMENTE como se observa no pacote Personagens, sendo que na hierarquia de personagens há apenas dois tipos de inimigos.
5	Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias e sendo pelo menos 3 instâncias por tipo.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
6	Ter três tipos de obstáculos, cada qual com sua representação	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .

	gráfica, sendo que ao menos um causa dano em jogador se colidirem.		
7	Ter em cada fase ao menos dois tipos de obstáculos com número aleatório de instâncias (*i.e.*, objetos), sendo pelo menos 3 instâncias por tipo.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
8	Ter em cada fase um cenário de jogo constituído por obstáculos, sendo que parte deles seriam plataformas ou similares, sobre as quais pode haver inimigos e podem subir jogadores.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
9	Gerenciar colisões entre jogador para com inimigos e seus projeteis, bem como entre jogador para com obstáculos. Ainda, todos eles devem sofrer o efeito da gravidade no âmbito deste jogo de plataforma vertical e 2D.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
10	Permitir: (1) salvar nome do usuário, manter/salvar pontuação do jogador (incrementada via neutralização de inimigos) controlado pelo usuário e gerar lista de pontuação (*ranking*). **E** (2) Pausar e **Salvar** Jogada.	Requisito previsto e **NÃO** realizado.	Requisito NÃO realizado.
Total de requisitos funcionais apropriadamente realizados. *(Cada tópico vale 10%, sendo que para ser contabilizado deve estar realizado efetivamente e não parcialmente)*			**80%** (oitenta por cento).

Isto feito, após a tabela de requisitos, a explicação textual objetiva do desenvolvimento segue, devendo-se para tal:

- Utilizar Diagrama(s) de Classes em *UML* para explicar os pacotes com as classes e suas relações, que DEVEM atender aos requisitos.
- Utilizar como base o Diagrama de Classes proposto na FIGURA 2, completando-o, melhorando-o, expandindo-o etc[2].
- A luz do(s) diagrama(s), explicar o desenvolvimento de maneira sucinta e suficiente no texto. Assim, em poucos parágrafos deve se explicar as principais classes, à luz do agrupamento de classes em pacotes e como elas se inter-relacionam.
- Em tempo, valorizar as 'sofisticações' que tenham sido realizadas, como comportamento mais elaborado de inimigos.
- Valorizar a interdisciplinaridade como a aplicação de conceitos de física e matemática aprendidos em disciplinas do ensino médio e preferencialmente em disciplinas da graduação.
- Deixar no diagrama apenas o que de fato foi implementado em C++ ~~ou, ao menos, bem indicar o que foi efetivamente implementado~~.

Para a implementação em C++ orientado a objetos (OO), que é a expressão técnica do projeto em diagrama(s) de classes em UML, deve-se usar uma biblioteca gráfica (*e.g.*, ***SFML***,

[2] Eventualmente, pode-se utilizar outros diagramas UML caso os conheça ou os tenha estudado.

Allegro, *TCL/TK*, *OpenGL* ou outra, mas preferencialmente uma OO), pois isto valoriza o trabalho esteticamente além de demonstrar a capacidade de uso de bibliotecas e suas interfaces. Não deixar de valorizar esta atividade de uso da biblioteca gráfica nesta seção. Em tempo, no site da disciplina há exemplos pedagógicos prontos com *SFML (OO em si)* e *Allegro* 4.x (procedimental em si), este tanto em programa procedimental quanto integrado em programa orientado a objetos.

Esta seção em questão é muito importante e fundamental no trabalho, sendo que será corrigida com muita atenção pelo professor. Pede-se, por fim, que todos os autores revisem cuidadosamente a versão final do trabalho (como um todo) para evitar erros de português, digitação e/ou formatação. Na verdade, além disto, uma equipe poderia revisar o trabalho escrito da outra e vice-versa para fins de aprimoramento mútuo.

