Universidade Federal de Campina Grande – UFCG
Centro de Engenharia Elétrica e Informática – CEEI
Departamento de Sistemas e Computação – DSC
Professores: Gustavo (turma1), Kyller (turma 2), Livia (turma3), Raquel (Turma 4)
Disciplina: Laboratório de Programação 2
Período: 2013.2

Laboratório 06

Neste laboratório trabalharemos desenvolvimento de sistemas através de TDD (desenvolvimento dirigido por testes). Para tal você deve implementar uma classe usando os testes desta classe previamente definidos. Além disso, será trabalhado o desenvolvimento de testes de unidade em Java usando a biblioteca JUnit4. Para começarmos a trabalhar, você deve recuperar os testes já implementados para a classe Sala.java, localizados aqui.

Instruções:

1. Data de entrega para todas as turmas: 10/12/2013 até 23:59h;
2. Data de entrega com penalidade (50% da nota): 11/12/2013 até 23:59h;
3. Crie o projeto lab06 no eclipse e programe todos os exercícios dentro do pacote lp2.lab06;
4. Salve as classes criadas em um arquivo zip chamado lab06-<seuNome>-<suaTurma>.zip (ou .tgz) e o submeta através do formulário neste link;
5. Certifique-se de que seus programas não têm erros de compilação, caso contrário eles não poderão ser corrigidos;
6. Antes da definição de cada classe escreva o seu nome em comentário (/*Aluno: <seu nome>*/);
7. Lembre de escrever o javadoc das classes criadas (não precisa para os testes).

A biblioteca JUnit

Como você já deve saber, a biblioteca JUnit permite que criemos testes de unidade para as classes desenvolvidas em Java. A criação de testes é extremamente importante pois permite que verifiquemos se as funcionalidades de uma classe ou de um conjunto de classes está correta (ou de acordo com sua especificação), garantindo que outras classes dependentes delas ou sistemas que as utilizem funcionarão corretamente. Em geral, o teste de unidade traz a especificação mais detalhada da classe que deve ser desenvolvida.

Para adicionarmos a biblioteca em um projeto no Eclipse devemos ir em: Project -> Properties -> Java Build Path -> Add Library. Abrirá uma caixa, na qual devemos selecionar JUnit e em seguida escolher JUnit4. Dessa forma o nosso projeto poderá

rodar classes de testes criadas através da biblioteca JUnit versão 4.

Parte 1: Implementado a Classe Sala.java

Vocês receberam uma fonte de informações extremamente importante para a criação de uma classe: seus testes. Através dos testes de uma classe podemos extrair toda a informação necessária para a sua implementação, como os métodos que foram implementados, suas respectivas assinaturas (tipo de retorno e parâmetros de entrada); como deve ser o funcionamento de cada método a partir dos resultados que eles retornam para os testes; que exceções devem ser tratadas; etc.

Dessa forma vocês devem implementar a classe Sala de tal forma que todos os testes implementados na classe TestaSala.Java funcionem corretamente.

Um pouco mais sobre a classe Sala:

- Uma sala representa um conjunto de espaços que podem ser ocupados por um robô ou por algum obstáculo que seja colocado.
- O tamanho da sala deve ser informado obrigatoriamente.
- Cada posição da sala pode estar livre ou ocupada.

Importante: lembre de escrever o javadoc da classe Sala. Sempre que você vai escrever uma classe é preciso escrever o seu javadoc.

Parte 2: Implementando a classe Robo.java e seus testes

Uma vez implementada a classe Sala, podemos criar novas classes que podem fazer uso do ambiente "criado" através de um objeto que representa uma nova sala. Para tanto vamos implementar uma classe Robo, que poderá se movimentar na sala através de todos os seus lugares que se encontram livres.

