Técnicas de Programação 1
Plano de Ensino

Rodrigo Bonifácio

10 de agosto de 2017

1 Identificação

- Professor: Rodrigo Bonifácio
- Semestre: 2/2017
- Horário: TER a QUI (10:00 — 11:50)
- Salas: PJC-60

2 Objetivo

O objetivo da disciplina é fazer com que os alunos conheçam técnicas para o desenho adequado (reusável, flexível) de um software usando recursos presentes na Orientação a Objetos (polimorfismo, herança, bibliotecas e frameworks).

Para atingir esse objetivo, a disciplina apresenta técnicas e boas práticas de programação comumente usadas em OO, como a adoção de testes unitários, refatoramento de código e padrões de projeto. Aulas expositivas serão intercaladas com exercícios práticos (sessões de DOJO) e acompanhamento de projetos, nos quais os alunos devem se familiarizar com a linguagem Scala (ou, em uma situação muito, muito específica, alguma outra linguagem OO como Java, C++, Python, Smalltalk).

3 Ementa

Esta disciplina objetiva cobrir aspectos conceituais e práticos associados à Orientação a Objetos. O conteúdo abordado inclui conceitos como Classes e Objetos, Herança e Polimorfismo (tanto por subtipos quanto paramétrico), Padrões de Projeto, Refatoramento de Código OO, Testes Unitários e Desenvolvimento Dirigido a Testes.

4 Programa Básico

O programa da disciplina está organizado nos seguintes módulos:

M1 Conceitos essenciais da programação OO, como classes, objetos, herança, polimorfismo e controle de exceções.

M2 Aspectos pragmáticos do desenho de software OO, como o uso de notações UML, Desenvolvimento Dirigido a Testes, persistência e mapeamento objeto-relacional.

M3 Qualidade de software OO e boas práticas de desenvolvimento, cobrindo métricas, padrões de projeto e refatoração de código.

5 Avaliação

P1 Projeto em grupo explorando o conteúdo do módulo M1. A entrega deve ser agendada para a primeira ou segunda semana após o primeiro módulo da disciplina (provavelmente segunda quinzena de setembro). A nota é individual, e leva em consideração a apresentação do projeto.

P2 Projeto em grupo explorando os conteúdos dos módulos M2 e M3. O projeto consiste na implementação de um interpretador para uma linguagem de programação relativamente simples (MiniHaskell). Entrega prevista para a penúltima semana de aula. A nota é individula, e leva em consideração a apresentação do projeto.

P3 Projeto de desenvolvimento conduzido ao longo da disciplina. Pelo menos três entregas do projeto devem ser realizadas, onde a primeira, que deve ser feita até o dia 24/08/2017, consiste na especificação do projeto.

$$NF = (\frac{P1 \times 0.3 + P2 \times 0.7}{2}) + (P3 \times 0.5)$$

Reforçando. Entregas intermediárias do projeto podem ser solicitadas, com o intuito de permitir um melhor acompanhamento do projeto. Tais entregas intermediárias influenciam na nota do projeto. As entregas estarão sujeitas a apresentação e a avaliação é individual.

6 Regras para o projeto

Esse semestre os alunos devem se organizar em equipes com até cinco participantes. Cada grupo deve ter um gerente de projetos, responsável pela distribuição das atividades de desenvolvimento. Cada grupo deve ter um responsável pela gerência de configuração do software (preferencialmente um aluno com alguma experiência com o GIT). Seguiremos uma abordagem de software ágil, distribuida e *open-source*, onde o acompanhamento das contribuições será feito pelo

GitHub. As contribuições dos alunos em termos de código fonte no repositório, bem como nas dicussões sobre as decisões de desenvolvimento, serão consideradas na atribuição da nota individual para o projeto.

Referências

[1] Selecão de artigos publicados em bibliotecas digitais (como as da ACM e IEEE). Sempre que pertinente, serão sugeridas leituras de artigos apresentados em conferências de impacto. Links para esses artigos serão disponibilizados no site da disciplina.

[2] Kent Beck. *Test Driven Development: By Example*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[3] Bruce Eckel. *Thinking in C++*. Prentice-Hall, Inc., 1995.

[4] Martin Fowler. *Refactoring: improving the design of existing code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[5] Erich Gamma, Richard Helm, Ralph E. Johnson, and John M. Vlissides. *Design Patterns: Elements of ReusableObject-Oriented Software*. Addison-Wesley, 1994.

[6] Cay Horstmann and Gary Cornell. *Core Java(Tm) Volume 1 / 2*. Prentice Hall, nona edição edition, 2013.

[7] Robert C Martin. *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009.

[8] Steve McConnell. *Code complete*. Pearson Education, 2004.

[9] Scott Meyers. *Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition)*. Addison-Wesley Professional, 2005.

[10] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in scala*. Artima Inc, 2008.

[11] Stephen Prata. *C++ Primer Plus (Fourth Edition)*. Sams, Indianapolis, IN, USA, 4th edition, 2001.