Especificação do Trabalho Final de MLP
Turma A

Prof. Leandro Krug Wives

Maio de 2017

1 Visão Geral

O objetivo deste trabalho consiste em fornecer aos alunos a oportunidade de **estudar uma linguagem de programação moderna com características híbridas** (i.e., multiparadigma). O trabalho permitirá aos alunos **demonstrarem que aprenderam os princípios de programação relacionados com os diferentes paradigmas** estudados ao longo do semestre, demonstrando, ainda, a **capacidade de analisar e avaliar linguagens de programação, seguindo os critérios abordados em aula**.

2 Detalhamento do Trabalho

Para tanto, cada grupo, formado de três pessoas, **deve selecionar uma linguagem de programação da lista abaixo** (no máximo três grupos por linguagem):

- **C++ 14** (maiores detalhes em: `http://isocpp.org/wiki/faq/cpp14-language`, `http://en.wikipedia.org/wiki/C++14`)
- **Clojure** (maiores detalhes em: `http://clojure.org/`)
- **F#** (maiores detalhes em: `http://fsharp.org/`)
- **Groovy** (maiores detalhes em: `http://www.groovy-lang.org/`)
- **Java 8** (maiores detalhes em: `http://www.java.com/pt_BR/download/faq/java8.xml`)
- **Julia** (maiores detalhes em: `http://julialang.org/`)
- **Objective-C** (maiores detalhes em: `http://en.wikipedia.org/wiki/Objective-C` `http://en.wikipedia.org/wiki/Objective-C`)
- **Objective CAML** (maiores detalhes em: `http://ocaml.org/`)
- **Python** (maiores detalhes em: `http://python.org/`)
- **Ruby** (maiores detalhes em: `http://ruby-lang.org/`)
- **Scala** (maiores detalhes em: `http://www.scala-lang.org/`)

- **Swift 3.0** (maiores detalhes em: `http://developer.apple.com/swift`, `http://swift.org/`)

A tarefa principal do trabalho consiste em experimentar e comparar as características e funcionalidades orientadas a objeto e funcionais da linguagem de programação escolhida. **Após selecionar definir um grupo e selecionar uma linguagem, é preciso especificar um problema a ser solucionado com ela** (ver opções de problema a seguir). O problema será, então, solucionado duas vezes na mesma linguagem, uma delas usando somente Orientação a Objetos e a outra usando características funcionais. Interfaces (gráficas ou textuais) podem ser feitas em qualquer paradigma ou plataforma, podendo inclusive serem compartilhadas entre as duas versões, visto que o foco não está na interface do programa.

2.1 Problemas disponíveis

- **xRisk (xWar)**. A ideia é desenvolver um jogo de batalha por turnos estilo o jogo americano Risk (`http://en.wikipedia.org/wiki/Risk_(game)/`) ou a versão Brasileira War (`http://pt.wikipedia.org/wiki/War`). A fim de tornar o jogo menos complexo e menos demorado, sua versão pode envolver somente dois adversários (seja outro ser humano ou o computador). Preferencialmente, o jogo pode utilizar a API do Google Maps para posicionar os exércitos.

- **SuperMarioBrosAI**. A ideia consiste em desenvolver um software que controle o personagem Mário Bros para o ambiente InfiniteMarioBros, utilizado na Mario AI Competition (`http://julian.togelius.com/mariocompetition2009/`). O Mário deve coletar o maior número de moedas no menor espaço de tempo, sem morrer.

- **Desenvolver um jogo de Batalha Naval**. O computador deve sortear uma configuração ao inicial do jogo, em que estarão colocados num tabuleiro de tamanho 15x15 os seguintes itens: 4 submarinos (2 casas), 3 navios (3 casas) e 5 minas (1 casa cada). Em cada jogada, o computador lê as coordenadas (linha e coluna) da casa em que o usuário quer atingir e indica o resultado, ou seja, se acertou na água ou em parte de um navio (navio inteiro se for uma mina). O jogo termina quando o usuário afundar toda a frota, ou quando indicar que não quer continuar a jogar. Ao invés de solicitar as coordenadas, você pode usar o mouse como entrada de dados.

