2021

# Snake
# Documentation Technique

Amstutz Thomas

DIVTEC

25/01/2021

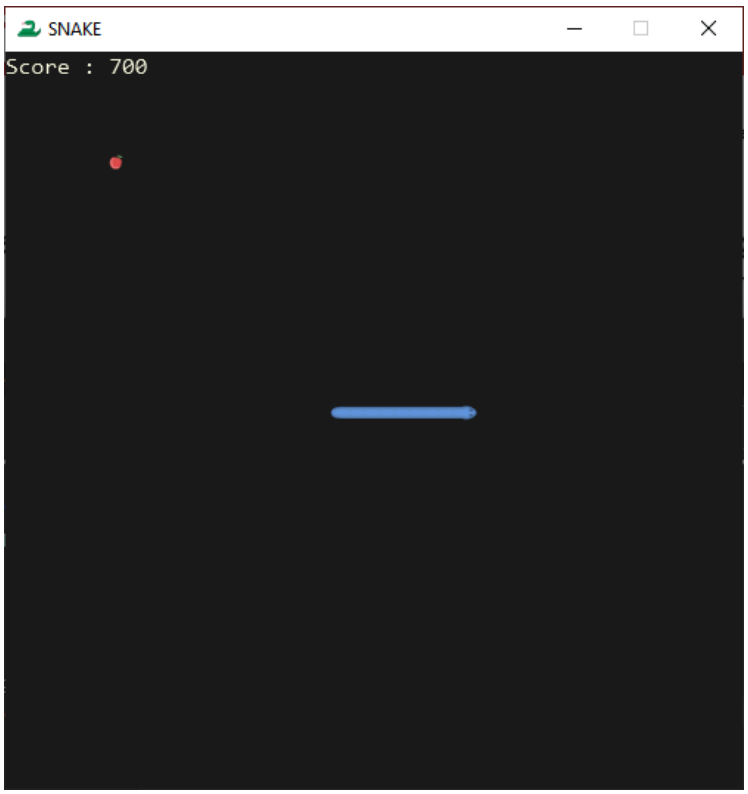
# Table des matières

# 1 Description

Dans le cadre de l'atelier 31, tous les apprentis informaticiens de 3ème année doivent créer un jeu vidéo 2D en C++ ou en Java.

J'ai choisi de mon côté de réaliser un Snake en Java, jouable seul ou à deux. L'environnement de développement que j'ai utilisé est IntelliJ en version 2020.3. Pour générer la documentation technique du code, j'utilise Javadoc.

Le genre de jeu Snake est surtout connu grâce à Nokia qui l'a intégré dans ses téléphones depuis 1998.



# 2 Convention de nommage

## 2.1 Variables

Toutes les variables sont nommées en anglais et commencent par une minuscule. Quand une variable est composée de plusieurs mots, on utilise le camelCase.

Exemple de variable :

```
    int myInt = 42;
```

## 2.2 Constantes

Comme les variables et les constantes sont en anglais, elles sont toutes en majuscule et des underscores séparent les différents mots.

Exemple de constante :

```
    private final int SNAKE_SPEED = 18;
```

## 2.3 Fonctions

Les fonctions ressemblent aux variables, elles sont donc en anglais et utilisent le camelCase.

Exemple de fonction :

```
    private void startGame();
```

## 2.4 Enumérés

Les types énumérés ont deux conventions de nommage : une pour le nom et une pour les valeurs de l'énuméré.

### 2.4.1 Nom

Les noms des types énumérés sont en anglais et commencent par une majuscule. Quand il y a plusieurs mots, on utilise le camelCase.

### 2.4.2 Valeurs

Les valeurs des énumérés sont en anglais et s'écrivent tous en majuscule et les différents mots sont séparés par des underscores.

Exemple d'énuméré :

```
public enum Direction {
    UP,
    DOWN,
    LEFT,
    RIGHT,
    NO_DIRECTION
}
```

# 3 Classes

Le programme est séparé en plusieurs classes. Voici un diagramme de classe simplifié qui les présente. La version complète est dans le dossier "/res" sur Github.

