Lista 2 de P.O.O

Herança e Polimorfismo == 2019

Implemente todos os exercícios abaixo em Java e Ruby (exceto quando indicada preferência por uma linguagem específica). Evitem copiar a implementação de uma linguagem para outra; pensem nas possibilidades de cada linguagem.

Todos os exercícios devem ser implementados com testes automatizados, de preferência com TDD.

1) Crie um programa orientado a objetos que calcule uma folha de pagamento, retornando uma coleção de nomes e salários a receber dos funcionários de uma empresa. Todos os funcionários têm seus salários nominais descontados de 10% de previdência. Funcionários com cargo de gerência têm um adicional de 30% no salário. Funcionários com cargo de diretor têm um adicional de 40%. Funcionários com filhos até 6 anos recebem auxílio-creche no valor de 400 reais para cada filho, com um limite de 3. Os funcionários também recebem um auxílio-alimentação no valor de 280 reais. Os auxílios não geram desconto de previdência.

2) Implemente a lógica (ou seja, nada de interface gráfica) de um jogo da velha utilizando orientação a objetos. Na medida do possível, procure distribuir a lógica do jogo.

3) Crie um programa que, dado um grafo orientado cujas arestas possuem pesos, calcule o menor caminho entre dois nós. Deve ser possível escolher, polimorficamente, entre os algoritmos de Dijkstra e Bellman-Ford.

4) (Somente Java) Crie uma classe UENFList que implementa uma lista encadeada (esqueça que o Java já possui uma). UENFList deve suportar tipos genéricos e implementar a interface java.util.List. Crie também uma classe abstrata para implementar algumas operações da interface List que possam ser resolvidos com outras operações da própria interface List.

5) Implemente uma máquina de estados simples, onde seja possível definir estados, um estado inicial e as transições possíveis. A máquina deve assegurar que as transições sejam respeitadas e as mudanças de estado associadas sejam realizadas, disparando uma exceção para requisições de mudanças de estado não autorizadas. Por exemplo, podemos dizer que uma porta possui três estados: "aberta", "fechada" e "quebrada". Uma porta "aberta" só aceita o evento "fechar", que a leva ao estado "fechada". Uma porta "fechada" aceita os eventos "abrir" e "arrombar", que a levam, respectivamente, aos estados "aberta" e "quebrada". Como exemplos de eventos não autorizados, uma porta "aberta" não pode receber eventos "abrir" ou "arrombar" e uma porta "quebrada" não pode receber evento algum. Um exemplo de uma máquina de estados pode úser visto [aqui](https://github.com/nsi-iff/fluidity) (<https://github.com/nsi-iff/fluidity>).

6) Um aluno, ao montar seu plano de estudo, somente pode incluir disciplinas cujos pré-requisitos ele já tenha cursado. Toda disciplina tem um número de créditos, e o plano de estudo deve ter um número máximo e mínimo de créditos para ser válido. Implemente este cenário.

7) Faça a implementação do seguinte problema: Uma empresa quer manter o registro da vida acadêmica de todos os funcionários, o modelo deve contemplar o registro das seguintes informações, de forma incremental:

a)Para o funcionário que não estudou, apenas o nome e o código funcional;

b)Para o funcionário que concluiu o ensino básico, a escola;

c)Para o funcionário que concluiu o ensino médio, a escola;

d)Para o funcionário que concluiu a graduação, a Universidade;

Estenda o modelo implementado de forma que todo funcionário possua uma renda básica de R$ 1000,00 e:

e) Com a conclusão do ensino básico a renda total é renda básica acrescentada em 10%;

e) Com a conclusão do ensino médio a renda total é a renda do nível anterior acrescentada em 50%;

f) Com a conclusão da graduação a renda total é a renda do nível anterior acrescentada em 100%;

Todos os cálculos são efetuados sempre sobre a última renda obtida.