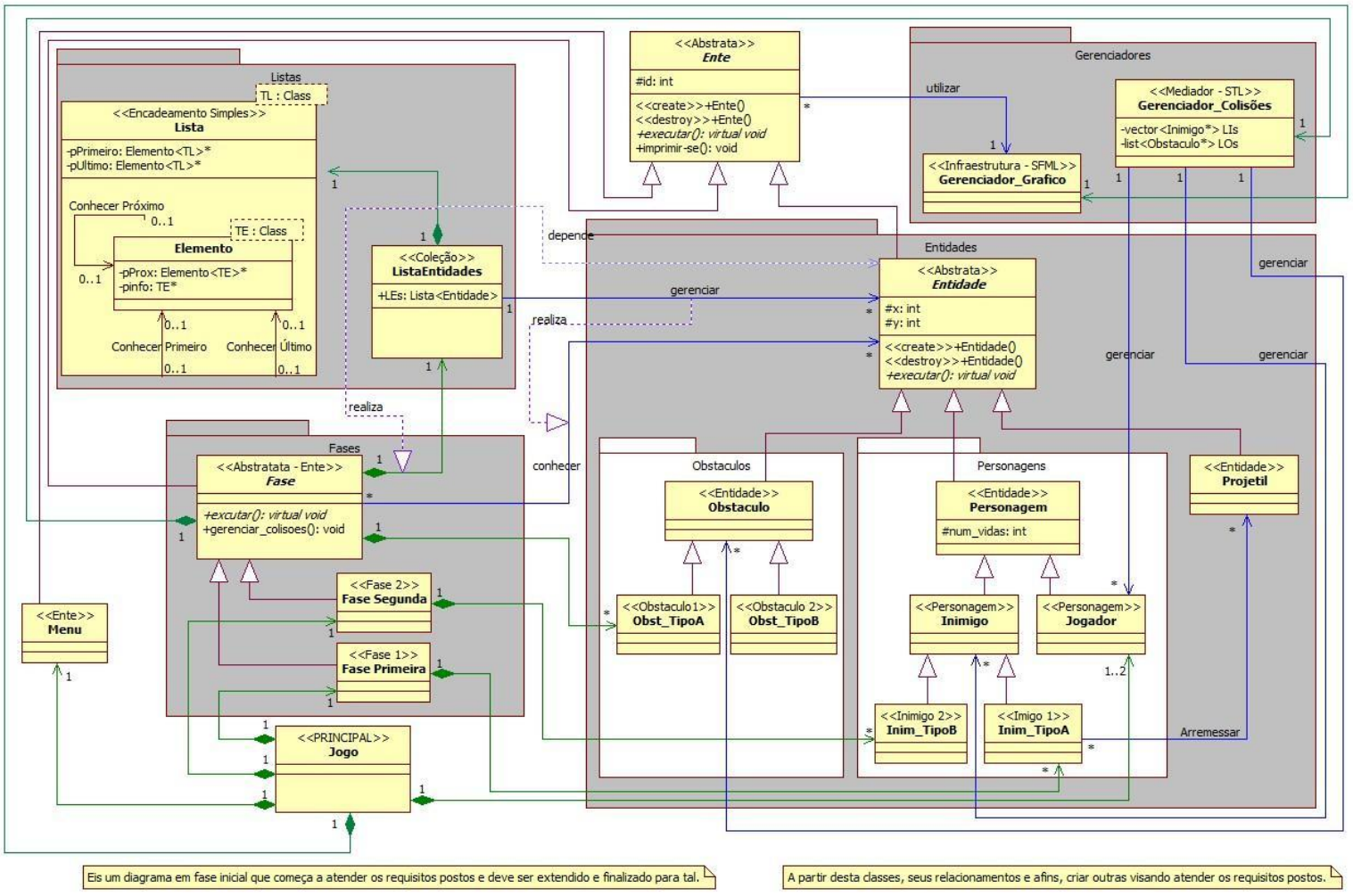

Figura 2. Diagrama de Classes de base em UML.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

Nesta seção, em relação aos conceitos aprendidos, deve-se apresentar uma tabela de conceitos utilizados e não utilizados tal qual a Tabela 2, sendo que a coluna de 'Conceitos' dessa tabela **NÃO** pode ser alterada, absolutamente. Deve-se também apresentar outra tabela justificando o uso ou não uso, tal qual a Tabela 3.

Oportunamente, todas as tabelas que venham a ser utilizadas deverão ser numeradas sequencialmente com algarismos arábicos, conforme o exemplo abaixo. Ainda, não se começa a seção de artigo-relatório diretamente com a tabela, sem antes apresentar texto a

introduzindo. Em verdade, toda seção deve ser começada por texto apropriado, antes de apresentar elementos outros como tabelas, figuras etc.

Tabela 2. Lista de Conceitos Utilizados e Não Utilizados no Trabalho.

