**Desenvolvimento SnakeGame**

*Nível de dificuldade: Médio*

O desafio propõe o desenvolvimento de um SnakeGame básico que fornece as

seguintes funcionalidades:

* A Cobra pode se mover em uma determinada direção e quando come a

comida, o comprimento da cobra aumenta.

* Quando a cobra se cruza ou quando um dos cantos da tela é atingido pela cobra o jogo acaba.
* Os alimentos serão gerados de em posição aleatória.
* todos estes elementos de funcionalidades devem ser apresentadas numa tela com interface de usuário.

Antes de tudo, é necessário pensar nas classes. As principais classes

serão - considerando que estas são as classes-base da aplicação:

1. Snake

2. Cell

3. Board

4. Game

*Obs.: deve-se pensar em criar uma Enum para definir um CellType – por*

*exemplo: se a célula esta vazia, com comida ou, ainda, se é um snake node(um*

*pedaço do corpo da cobra);*

*Também é importante identificar, com comentários, qual é o classificação e*

*funcionalidade de cada bloco de código.*

*Customização da tela do jogo – usando de cores e fontes diferentes do padrão*

*– será considerado um grande diferencial.*

**Dicas:**

A classe *Game* representa o corpo do programa. Ele armazena informações

sobre a cobra e o tabuleiro por onde o personagem principal circula.

A classe *Cell* representa o único ponto de exibição/quadradinho. Ele contém o numero

de linhas e colunas e as informações.

A classe *Snake*, que contém o corpo e a cabeça da cobra.

Pode-se usar o recurso LinkedList para armazenar o corpo porque, dessa forma,

fica mais fácil adicionar uma célula em O(1).

O método *Grow* será chamado quando ele -o personagem - comer o alimento.

A classe Board representa o display (a tela do jogo). É uma matriz de

células. Possui um método de geração de alimentos que gera

o alimento em uma posição aleatória.

A classe principal (Game) mantém as instâncias de Snake e Board. Seu método

*“update”* precisa ser chamado em um intervalo fixo.

Obs.: a estruturas de classe do seu projeto não precisam seguir estritamente a indicação de nomes e quantidades de classes indicadas acima. Se for necessário, cada dupla pode nomear as classes da forma que lhe for conveniente. Também, se preciso for, crie a quantidade de classes necessárias para que o projeto funcione adequadamente.

é recomendado o uso das classes JFrame e JPanel - superclasses oferecidas pelo proprio Java - e praticar a herança (quando necessário) para gerar gráficos e interfaces para a aplicação:

*"JFrame é uma classe java que é estendida pela classe Frame. O JFrame é tratado como a janela principal. No JFrame, diferentes elementos, como rótulos, campos de texto e botões, podem ser adicionados. Esses elementos no JFrame criam a interface gráfica do usuário.O JFrame também é conhecido como contêiner de nível superior Swing. Swing refere-se ao kit de ferramentas do widget da GUI para criar aplicativos em JAVA. "*

- exemplo de herança com JFrame

**public** **class** MinhaClasse **extends** JFrame

*"O JPanel é uma classe de contêiner leve e mais simples que faz parte do pacote java.swing. Ele pode agrupar ou armazenar um conjunto de componentes, principalmente para criar uma interface com o usuário. É semelhante ao painel no Abstract Window Toolkit (AWT). O JPanel não contém borda, barra de título ou barra de menus. Embora existam vários gerenciadores de layout, o FlowLayout é o gerenciador de layout padrão do JPanel e herda a classe JComponents. Se um componente precisar ser anexado a um aplicativo, o JPanel fornecerá espaço."*

- exemplo de herança com JPanel

**public** **class** UmaOutraClasse **extends** JPanel

Alguns resursos que são indicados para importação e uso:

**import** java.awt.Color;

**import** java.awt.BorderLayout;

Referencias:

<https://www.geeksforgeeks.org/design-snake-game/>

<https://massivetechinterview.blogspot.com/2015/10/snake-game-design.html>

<https://pt.education-wiki.com/4283046-jpanel-in-java>

<https://pt.education-wiki.com/4171594-jframe-in-java>