- **Implementação de um jogo do tipo *Tower Defence***. Neste tipo de jogo você precisa defender algum elemento ou posição na tela, normalmente em algum cenário composto de uma ou mais estradas ou caminhos que são percorridos por uma série de inimigos (por rounds). A cada round você tem um saldo a gastar em torres ou elementos de defesa (ou ainda em `upgrades`), que podem ser posicionados em locais fixos ou abertos ao longo do cenário. Esses elementos de defesa devem atacar os inimigos, destruindo-os antes que cheguem ao alvo. Cada inimigo tem um poder de ataque, cura ou quantidade de vida específico, o qual diminui cada vez que recebe algum tiro de defesa. Cada vez que um inimigo é acertado ou morto, você ganha créditos. O jogo termina quando uma quantidade x de inimigos chega no objetivo ou quando seu ponto de defesa fica muito fraco. Maiores detalhes em: `http://en.wikipedia.org/wiki/Tower_defense/`.

- **Simulador de determinação de escopo (dinâmico versus estático)**. Ou seja, desenvolver um simulador capaz de aceitar definições de subprogramas e variáveis locais, utilizando uma pseudolinguagem simples. Com base nisso, demonstrar como ficaria sua pilha de chamadas (call-stack) e o conteúdo das variáveis locais a cada passo de execução.

- **Escolha de um problema pessoal ou motivador**. No caso, o grupo deve encaminhar sua ideia ao professor, descrita em detalhes, que avaliará sua viabilidade.

3 Definição dos grupos

Uma vez que um grupo tenha sido formado e os participantes tenham escolhido um problema e definido a linguagem que o grupo irá utilizar, um dos participantes deverá informar no Moodle (via ferramenta disponibilizada pelo professor) um nome de grupo, a lista de pessoas que fazem parte do grupo, a linguagem escolhida e o problema a ser resolvido. Trabalhos que não sejam em grupos de 3 pessoas devem ser evitados. Em último caso, se não houver outra opção, grupos com menos ou mais pessoas devem ser discutidos e autorizados pelo professor. Também não é possível realizar trabalhos em grupos com alunos de outras turmas.

4 Recursos Necessários (critérios mínimos)

O trabalho realizado **deve considerar os aspectos especificados nesta seção** (sendo um conjunto específico de recursos para a solução orientada a objetos e outro para a solução funcional). **Caso um recurso não esteja disponível** na linguagem, **explique e justifique** os motivos para ele não existir **e utilize um mecanismo alternativo**.

Segue a lista de aspectos necessários de serem estudados durante o estudo da linguagem e que devem estar presentes no relatório do grupo:

- Requisitos para a solução orientada a objetos:
  - Especificar e utilizar classes (utilitárias ou para representar as estruturas de dados utilizadas pelo programa).
  - Fazer uso de encapsulamento e proteção dos atributos, com os devidos métodos de manipulação (setters/getters) ou propriedades de acesso, em especial com validação dos valores (parâmetros) para que estejam dentro do esperado ou gerem exceções caso contrário.
  - Especificação e uso de construtores-padrão para a inicialização dos atributos e, sempre que possível, de construtores alternativos.
  - Especificação e uso de destrutores (ou métodos de finalização), quando necessário.
  - Organizar o código em espaços de nome diferenciados, conforme a função ou estrutura de cada classe ou módulo de programa.
  - Usar mecanismo de herança, em especial com a especificação de pelo menos três níveis de hierarquia, sendo pelo menos um deles correspondente a uma classe abstrata, mais genérica, a ser implementada nas classes-filhas.

  - Utilizar polimorfismo por inclusão (variável ou coleção genérica manipulando entidades de classes filhas, chamando métodos ou funções específicas correspondentes).
  - Usar polimorfismo paramétrico:
    * através da especificação de *algoritmo* (método ou função genérico) utilizando o recurso oferecido pela linguagem (i.e., generics, templates ou similar)
    * e da especificação de *estrutura de dados* genérica utilizando o recurso oferecido pela linguagem.
  - Usar polimorfismo por sobrecarga (vale construtores alternativos).
  - Especificar e usar delegates.

- Recursos para a solução funcional:
  - Priorizar o uso de elementos imutáveis e funções puras (por exemplo, sempre precisar manipular listas, criar uma nova e não modificar a original, seja por recursão ou através de funções de ordem maior).
  - Especificar e usar funções não nomeadas (ou lambda).
  - Especificar e usar funções que usem currying.
  - Especificar funções que utilizem pattern matching ao máximo, na sua definição.
  - Especificar e usar funções de ordem superior (maior) criadas pelo programador.
  - Usar funções de ordem maior prontas (p.ex., map, reduce, foldr/foldl ou similares).
  - Especificar e usar funções como elementos de 1ª ordem.
  - Usar recursão como mecanismo de iteração (pelo menos em funções de ordem superior que manipulem listas).

