

2°. Trabalho – Tratamento de Exceção

Valor: 2 pontos

Data de entrega: 22 de outubro

Equipe: até 3 pessoas

Existe uma linguagem de programação chamada Logo que é muito usada no Brasil em escolas de primeiro e segundo grau. Nesta linguagem você pode programar os movimentos na tela de um personagem, como uma tartaruga. Este personagem descreve movimentos riscando ou não a tela. Imagine que o personagem move-se no eixo cartesiano (x,y) e que seu objetivo é alcançar sua comida que está previamente definida em alguma posição do eixo. O personagem não pode se mover nas regiões cujas coordenadas são negativas. Com base nessa descrição, faça o que se pede:

- a) Crie uma exceção chamada `MovimentoInvalidoException` que informe na mensagem qual movimento foi inválido
- b) Crie uma classe `Robo` que represente esse personagem contendo dois atributos representando sua posição no eixo cartesiano e uma cor que o identifica. Crie um construtor que recebe a cor do robô e o inicialize na posição (0,0). Crie também métodos de `get` e `set` para as posições.
- c) Crie um método `mover`, que recebe como parâmetro uma `String` e altera a posição do robô da seguinte forma:
 - “up” move o robô no eixo y em uma posição acima.
 - “down” move o robô no eixo y em uma posição abaixo.
 - “right” move o robô no eixo x em uma posição para a direita.
 - “left” move o robô no eixo x em uma posição para a esquerda.

Caso o movimento faça com que o robô entre numa zona negativa (x ou y menor que 0), lance a exceção da questão anterior e não permita o movimento. Após cada movimento, mostre a posição do robô.
- d) Sobrecarregue o método `mover`, mas ao invés de receber uma `String`, ele rece como parâmetro um inteiro de 1 a 4, onde 1 representa “up”, 2 representa “down”, 3 representa “right” e 4 representa “left”
- e) Um método que verifique se o robô encontrou o alimento (está na mesma posição) e retorne um boolean

1) Crie uma classe `Main` que instancie um robô, peça ao usuário para determinar a posição do alimento, e peça ao usuário para ficar movendo o robô até ele encontrar o alimento – não esqueça de tratar a exceção.

2) Crie outra classe `Main` que instancie dois robôs, peça ao usuário para entrar com a posição do alimento, e faça os dois robôs se moverem randomicamente, um de cada vez,

até que um deles encontre o alimento. Ao final, mostre quem achou o alimento e o número de movimentos válidos e inválidos de cada robô.

3) Crie uma subclasse RoboInteligente que sobrescreve o método mover de forma que se robô fez um movimento inválido, garanta que o próximo movimento não será o mesmo. Neste caso, se o robô fez um movimento inválido, ele realiza outros movimentos até que seja feito um movimento válido.

Crie uma classe Main que instancie um robô normal e outro inteligente, peça ao usuário para entrar com a posição do alimento, e faça os dois robôs se moverem randomicamente, um de cada vez, até que **ambos** encontrem o alimento. Ao final, mostre o número de movimentos que cada robô fez para encontrar o alimento.

Para todos os itens anteriores, mostre os robôs se movendo na em uma matriz que representa a área de locomoção. Mostre também o alimento na posição indicada pelo usuário. Considere a área fixa de um quadrado com 4 unidades de lado.

4) Crie uma classe abstrata Osbtaculo que possui como atributo um id e o método abstrato *bater*. Crie pelo menos duas classes concretas de obstáculos, Bomba e Rocha, sendo que na primeira o método *bater* implica em o robô que encostou nela explode (desaparece do jogo ou não anda mais), enquanto na segunda o método *bater* faz o robô voltar para a posição anterior. Ao explodir a bomba desaparece do tabuleiro.

Crie uma classe Main que instancie um robô normal e outro inteligente, peça ao usuário para entrar com a posição do alimento e inserir algumas bombas e rochas no tabuleiro, e faça os dois robôs se moverem randomicamente, um de cada vez, até que **um** deles encontre o alimento ou ambos explodam. Ao final, mostre o número de movimentos que cada robô fez para encontrar o alimento ou até explodir.

Sugestões:

- Verificar uso de cores para colocar nos robôs.
- Use mecanismos para retardar a execução de forma que seja possível ver os robôs se mexendo.

Desafio:

- Use interface gráfica (Swing ou JavaFX) para tornar seu jogo mais agradável!