	N.	Conceitos	Uso	Onde / O quê
	1	**Elementares:**		
	1.1	- Classes, objetos. & - Atributos (privados), variáveis e constantes. & - Métodos (com e sem retorno).	Sim	Todos .h e .cpp, como nas classes nos *namespaces* X e Y.
	1.2	- Métodos (com retorno *const* e parâmetro *const*). & - Construtores (sem/com parâmetros) e destrutores	Sim	Na maioria dos .h e .cpp, como nas classes nos *namespaces* W e Z.
	1.3	- Classe Principal.	Sim	Main.cpp & Principal.h/.cpp
	1.4	- Divisão em .h e .cpp.	Sim	No desenvolvimento como um todo, como nas classes nos *namespaces* A e B.
	2	**Relações de:**		
	2.1	- Associação direcional. & - Associação bidirecional.	Sim	Em vários dos .h e .cpp, como nas classes nos *namespaces* W e Z.
	2.2	- Agregação via associação. & - Agregação propriamente dita.	Sim	Em vários dos .h e .cpp, como nas classes nos *namespaces* W e Z.
	2.3	- Herança elementar. & - Herança em diversos níveis.	Sim	Em alguns dos .h e .cpp, como nas classes nos *namespaces* W e Z.
	2.4	- Herança múltipla.	Não	Precisamente nos .h e .cpp, das classes C, D e F.
	3	**Ponteiros, generalizações e exceções**		
	3.1	- Operador *this* para fins de relacionamento bidirecional.	Não	Precisamente nos .h e .cpp, das classes X, Y e Z.
	3.2	- Alocação de memória (*new* & *delete*).		. . .
	3.3	- Gabaritos/*Templates* criada/adaptados pelos autores (*e.g.*, Listas Encadeadas via *Templates*).		. . .
	3.4	- Uso de Tratamento de Exceções (*try catch*).		. . .
	4	**Sobrecarga de:**		
	4.1	- Construtoras e Métodos.		. . .
	4.2	- Operadores (2 tipos de operadores pelo menos – Quais?).		Foi usado o *operator*== e o *operator*++ no PPP.h. ***[Especifar qual e onde aqui!]***
	---	**Persistência de Objetos (via arquivo de texto ou binário)**		
	4.3	- Persistência de Objetos.		. . .
	4.4	- Persistência de Relacionamento de Objetos.		. . .
	5	**Virtualidade:**		. . .
	5.1	- Métodos Virtuais Usuais.		. . .
	5.2	- Polimorfismo.		. . .
	5.3	- Métodos Virtuais Puros / Classes Abstratas.		. . .
	5.4	- Coesão/Desacoplamento efetiva e intensa com o apoio de padrões de projeto.		. . .

	6	**Organizadores e Estáticos**		
	6.1	- Espaço de Nomes (*Namespace*) criada pelos autores.		. . .
	6.2	- Classes aninhadas (*Nested*) criada pelos autores.		. . .
	6.3	- Atributos estáticos e métodos estáticos.		. . .
	6.4	- Uso extensivo de constante (*const*) parâmetro, retorno, método...		. . .
	7	**Standard Template Library *(STL)* e String OO**		
	7.1	- A classe Pré-definida *String* ou equivalente. **&** - *Vector* e/ou *List* da *STL* (p/ objetos ou ponteiros de objetos de classes definidos pelos autores)		. . .
	7.2	- Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, Multi-Conjunto, Mapa **OU** Multi-Mapa.		. . .
	---	**Programação concorrente**		
	7.3	- *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time **OU** Win32API ou afins.		. . .
	7.4	- *Threads* (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, **OU** Troca de mensagens.		. . .
	8	**Biblioteca Gráfica / Visual**		
	8.1	- Funcionalidades Elementares. **&** - Funcionalidades Avançadas como: ● tratamento de colisões ● duplo *buffer*		*Especificar aqui quais funcionalidades.*
	8.2	- Programação orientada e evento efetiva (com gerenciador apropriado de eventos inclusive) em algum ambiente gráfico. **OU** - *RAD – Rapid Application Development* (Objetos gráficos como formulários, botões etc).		. . .
	---	**Interdisciplinaridades via utilização de Conceitos de Matemática Contínua e/ou Física.**		
	8.3	- Ensino Médio Efetivamente.		Especificar quais conceitos aqui.
	8.4	- Ensino Superior Efetivamente.		Especificar quais conceitos aqui.
	9	**Engenharia de Software**		
	9.1	- Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. &		. . .
	9.2	- Diagrama de Classes em *UML*.		. . .
	9.3	- Uso efetivo e intensivo de padrões de projeto *GOF*, *i.e.*, mais de 5 padrões.		. . .
	9.4	- Testes à luz da Tabela de Requisitos e do Diagrama de Classes.		. . .
	10	**Execução de Projeto**		
	10.1	- Controle de versão de modelos e códigos automatizado (via github e/ou afins). & - Uso de alguma forma de cópia de segurança (*i.e.*, *backup*).		Especificar qual modo.