5 Relatório

O grupo deve apresentar um relatório técnico com os itens descritos abaixo. Sugere-se que este relatório seja escrito utilizando recursos do LaTeX. Um modelo de relatório (tanto em LaTeX quanto em .doc, entra-se no Moodle). Segue a lista dos itens obrigatórios:

1. **Capa**. Com identificação do grupo e da linguagem escolhida.
2. **Visão geral da Linguagem**. Apresentação da linguagem escolhida, descrevendo suas características, fundamentos, funcionalidades, benefícios e principais aplicações (inclusive com discussão de sua aplicabilidade em questões práticas).
3. **Análise Crítica**. Uma análise crítica da linguagem estudada, envolvendo uma tabela com os critérios e propriedades estudados em aula (i.e. simplicidade, ortogonalidade, expressividade, adequabilidade e variedade de estruturas de controle, mecanismos de definição de tipos, suporte a abstração de dados e de processos, modelo de tipos, portabilidade, reusabilidade, suporte e documentação, tamanho de código, generalidade, eficiência e custo, e outros que o grupo achem convenientes),

com notas/valores justificados (ilustrando com exemplos utilizados no código ou descrevendo situações que contariam como pontos favoráveis ou desfavoráveis para cada critério ou propriedade). Indicar qual paradigma foi mais adequado para resolver o problema e por que.

4. **Conclusões**. Conclusões, descrevendo as facilidades e dificuldades encontradas, benefícios, problemas e limitações da linguagem estudada.

5. **Bibliografia**. Todo material consultado ou não, incluindo livros, artigos, páginas na Internet, etc., que tenha relação com o assunto. Elaborar a lista preferencialmente usando *Bibtex*.

Atenção: não serão aceitos trabalhos com indícios de plágio (cópia integral ou parcial de outros trabalhos). Utilizar trechos e exemplos, mesmo que em forma de paráfrase, é permitido e estimulado, desde que a menção (citação) ao autor do original seja feita corretamente.

6 Boas Práticas

Sugere-se uma lista de boas práticas para a execução deste trabalho. Elas são opcionais. Caso o grupo tenha interesse em utilizá-las, o professor pode auxiliar no seu uso durante o semestre.

Trello - para manter o registro de cada atividade e seus responsáveis.

Editor online de documentos - para que todos possam criar e editar de maneira colaborativa o relatório final (sugere-se ShareLatex ou Overleaf).

Github ou Bitbucket - para gerenciar o desenvolvimento em grupo e manter um repositório único de código, permitindo não só gerenciar versões, mas também controlar a contribuição de cada participante.

Máquina Virtual ou Docker - para que você possa configurar todas as bibliotecas, plug-ins e componentes necessários para o desenvolvimento e a execução de seu software.

7 Forma da Entrega

A entrega do trabalho é realizada através de um arquivo compactado (zip, tar ou gzip), contendo o relatório (em PDF) e os códigos-fontes desenvolvidos (não incluir os executáveis), o qual deverá ser encaminhado até a data estipulada pelo professor e indicada em atividade específica de entrega via Moodle.

8 Prazos

O prazos de cada etapa (i.e., "Definição do Trabalho Final" e "Entrega do Trabalho Final") serão definidos nas respectivas atividades disponibilizadas no Moodle e serão rigorosamente observados. Qualquer atraso implicará em perda de nota (10% a cada dia).

9 Apresentação

Os resultados dos trabalhos serão apresentados em datas definidas no cronograma da disciplina disponibilizado na plataforma de apoio pedagógico.

10 Avaliação

A avaliação do trabalho incluirá os seguintes critérios: desenvolvimento e detalhamento dos itens do relatório, aplicação dos conceitos de programação estudados, utilização correta dos recursos da linguagem escolhida, correção, legibilidade, confiabilidade e originalidade, uso de referências, formatação e estilo do texto. Outros aspectos de avaliação poderão ser incluídos a critério do professor. O peso deste trabalho corresponde ao valor especificado no plano da disciplina disponível na plataforma de apoio pedagógico.

Atenção: conforme instruções presentes no plano de ensino da disciplina, todas as etapas do trabalho devem ser cumpridas para que a sua nota de trabalho seja contabilizada!

11 Dúvidas

Em caso de dúvidas, não hesite em consultar o professor.