Problema 1: FoliTron – Um Sistema de Controle de Blocos e Folias Carnavalescas

Descrição:

O formando de Engenharia de Computação Odal Odece (vulgo OO), cansado de não fazer nada devido às longas férias de verão da UEFS, resolveu brincar o carnaval 2014. Como ele soube por um colega de Civil que ia sair um bate-volta do bairro Feira VI para Salvador já com abadá e bebida-free incluso, não pensou duas vezes, juntou seus últimos trocados de bolsista PIBIC e se jogou na oportunidade.

Contudo, como as festas populares na Bahia costumam sofrer com a falta de planejamento, OO perdeu o horário do bloco. A organização da folia, em "comum acordo" decidiu antecipar a saída do bloco devido ao atraso de artistas e falhas mecânicas de alguns trios-elétricos. Com a saída antecipada OO se desencontrou de seus amigos de folia e perdeu seu "esquema" pré-arranjado de carnaval, desperdiçando com isso um longo investimento de horas de conversa madrugada adentro via Facebook e WhatsApp. Para piorar sua situação, OO perdeu o horário do bate-volta que, em comum acordo com os demais viajantes (que por sinal preferiram ficar na bebida-free ao invés de correr atrás do bloco), resolveu "antecipar" seu retorno para Feira de Santana.

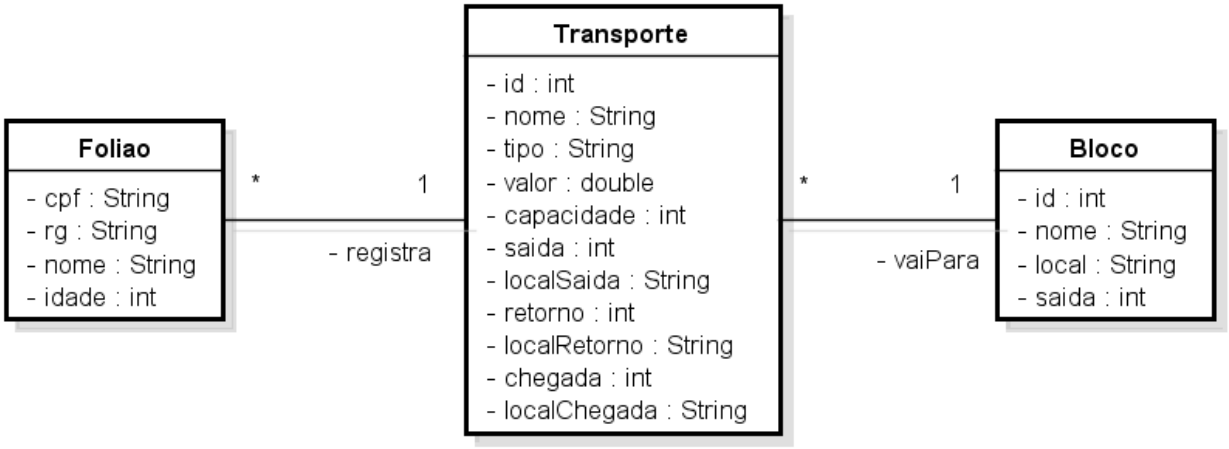
Como resultado, ao final da sua aventura carnavalesca, e dentro do princípio otimista de que "brasileiro não desiste nunca", OO percebeu uma grande oportunidade de negócio. Em um rápido *survey* com outros integrantes que também perderam o bate-volta, ele descobriu que outros carnavais brasileiros também sofrem com os mesmos problemas de: perda de horários de blocos, desencontros entre pessoas, falta de locomoção origem-folia, falta de identificação de pontos de alimentação e higiene, falta de conhecimento de eventos de folia dentro do próprio carnaval da região, etc. Ou seja, problemas diversos que poderiam ser amenizados caso o folião tivesse a sua disposição um sistema para celular de controle de blocos e folias carnavalescas.

Ao retornar para Feira de Santana e ainda empolgado com a sua ideia inovadora, OO começa a rascunhar um modelo conceitual do FoliTron: "Um Bloco Carnavalesco é cadastrado no sistema com seu nome, local e horário de saída. Um Transporte pode ser cadastrado no sistema com seu nome, tipo de transporte, valor, limite de passageiros, local-horário de saída para o Bloco, local-horário de coleta dos foliões após término do Bloco para retorno, e local-horário final de chegada do transporte. O código de identificação do Bloco e do Transporte é gerado automaticamente pelo sistema. Um Folião é cadastrado no sistema com seu nome, CPF, RG e idade. Não pode haver cadastros duplicados de Blocos Carnavalescos, Transportes e Foliões. O sistema só deve permitir o cadastro de foliões em transportes que tenham vaga, o sistema também deve permitir a consulta de Transportes para um determinado evento, o qual deve informar: horário-local de saída, horário-local de retorno, horário-local de chegada, tipo de transporte, valor e vagas disponíveis. O sistema também deve permitir a consultar os foliões registrados em um determinado transporte.".