	10.2	- Reuniões com o professor para acompanhamento do andamento do projeto. **[ITEM OBRIGATÓRIO PARA A ENTREGA DO TRABALHO]**		Especificar quantidade e quando, sendo o mínimo de 4 reuniões.
	10.3	- Reuniões com monitor da disciplina para acompanhamento do andamento do projeto.		Especificar quantidade e quando (mínimo 10).
	10.4	- Revisão do trabalho escrito de outra equipe e vice-versa.		Especificar qual equipe
	Total de conceitos apropriadamente utilizados. *(Cada grande tópico vale 10% do total de conceitos. Assim, por exemplo, caso se tenha feito metade de um tópico, então valeria 5%.)*			**70%** (setenta por cento). *(Naturalmente e obviamente, este tipo de observação aqui entre parênteses deve ser retirada dos relatórios.)*

Ressalta-se que as legendas das figuras devem ser colocadas abaixo, enquanto as legendas das tabelas devem ser colocadas acima das mesmas. Em tempo, entre tabelas também deve sim ter texto apropriado. Idem entre figuras, as quais sempre são intercaladas por textos pertinentes.

Tabela 3. Lista de Justificativas para Conceitos Utilizados.

No.	**Conceitos**	***Listar apenas os utilizados* Situação**
1	**Elementares**	***FAZER JUSTIFICATIVAS PRAGMÁTICAS***
1.1	- Classes, objetos. & - Atributos (privados), variáveis e constantes. & - Métodos (com e sem retorno).	Classe, Objetos, Atributos e Métodos foram utilizados porque são conceitos elementares na orientação a objetos.
1.2	- Métodos (com retorno *const* e parâmetro *const*). & - Construtores (sem/com parâmetros) e destrutores	A constância de atributos, métodos e afins, quando pertinente, evite mudanças equivocadas. Construtores são finalmente mandatórios para bem inicializar os atributos, enquanto que destrutores ....
1.3	- Classe Principal.	Ter uma classe Principal permite se alcançar um melhor 'purismo' em termos de OO.
1.4	- Divisão em .h e .cpp.	Permite melhor organização das classes e afins que compõem o sistema.
2	**Relações**	***FAZER JUSTIFICATIVAS PRAGMÁTICAS***
2.1	- Associação direcional. & - Associação bidirecional.	Associação foi utilizado porque....
2.2	- Agregação via associação. & - Agregação propriamente dita.	...
2.3	- Herança elementar. & - Herança em diversos níveis.	...
2.4	...	...
...	. . .	...
9	**Engenharia de Software**	***FAZER JUSTIFICATIVAS PRAGMÁTICAS***
10	. . .	...

REFLEXÃO COMPARATIVA ENTRE DESENVOLVIMENTOS

De forma sucinta e suficiente, nesta seção se deve apresentar reflexão comparativa entre o desenvolvimento orientado a objetos e o desenvolvimento procedimental. Isto segundo a percepção dos desenvolvedores nos trabalhos realizados no âmbito da disciplina de Fundamentos de Programação 1 e Técnicas de Programação. Naturalmente, isso também

pode ser baseado em âmbito de outras eventuais disciplinas ou experiências prévias pertinentes.

DISCUSSÃO E CONCLUSÕES

Esta seção deverá apresentar reflexão sobre o desenvolvimento e os resultados obtidos. Certamente uma conclusão bem elaborada auxilia na avaliação do Professor. **Outro item ainda mais fundamental para a avaliação são as reuniões com o professor, sendo que o trabalho não pode ser entregue sem elas terem ocorrido, bem entendido**.

