

TRABALHO PARA A DISCIPLINA DE TÉCNICAS DE PROGRAMAÇÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DA UTFPR: *AQUI NOME DO JOGO - MODELO PARA O 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 para o trabalho ele mesmo. 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) usando como recurso o Diagrama de Classes em Linguagem de Modelagem Unificada (Unified Modeling Language - UML) usando como base um diagrama 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 o modelagem elaborado. 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 shows a model for the manuscript to the academic work of Técnicas de Programação as well as it presents general instructions about this academic work. 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, este modelo traz instruções para o trabalho ele mesmo. 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 levantar textualmente requisitos, 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, ademais e sobretudo, 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 trabalhado 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 ou continua apropriado (caso alguém ainda não o tenha feito...). Não obstante, no decorrer deste documento são apresentados os requisitos mínimos 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, 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 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 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 subsequente 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@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 Dropbox. Outrossim, o(s) diagrama(s) de classes sim deve(m) ser entregue(s) de maneira impressa, além de maneira digital.

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 clássico 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 símbolos e legendas nelas contidos.
- 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 negrito, 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 negrito, 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 negrito, 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 negrito, 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 em branco. Todo o texto deverá ser escrito em espaço simples. Para as subseções, somente a primeira letra do subtítulo deve ser maiúscula, sendo todas em negrito, sem numeração, com o título alinhado à esquerda. Uma vez escrito o subtítulo, pular uma linha. Após esta linha (em branco), iniciar o texto da subseção.
- As ilustrações e gráficos podem ser em preto-e-branco ou em escala cinza ou mesmo coloridos, mas sempre centralizados. As notas de rodapé¹ 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). 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 facilitará por demais a compreensão do jogo contemplado no trabalho.**

A propósito, as figuras, tabelas etc., devidamente referenciadas 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.

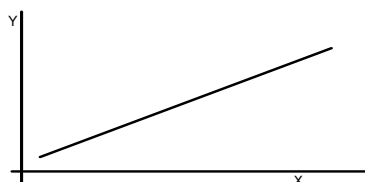


Figura 1. Centralizada na coluna e com legenda abaixo da figura.

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:

$$(1) \quad e(t) = \sum_{n=1}^5 \frac{1}{2+n} \cos(2\pi nt)$$

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

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.

DESENVOLVIMENTO DO/DE JOGO NA VERSÃO PROCEDIMENTAL

~~[ESTA SEÇÃO FOI EXTINTA DESSE MODELO! Ainda assim, queiram a ler para saber como era.]~~

~~Idealmente, o jogo ou *software* em questão teria sido implementado em disciplina anterior sob o viés de abordagem procedimental (i.e., ‘orientação’ a procedimentos ou funções), o que permitiria subsequentemente tecer reflexão comparativa entre a programação (e mesmo projeto) procedimental e a orientada a objetos.~~

~~Isto dito, nesta seção se deveria discorrer a explicação resumida do desenvolvimento do jogo ou *software* em questão caso ele já tivesse sido implementado anteriormente sob o viés de abordagem procedimental em linguagem C (ou mesmo em outra linguagem ‘procedimental’).~~

~~Entretanto, caso o jogo ou *software* em questão não tenha sido previamente implementado sob a abordagem procedimental, pode-se então discutir e mesmo apresentar (sucintamente) algum outro jogo ou *software* feito anteriormente, utilizando abordagem procedimental.~~

~~Isto também permitiria mostrar que se teria experiência prévia para tecer reflexão comparativa entre a abordagem procedimental e a orientada a objetos. Tal comparação se daria na seção chamada COMPARAÇÃO ENTRE DESENVOLVIMENTOS explicada mais ao final deste modelo de artigo-relatório.~~

~~Nesta explicação em questão, sobre desenvolvimento procedimental, aconselhar-se-ia:~~

- ~~— Listar sucintamente os requisitos funcionais do jogo/*software* visado.~~
- ~~— Utilizar um fluxograma para explicar a dinâmica funcional do jogo/*software* visado e mesmo utilizar parte do código. Este último, caso realmente se faça necessário e enquadrado em uma moldura com se fosse uma figura.~~
- ~~— Utilizar, até mesmo, um ‘diagrama de blocos’ e/ou ‘diagrama de fluxo de dados’ para explicar os módulos existentes (e.g. funções ou conjunto de funções) e suas conexões.~~
- ~~— Valorizar as ‘sophistications’ que tenham sido realizadas, como uma eventual função que permita ao jogador humano enfrentar um dado ‘jogador artificial’.~~
- ~~— Evitar frases que não reflitam a realidade como “implementou-se a inteligência artificial”. Este exemplo dado não reflete a realidade porque certamente os autores não implementaram todo conteúdo da disciplina/domínio de Inteligência Artificial. Neste caso o correto seria: “implementou-se uma técnica ou solução que pode ser classificado como uma solução (elementar) de Inteligência Artificial”.~~

DESENVOLVIMENTO DO JOGO NA VERSÃO ORIENTADA A OBJETOS

Nesta seção se deve discorrer a explicação do desenvolvimento do jogo/*software* utilizando orientação a objeto, **salientado diagrama(s) de classes em UML**, 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).