Percebendo que iria ter um pouco de trabalho para fazer o FoliTron, e tendo restado pouco dinheiro após os festejos de Momo, OO contratou você, estudante do segundo semestre do curso de Engenharia de Computação, como estagiário, desenvolvedor e gerente do projeto. Contudo, ainda empolgado com seu brilhante *insight* e focando na Micareta 2014 que está chegando, OO quer algo funcional em apenas um mês. Por conta disso, ele resolveu lhe dar um tratamento de choque, dizendo, "meu irmão, o negócio é o seguinte, você vai ter que pegar essa galinha voando. Preciso de uma coisa funcional em um mês, pra ter uma ideia melhor das possibilidades. Por isso, montei umas *user stories*

do sistema e criei também um modelo conceitual do problema". Você se pergunta o que são essas coisas, mas ele já adianta que "você vai se acostumar com esse papo todo bem rapidinho, que a prática é o melhor professor".

Modelo Conceitual:



User Stories:

User Story	Título	Breve Descrição
1	Manutenção de Bloco	Inserir e consultar Bloco.
2	Manutenção de Transporte	Inserir e consultar Transporte.
3	Manutenção de Folião	Inserir e consultar Folião.
4	Listar Blocos	Mostrar todos os Blocos que irão sair em um determinado local e período.
5	Listar Transportes	Mostrar todos os Transportes cadastrados para um Bloco.
6	Registrar Folião no Transporte	Inserir um registro de um Folião em um Transporte.
7	Listar Foliões de um Transporte	Mostrar todos os Foliões que estão registrados em um Transporte.

OO ainda disse para você: "Tenho de trabalhar aqui na interface do sistema, o tempo é curto. Então, te vira, tá tudo lá no site do nosso projeto. Vamos fazendo as coisas à medida que a gente conversa com o cliente, ok?" Você para então para analizar o material que OO lhe passou. Senta por algumas horas lendo e relendo o modelo conceitual e as tais *user stories*. Procurando manter a calma, você vai para casa, tentando saber como vai fazer para sair dessa.

No dia seguinte, depois de se acomodar em sua cadeira, você resolve explorar o site do projeto. Você percebe que, além do modelo conceitual e das *user stories*, o site já tem também um arquivo compactado com código-fonte. Descompactando tudo, você percebe que eles estão organizados em dois diretórios um chamado *src* e outro chamado *test*. Você tenta compilar, mas o compilador mostra vários erros, indicando um conjunto de classes que estão faltando no sistema. Você logo percebe que precisará criar essas classes, inserindo atributos e métodos para que os testes passem.

Você corre para OO e pede a ele pra explicar aquele código. Ele explica que "os testes de unidade são para garantir que você está fazendo tudo certo, porque eu não tenho tempo de ficar metendo a mão nesse código do modelo. Afinal, você tem de fazer alguma coisa aqui, não é? Os testes estão todos no padrão JUnit, é muito fácil. E o resto você vai completando para poder finalizar o sistema. Disponibilizei também um *facade* para representar a interface do sistema. Ela substitui a interface gráfica por enquanto. Eu fiz ela pra que você pudesse trabalhar no código do modelo enquanto eu trabalho na interface". Sem muita paciência para lhe explicar muito mais, ele diz que tem "outras coisas para resolver. Tem livros de Java e de programação orientada a objetos na biblioteca, você aprende rápido". Não lhe restando muito mais a fazer, você resolver encarar o problema com toda a vontade. Afinal, "depois deste projeto, eu vou ser um programador profissional.", você pensa.

Produto:

Você deve enviar um e-mail com o produto final para o seu tutor até às 12 horas (meio-dia) do dia 04/04/2014 (sexta-feira), anexando o arquivo compactado com o código-fonte e os testes. O código fonte deve estar todo documentado utilizando o padrão javadoc.

Você deve entregar o código-fonte em um só arquivo compactado, com todas as classes que você desenvolveu para este projeto, junto com os demais arquivos que você recebeu (estão todos no arquivo folitron.zip, que está localizado na página do tutorial, em https://sites.google.com/site/mip20141/home/tutorial. Todas as classes devem estar compilando e implementando as funcionalidades adequadamente. Todos os testes devem estar executando corretamente. Todas as classes, atributos e métodos que você criou devem estar documentados utilizando o padrão javadoc. A correção será feita analizando o código, executando os testes de unidade que foram passados no arquivo dentro de folitron.zip e avaliando a documentação javadoc. É uma boa ideia você executar todos os testes antes de entregar o seu projeto.

Avaliação:

O problema entregue corresponde a 60% da nota da unidade, e o desempenho nos tutoriais corresponde a 40% da nota.

Calendário:

Aula	**Data**	**Atividade**
1	11/03	Apresentação Problema
2	14/03	PBL

3	18/03	PBL
4	21/03	PBL
5	25/03	PBL
6	28/03	PBL
7	01/04	PBL
8	04/04	Entrega Problema 1

Links Interessantes:

http://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos
http://en.wikipedia.org/wiki/Object-oriented_programming
http://www.eclipse.org