Métodos Avançados de Programação

Auto Avaliação

Para cursar a disciplina:

1. Você deve ser um bom programador;
2. Você deve estar familiarizado com os conceitos de Orientação a Objeto;
3. Você deve saber *bem* pelo menos uma linguagem orientada a objeto.

Para avaliar se está pronto para cursar esta disciplina, responda as perguntas abaixo.

- Se você resolver os exercícios com facilidade (sem consultar nenhuma bibliografia e usando uma ou duas horas): você está **pronto** para cursar a disciplina.
- Se você conseguiria resolver os exercícios dado mais tempo (um dia) e consultando bibliografia: você pode cursar a disciplina mas tem que **suprir deficiências**.
- Se você tiver dificuldade ou não conseguir resolver os exercícios: você **não está pronto**.

Exercícios

Nos exercícios abaixo, onde aparecer "Java", você poderá também usar C++/STL.

1. Defina o que é feito na etapa de *Análise* e o que é feito na etapa de *Projeto* ao desenvolver software.
2. Enumere as vantagens da abordagem Orientada a Objetos para o desenvolvimento de software.
3. Qual é o motivo de levantar Requisitos Funcionais para desenvolver software e o que faz parte de uma descrição de Requisitos Funcionais?
4. Mostre como instanciar um objeto da classe ContaBancária em Java fornecendo o CPF (um string) do titular como argumento. Com o objeto resultante, faça um depósito de R$100,00 e imprima o saldo. Você pode escolher nomes apropriados para os métodos.
5. Explique o que é um Iterator em Java. Qual é sua principal vantagem?
6. Mostre a implementação de uma classe ContaBancária. Invente atributos e métodos.
7. Explique a diferença de funcionamento entre um "return" e um "throw". Seja específico.
8. Mostre, usando Java, como especializar uma classe ContaBancária para criar uma ContaCorrente e uma ContaPoupança.
9. Explique as vantagens e desvantagens do polimorfismo. Dê exemplos.

10. Explique a afirmação: "Em Java, o conceito de interfaces permite obter mais polimorfismo do que seria possível com classes abstratas".
11. Qual é a diferença entre "herança de tipo" e "herança de implementação"?
12. Quais são as vantagens e desvantagens de acoplamento forte entre objetos?
13. Ao falar de boa programação, fala-se: "A decomposição deve esconder algo." O que poderia ser escondido, por exemplo?
14. O que é uma "responsabilidade de uma classe"? Por que queremos minimizar o número de responsabilidades? "Mais" não seria melhor?
15. Por que modelar papeis (roles) através de herança é inferior a modelá-los através de composição?