Nesta explicação, deve-se primeiramente listar (textualmente) os **requisitos funcionais** definidos para o jogo/*software* em questão (requisitos mínimos definidos na Tabela 1). 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.

Ademais, quando o requisito estiver como ‘realizado’, ‘semi-realizado’ ou similares, faz-se absolutamente necessário indicar sucintamente quais classes ou conjuntos de classes 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, ademais, os requisitos mínimos que cada jogo deve ter.**

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

N.	Requisitos Funcionais	Situação	Implementação
1	Apresentar menu de opções aos usuários do Jogo.	Requisito previsto inicialmente e realizado.	Requisito cumprido via classe Menu e seu respectivo objeto.
2	Permitir um ou dois jogadores aos usuários do Jogo, sendo que no último caso seria para que os dois joguem de maneira concomitante.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via classe Jogador cujos objetos são agregados em jogo, podendo ser um ou dois efetivamente.
3	Disponibilizar ao menos duas fases que podem ser jogadas sequencialmente ou selecionadas.	Requisito previsto inicialmente, mas realizado apenas PARCIALMENTE – faltou ainda a última fase.	Requisitos foi realizado PARCIALMENTE porque a classe XYZ apenas . . .
4	Ter três tipos distintos de inimigos (o que pode incluir ‘Chefão’, vide abaixo), sendo que pelo menos um dos inimigos deve ser capaz de lançar projétil contra o(s) jogador(es).	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 5 instâncias por tipo.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
6	Ter inimigo “Chefão” na última fase	Requisito previsto mas NÃO realizado.	Requisito NÃO realizado
7	Ter três tipos de obstáculos.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
8	Ter em cada fase ao menos dois tipos de obstáculos com número aleatório de instâncias (i.e., objetos) sendo pelo menos 5 instâncias por tipo.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
9	Ter representação gráfica de cada instância.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .

10	Ter em cada fase um cenário de jogo com os obstáculos.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
11	Gerenciar colisões entre jogador e inimigos, bem como seus projeteis (em havendo).	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
12	Gerenciar colisões entre jogador e obstáculos.	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
13	Permitir cadastrar/salvar dados do usuário, manter pontuação durante jogo, salvar pontuação e gerar lista de pontuação (<i>ranking</i>).	Requisito previsto inicialmente e realizado.	Requisito cumprido inclusive via . . .
14	Permitir Pausar o Jogo	Requisito previsto inicialmente e realizado	Requisito cumprido inclusive via . . .
15	Permitir Salvar Jogada.	Requisito previsto e NÃO realizado	Requisito NÃO realizado

Isto feito, a explicação do desenvolvimento segue devendo-se:

- Utilizar Diagrama(s) de Classes em *UML* para explicar 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 etc².
- 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 ou agrupamentos de classes e como elas se inter-relacionam.
- Em tempo, valorizar as ‘sofisticações’ que tenham sido realizadas, como persistência de objetos relativas ao requisito Permitir Salvar Jogada.
- 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*, *Allegro*, *TCL/TK*, *OpenGL* ou outra, mas preferencialmente uma OO), pois isto valoriza o trabalho esteticamente além de demonstrar a capacidade de ‘pesquisa’ e aprendizado nisso. Não deixe de valorizar esta capacidade de pesquisa e aprendizado nesta seção. Em tempo, no site da disciplina a 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 no trabalho e 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

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

formatação. Na verdade, além disto, uma equipe poderia revisar o trabalho escrito da outra e vice-versa para fins de aprimoramento mútuo.