Sobre a classe Robo:

- O robô deve possuir uma sala que representará o ambiente onde ele pode se movimentar;
- Ao criar um robô, este deve estar na posição 0,0 da sala, se esta estiver vazia. Se a posição [0,0] estiver ocupada o robô deve ser posicionado na próxima posição livre (realizando um caminhamento linha por linha na sala). Nesse momento a posição inicial do robo deve ser indicada como ocupada na sala;
- O robô deve realizar 4 movimentos: andar para frente, andar para trás, andar para a esquerda e andar para a direita;
- A cada movimento efetivado pelo robô, a quantidade de passos dados por ele deve ser incrementada em 1. Isto significa que se o robô está na posição [0,0]

e vai andar para a direita, então ele deve ir para a posição [0,1], se esta estiver vazia. Além disso, nosso robô possui uma quantidade limitada de energia, que deve ser decrementada em 1 a cada novo movimento realizado;

- Ao ser realizado um movimento, a posição anterior onde o robô se encontrava deve ser liberada e a nova posição ocupada;

- De acordo com a localização do robô alguns movimentos não poderão ser realizados. Isso ocorre porque a posição de destino não é válida, ou porque ela já se encontra ocupada. Por exemplo, estando na posição [0,0] da sala, o robô não poderá andar para a esquerda e nem para cima;

- O robô só poderá se movimentar caso ele ainda possua energia.

Implemente a classe Robo seguindo as especificações acima e implemente os testes da classe Robo. A seguir algumas dicas de testes importantes são oferecidas. Esses testes devem ser realizados, mas outros testes também são possíveis e outros testes devem ser realizados.

- Criar um robô com energia negativa e ver que uma exceçãoo é lançada
- Criar um robô com energia zero e ver que uma exceção é lançada
- Criar um robô com uma sala null e ver que uma exceção é lançada
- Criar um robô com uma sala cheia e ver que uma exceção é lançada
- Criar um robô com energia positiva e uma sala vazia e ver que: (i) a posição inicial do robô é (0,0) e (ii) a sala não está livre na posição (0,0)
- Criar uma sala com um obstáculo em (0,0) e usar essa sala na criação de um robô (com energia adequada); verificar que a posição inicial do robô é (0,1)
- Criar um robô com energia 10 e verificar que:
    - o robô não pode subir, o robô não pode ir para a esquerda.
    - Depois de tentar fazer o robô subir e tentar fazer o robô ir para a esquerda a energia do robô deve continuar sendo 10 e robô e o número de passos dados foi zero, já que o robô de fato não se movimentou
    - Deve ser possível fazer o robô descer (verificar isso) e verificar que a posição nova do robô é (1,0)
    - Deve ser possível fazer o robô ir para a direita (verificar isso) e verificar que a posição nova do robô é (1,1)
    - Realize testes semelhantes aos acima movimentando o robô para cima e para a esquerda
- Teste movimentações com obstáculos
    - Crie um robô com energia 10 em uma sala vazia
    - Insira obstáculos nas posições (1,0) e (0,2) da sala
    - Faça o robô descer – isso não deve ser possível, já que a posição está ocupada – teste que a energia e a posição do robô não alteraram
    - Faça o robô ir para a direita – isso deve ser possível. Teste que a nova posição do robô é (0,1) e teste que o número de passos dados pelo robô foi 1 e a energia é 9.
    - Faça o robô ir para a direita e verifique que ele não se movimentou (use número de passos que deve ser 1 e energia deve ser 9)

    - Faça o robô descer. Verifique que a nova posição é (1,1), o número de passos foi 2 e a energia está em 8
    - Faça o robô ir para a esquerda e verifique que a nova posição é (1,0), a energia é 7 e numero de passos 3
    - Continue realizando testes de forma que o obstáculo (1,0) seja alcançado e impeça o robô de se movimentar para uma dada direção
- Crie um robô com uma sala vazia e faça ele se movimentar. À medida que vai se movimentando teste se o número de passos, a posição e o nível de energia estão corretos
- Você precisa testar se o robô obedece as regras quando ele está nas "quinas" da sala. Por exemplo, se uma sala tem 3 linhas e 3 colunas, teste os movimentos possíveis quando o robô está na posição (0,2), (2,0) e (2,2) (a posição 0,0 já foi testada)
- Você precisa testar que dois robôs são iguais se e somente se eles tiverem uma sala idêntica e estiverem na mesma posição da sala.

Importante: lembre de escrever o javadoc da classe Robo. Sempre que você vai escrever uma classe é preciso escrever o seu javadoc.