Instituto de Matemática e Estatística da Universidade de São Paulo

São Paulo, 1º de Dezembro de 2015

Mac0332 – Engenharia de Software

Entrega 3

Dados de Testes

Bruno Guilherme Ricci Lucas (4460596)

Leonardo Pereira Macedo (8536065)

Vinícius Bitencourt Matos (8536221)

**Testes:**

**testQuit:**

**Entrada:** É criado e inicializado um Client, que logo envia um comando para sair.

**Saída:** A saída esperada seria receber uma afirmação positiva do assertFalse, ou seja, é falso que o Client esteja conectado após o comando QUIT.

**testNick\_Invalid:**

**Entrada:** Cliente envia NICK (comando de terminal) com os seguintes parâmetros: - aa - 33444 - abc% - todos inválidos.

**Saída:** Espera-se que o nickname do cliente não mude em nenhum dos casos. Ou seja, que client.getNickname() seja null (valor para nickname indefinido) em todos os casos testados.

**testNick\_Valid:**

**Entrada:** Cliente envia NICK com o seguinte parâmetro: niceNick, um parâmetro válido.

**Saída:** Esperamos que o Cliente de fato recebe e guarde a string niceNick como seu nickname, pois ela é válida.

**testHost\_Invalid:**

**Entrada:** Um Cliente vira host de uma partida. Temos três situações disponíveis:

1) Cliente sem nickname usando um mapa que não existe. No caso os parâmetros são: NULL (nickname) e “thisIsFake” (nome do mapa).

2) Cliente sem nickname usando um mapa válido. No caso os parâmetros são: NULL e “simpleMap”.

3) Cliente com nickname válido usando um mapa inválido. No caso, os parâmetros são: “hostInvalid” (um nickname válido) e “thisIsFake”.

**Saída:** Espera-se que os testes nos digam que:

Em 1), nickname e nome de mapa de fato não valem.

Em 2), nickname não vale, mas nome do mapa vale.

Em 3), que nickname vale, mas nome do mapa não.

**testHost\_Valid:**

**Entrada:** Cliente com nickname válido vira host de uma partida usando um mapa válido. Após um tempo, ele muda para um outro mapa válido. Os parâmetros usados são: “hostValid” (nickname do host), “simpleMap” e “happyPlains” (nome dos mapas válidos).

**Saída:** Espera-se que o Cliente de fato sirva de host para os mapas escolhidos, mesmo após a mudança.

**testUnhost\_Invalid:**

**Entrada:** Há dois casos:

1. Cliente com nickname inválido e sem ser um host (não deu o comando ou não escolheu um mapa válido) tenta dar UNHOST. Neste caso, os parâmetros são nulos.
2. Cliente com nickname válido e sem ser um host tenta dar UNHOST. Neste caso, os parâmetros acabam sendo nulos, já que não há sala.

**Saída:** Espera-se que, pelo Cliente não ser host, o comando UNHOST não funcione.

**testeHost\_Valid:**

**Entrada:** Cliente com nickname válido começa a ser host de uma sala usando um mapa válido. Após um tempo, ele dá o comando unhost. Os parâmetros são: “unhostVal” (nickname) e “testmap”.

**Saída:** Espera-se que o Cliente possa se tornar host e cancelar o comando sem problemas.

**testJoin\_Valid:**

**Entrada:** Dois Cliente são conectados ao servidor com nicknames válidos. Um será o host e o outro o guest. O host cria uma sala usando um mapa válido e o guest entra ela. Os parâmetros são: “hostJoin” e “guestJoin” (nomes do host e do guest) e “testmap” (nome do mapa). Em seguida, eles desconectam.

**Saída:** Espera-se que o servidor reconheça que os dois Clientes estão em jogo antes de eles se desconectarem.

**testSurrender:**

**Entrada:** Dois Clientes, um host e um guest, se conectam ao servidor e uma sala é criada. O jogo começa e o host (primeiro a jogar) dá o comando SURRENDER para desistir da partida. Logo após os dois se desconectam. Os parâmetros são “hostSur” e “guestSur” como nicknames do host e do guest e “testmap” como mapa.

**Saída:** Espera-se que os dois Clientes sejam reconhecidos como “isPlaying” até o momento em que o host dá o comando surrender. Na averiguação subsequente, espera-se que nem o host nem o guest sejam reconhecidos como “isPlaying”.

**testEndTurn:**

**Entrada:** Um Cliente é criado e vira um host. É verificado o turno (-1). Então um guest entra na sala e o jogo começa. É verificado o turno (1-turno do host). O host dá o comando endTurn. É verificado o turno (2-turno do guest). O guest dá o comando endTurn. É verificado o turno (1-turno do host). Os parâmetros são “hostEnd”, “guestEnd” e “testmap”. Além dos já mencionados -1, 1 e 2 para as comparações de turno.

**Saída:** Espera-se que de fato os turnos mudem da forma descrita acima.

**testMove\_Valid:**

**Entrada:** Um host e um guest estão em jogo. O host manda na localização 0 0 se mover para a localização 1 0. Após uma pausa ele tenta movê-la de novo.

**Saída:** Espera-se que de fato o host consiga mover a unidade da primeira vez, mas que ele falhe na segunda tentativa.

**testMove\_Invalid:**

**Entrada:** Durante uma partida, um host tenta mover uma unidade usando diversos comando inválidos, seja tentando mover uma unidade em uma posição inválida, seja tentando colocar uma unidade em uma posição inválida, ou partindo de uma posição inválida para tentar posicionar em outra posição inválida. Também é testado posicionar uma unidade em uma tile ocupada, ou posicionar em um lugar que extrapole o movimento da unidade selecionada, assim como tentar mover uma unidade inimiga.

**Saída:** Espera-se que nenhum dos comandos testados sejam aceitos, já que são todos inválidos.

**testAttack\_Valid:**

**Entrada:** Durante uma partida, o host comanda uma unidade para atacar um inimigo. É dada uma pausa e o host tenta mover e atacar com a mesma unidade novamente. Os parâmetros são “hostAtkVl”, “guestAtkVl” e “testmap”.

**Saída:** Espera-se que a unidade selecionada ataque com sucesso na primeira vez. Em seguida, é esperado que o host falha em tentar movimentar ela e atacar com ela.

**testAttack\_Invalid:**

**Entrada:** Durante uma partida, o host tenta atacar. São testadas diversas coordenadas inválidas, tiles vazias como sendo alvo, atacar uma unidade além do range da unidade sendo usada, atacar usando uma unidade inimiga e atacar uma unidade aliada. Os parâmetros são “hostAtkIn”, “guestAtkIn” e “testmap”.

**Saída:** Espera-se que nenhuma das entradas testadas seja bem-sucedida, já que são inválidas.

**testUnitsNextTurn:**

**Entrada:** Durante uma partida, um host movimenta uma unidade e ataca uma unidade inimiga, encerrando seu turno em seguida. O guest então encerra seu turno, passando a vez para o host, que tenta mover uma unidade e atacar usando ela. Os parâmetros são “hostCtrl”, “guestCtrl” e “testmap”.

**Saída:** Espera-se que o host possa usar suas unidades normalmente após o guest encerrar seu turno.