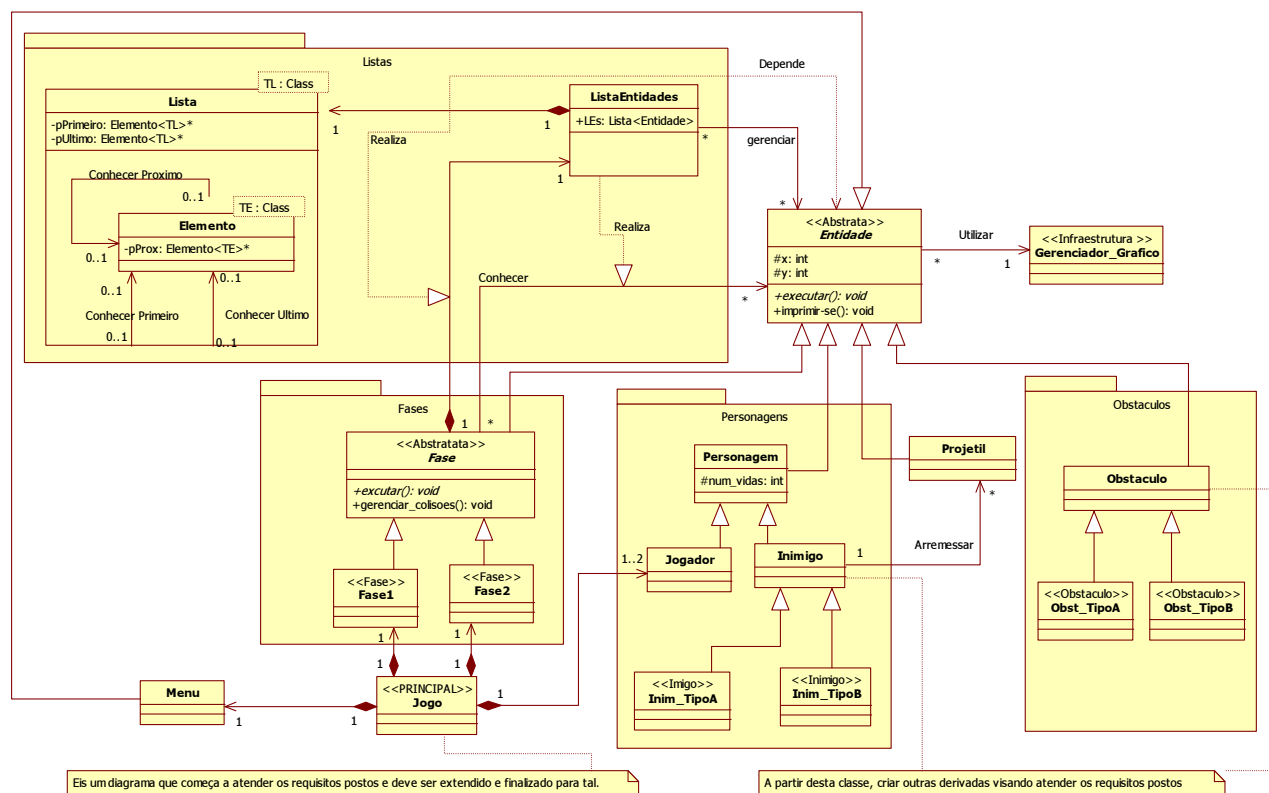


Figura 2. Diagrama de Classes de base em UML.

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:

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

N.	Conceitos	Uso	Onde / O quê
1	Elementares:		
	- Classes, objetos. & - Atributos (privados), variáveis e constantes. & - Métodos (com e sem retorno).	Sim	Todos .h e .cpp
	- Métodos (com retorno <i>const</i> e parâmetro <i>const</i>). & - Construtores (sem/com parâmetros) e destrutores	Sim	Todos .h e .cpp
	- Classe Principal.	Sim	Main.cpp & Principal.h/cpp
	- Divisão em .h e .cpp.	Sim	No desenvolvimento como um todo.
2	Relações de:		

	- Associação direcional. & - Associação bidirecional.		
	- Agregação via associação. & - Agregação propriamente dita.		
	- Herança elementar. & - Herança em diversos níveis.		
	- Herança múltipla.		
3	Ponteiros, generalizações e exceções		
	- Operador <i>this</i> .		
	- Alocação de memória (<i>new & delete</i>).		
	- Gabaritos/ <i>Templates</i> criada/adaptados pelos autores (e.g. Listas Encadeadas via <i>Templates</i>).		
	- Uso de Tratamento de Exceções (<i>try catch</i>).		
4	Sobrecarga de:		
	- Construtoras e Métodos.		
	- Operadores (2 tipos de operadores pelo menos).		
	Persistência de Objetos (via arquivo de texto ou binário)		
	- Persistência de Objetos.		
	- Persistência de Relacionamento de Objetos.		
5	Virtualidade:		
	- Métodos Virtuais.		
	- Polimorfismo		
	- Métodos Virtuais Puros / Classes Abstratas		
	- Coesão e Desacoplamento		
6	Organizadores e Estáticos		
	- Espaço de Nomes (<i>Namespace</i>) criada pelos autores.		
	- Classes aninhadas (<i>Nested</i>) criada pelos autores.		
	- Atributos estáticos e métodos estáticos.		
	- Uso extensivo de constante (<i>const</i>) parâmetro, retorno, método...		
7	Standard Template Library (STL) e String OO		
	- A classe Pré-definida <i>String</i> ou equivalente. & - <i>Vector</i> e/ou <i>List</i> da <i>STL</i> (p/ objetos ou ponteiros de objetos de classes definidos pelos autores)		
	- Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, Multi-Conjunto, Mapa OU Multi-Mapa.		
	Programação concorrente		
	- <i>Threads</i> (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time OU Win32API ou afins.		
	- <i>Threads</i> (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, OU Troca de mensagens.		
8	Biblioteca Gráfica / Visual		
	- Funcionalidades Elementares. & - Funcionalidades Avançadas como: • tratamento de colisões • duplo <i>buffer</i>		<i>Especificar aqui quais funcionalidades.</i>