Por sua vez, a avaliação do trabalho como um todo pelo Professor será baseada em:

- Quantidade e qualidade dos requisitos funcionais cumpridos na elaboração do *software*, à luz do conjunto e qualidade da modelagem e códigos, resultando em número apropriado de classes e objetos, número/forma apropriada de relacionamentos e apropriada complexidade algorítmica. Naturalmente, isto tudo envolve particularmente os bons princípios de Orientação Objetos, como organização, encapsulamento e reutilização, todos baseados no princípio de coesão e desacoplamento doutrinados na disciplina.
- Quantidade e qualidade dos conceitos utilizados na elaboração do *software*, em termos de modelagem e realização de código, o que novamente envolve a correção na aplicação dos princípios da Orientação a Objetos (*e.g.*, coesão, desacoplamento, encapsulamento, organização e reutilização), além da utilização apropriada de cada conceito em si, naturalmente.
- O conteúdo das reuniões e as evoluções a partir delas, bem como a qualidade do trabalho escrito, da apresentação, de diagramas, de códigos e afins.

Em termos gerais, pode-se considerar o primeiro item com um peso de 35%, o segundo com um peso de 35% e o terceiro com um peso de 30%, lembrando que eles estão inter-relacionados. Não obstante, esta porcentagem é relativa, pois (por exemplo) um item muito bem desenvolvido pode até compensar (um pouco) outro não tão bem desenvolvido em proporções diferentes deste referencial dado. Lembrar ainda e novamente que fazer o projeto ser acompanhado pelo professor e monitor, à medida que avança, é **fundamental**.

CONSIDERAÇÕES PESSOAIS

Caso os estudantes desejam expressar algum sentimento relativo, por exemplo, aos aprendizados e dificuldades encontradas, isto deve ser feito nesta seção opcional. Nesta seção pode-se até utilizar primeira pessoa, entretanto seria melhor a forma impessoal.

Neste sentido, **todas as demais seções devem ser escritas de forma impessoal** (o que significa **não** usar primeira (e mesmo segunda) pessoa singular ou plural – em suma não usar "eu" ou "nós" no texto). Ademais, salienta-se que o trabalho deve ser redigido em linguagem correta, na forma culta e sem exageros poéticos, com textos não prolixos e bem encadeados.

DIVISÃO DO TRABALHO

Esta seção deverá ter uma tabela salientando quem desenvolveu cada classe/módulo do *software* e realizar demais atividades como as de 'engenharia de *software*', a redação do trabalho escrito, a revisão da redação do trabalho e a preparação da apresentação do trabalho. A tabela 4 pode e mesmo deveria ser melhorada à luz das tabelas de requisitos e conceitos.

Tabela 4. Lista de Atividades e Responsáveis.

Atividades.	Responsáveis
Compreensão de Requisitos	Fulano e Ciclano

Diagramas de Classes	Fulano e Ciclano
Programação em C++	Fulano e Ciclano em geral
Implementação de *Template*	Fulano
Implementação da Persistência dos Objetos...	Ciclano
...	
Escrita do Trabalho	Mais Fulano que Ciclano
Revisão do Trabalho	Mais Ciclano que Fulano

Aqui, após a tabela deve-se **obrigatoriamente** constar o quanto cada um trabalhou no projeto em termos de realização (*e.g.*, modelagem e escrita de código) e colaboração (*e.g.*, revisão de código e testes):

- Fulano trabalhou em 100% das atividades ou as realizando ou colaborando nelas efetivamente.

- Ciclano trabalhou em 20% das atividades ou as realizando ou colaborando nelas efetivamente.

AGRADECIMENTOS

Havendo agradecimentos de ordem profissional, como ajuda de monitores, estes deverão vir antes das referências. Neste sentido, aqui se pode também salientar e agradecer caso outra equipe tenha revisado o trabalho.

REFERÊNCIAS CITADAS NO TEXTO

[1] DEITEL, H. M.; DEITEL, P. J. C++ Como Programar. 5ª Edição. Bookman. 2006.

[2] STADZISZ, P. C. Projeto de Software usando UML. Apostila CEFET-PR 2002. http://www.dainf.ct.utfpr.edu.br/~jeansimao/Fundamentos2/EngSoftware/Apostila%20UML%20-%20Stadzisz%202002.pdf

[3] SIMÃO, J. M. Site das Disciplina de Fundamentos de Programação 2, Curitiba – PR, Brasil, Acessado em 20/06/2021, às 20:32 - http://www.dainf.ct.utfpr.edu.br/~jeansimao/Fundamentos2/Fundamentos2.htm.

REFERÊNCIAS UTILIZADAS NO DESENVOLVIMENTO

[A] BEZERRA, E. Princípios de Análise e Projeto de Sistemas com UML. Editora Campus. 2003. ISBN 85-352-1032-6.

[B] HORSTMANN, C. Conceitos de Computação com o Essencial de C++, 3ª edição, Bookman, 2003, ISBN 0-471-16437-2.