	- Programação orientada a evento em algum ambiente gráfico. OU - RAD – <i>Rapid Application Development</i> (Objetos gráficos como formulários, botões etc).		
	Interdisciplinaridades por meio da utilização de Conceitos de Matemática e/ou Física.		
	- Ensino Médio.		Especificar quais conceitos aqui.
	- Ensino Superior.		Especificar quais conceitos aqui.
9	Engenharia de Software		
	- Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. &		
	- Diagrama de Classes em <i>UML</i> .		
	- Uso efetivo (quicá) intensivo de padrões de projeto (particularmente GOF).		
	- Testes a luz da Tabela de Requisitos e do Diagrama de Classes.		
10	Execução de Projeto		
	- Controle de versão de modelos e códigos automatizado (via SVN e/ou afins) OU manual (via cópias manuais). & - Uso de alguma forma de cópia de segurança (backup).		Especificar qual modo.
	- Reuniões com o professor para acompanhamento do andamento do projeto.		Especificar quantidade e quando (mínimo 4).
	- Reuniões com monitor da disciplina para acompanhamento do andamento do projeto.		Especificar quantidade e quando (mínimo 10).
	- Revisão do trabalho escrito de outra equipe e vice-versa.		Especificar qual equipe

Ressalta-se que as legendas das figuras devem ser colocadas abaixo, enquanto que as legendas das tabelas devem ser colocadas acima das mesmas.

Tabela 3. Lista de Justificativas para Conceitos Utilizados e **Não** Utilizados no Trabalho.

No.	Conceitos	<i>Listar apenas os utilizados Situação</i>
1	Elementares	Classe e Objetos foram utilizado porque ... Atributos foram utilizados porque
2	Relações	Associação foi utilizado porque.... ...
...
9	Engenharia de Software	... Padrões de Projeto não foram utilizados porque....
10

REFLEXÃO COMPARATIVA ENTRE DESENVOLVIMENTOS

Nesta seção se deve apresentar uma reflexão comparativa (sucinta e suficiente) entre o desenvolvimento orientado a objetos e o desenvolvimento procedimental segundo a percepção dos desenvolvedores nos trabalhos realizados no âmbito da disciplina de Fundamentos de

Programação 1 e Técnicas de Programação (ou até no âmbito de outras disciplinas ou experiências prévias se for pertinente).

DISCUSSÃO E CONCLUSÕES

Esta seção deverá apresentar uma reflexão sobre o desenvolvimento e os resultados obtidos. Certamente uma conclusão bem elaborada ajuda na avaliação do Professor. Por sua vez, a avaliação do trabalho como um todo (pelo Professor) será baseada em:

- Quantidade e qualidade de conceitos utilizados na elaboração do *software*, envolvendo particularmente os bons princípios de Orientação Objetos, como organização, encapsulamento e reutilização, todos baseados no princípio de coesão e desacoplamento doutrinados em classe.
- Complexidade do problema e respectiva quantidade e qualidade de projeto e código o que, novamente, envolve a correção na aplicação dos princípios da Orientação a Objetos (i.e. coesão, desacoplamento, encapsulamento, organização e reutilização), o número de classes e objetos, o número e a forma de relacionamento, a complexidade algorítmica, o cumprimento de requisitos etc.
- Qualidade do trabalho escrito e da apresentação.

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 (sobretudo o primeiro e o segundo). 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 que fazer seu projeto ser acompanhado pelo professor e monitor (a medida que avança) é **fundamental**.

CONSIDERAÇÕES PESSOAIS

Caso o aluno ou os alunos desejam expressar algum sentimento relativo, por exemplo, aos aprendizados e dificuldades encontradas, isto deve ser feita 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 realizou 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.

Tabela 4. Lista de Atividades e Responsáveis.

Atividades.	Responsáveis
Levantamento de Requisitos	Fulano e Ciclano
Diagramas de Classes	Fulano e Ciclano
Programação em C++	Fulano e Ciclano em geral
Implementação de <i>Template</i>	Fulano
Implementação da Persistência dos Objetos...	Ciclano

...	
Escrita do Trabalho	Mais Fulano que Ciclano
Revisão do Trabalho	Mais Ciclano que Fulano

AGRADECIMENTOS

Havendo agradecimentos de ordem profissional, estes deverão vir antes das referências. Aqui se pode 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 09/09/2015, às 15:15:
<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.