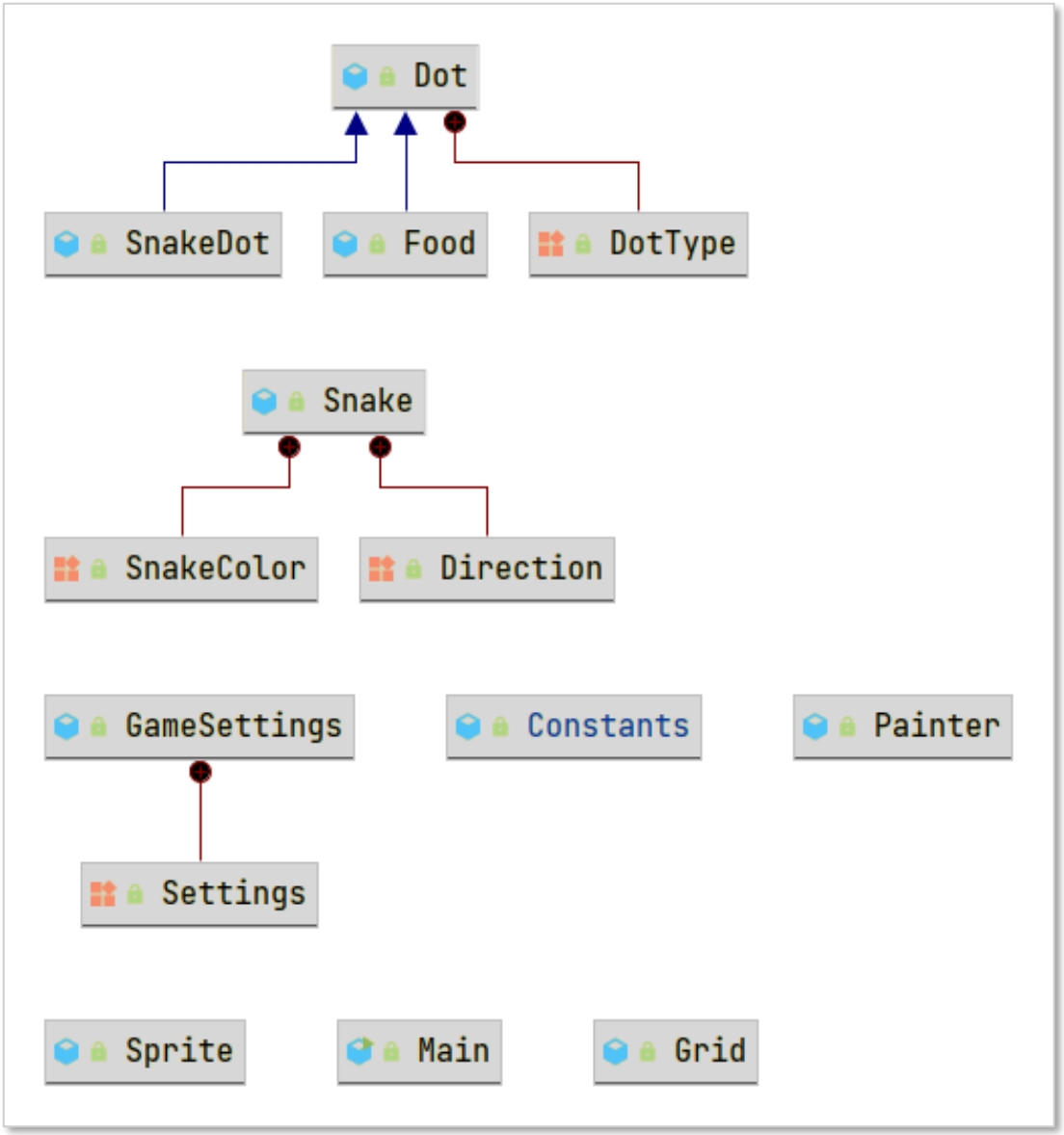



*Diagramme de classe simplifié*

Voici une description des classes :

- **Dot** : Représente un point sur le plateau de jeu
  - **DotType** : Enum ; genre de points possible
- **SnakeDot** : Hérite de Dot ; représente un point appartenant au serpent
- **Food** : Hérite de Dot ; représente une pomme
- **Snake** : Représente un serpent
  - **SnakeColor** : Enum ; couleurs qu'un serpent peut avoir
  - **Direction** : Enum ; directions qu'un serpent peut avoir

- **GameSettings** : Gère les paramètres du jeu
    - **Settings** : Enum ; contient tous les paramètres et leur valeur
- **Constants** : Regroupe toutes les constantes du jeu souvent utilisées
- **Painter** : Affiche tous les éléments dans la fenêtre
- **Sprite** : Hérite de la classe Image ; Image avec une taille fixe
- **Main** : Point d'entrée du programme ; C'est depuis là que le jeu est contrôlé
- **Grid** : Représente le plateau de jeu où se déplacent les serpents


*Utilisation des différentes classes en jeu*

# 4 Explications supplémentaires

## 4.1 La classe Painter

La classe `Painter` contient tout l'affichage du jeu. Elle est appelée à chaque passage de la boucle de jeu avec la méthode `paintGame()`. Cette méthode va effacer tout le contenu actuel de la fenêtre et le remplacer par de nouvelles valeurs, que ce soit le score, une pomme ou un serpent.

D'autres méthodes de la classe `Painter` permettent d'afficher le menu principal, le menu d'options et le menu de fin de jeu.

### 4.1.1 Fonctionnement

Quand le jeu est lancé, le `Painter` est initialisé avec la méthode `initPainter()` à laquelle on passe un `GraphicsContext` en paramètre. Ensuite, le menu principal s'affiche avec la méthode `paintMenu()`. Le joueur peut aller dans les paramètres du jeu, ce qui appelle la méthode `selectOption()` (qui prend en paramètre l'option sélectionnée) qui elle-même appelle `paintConfigMenu()` qui va afficher le menu.

Si le joueur lance une partie, la méthode `paintGame()` sera appelée. Cette méthode va s'occuper d'afficher l'interface en appelant toutes les méthodes nécessaires :

- `initGrid()` qui efface le contenu précédent
- `paintDot()` qui dessine des carrés
- `paintSnake()` qui dessine les serpents
- `paintCommands()` qui affiche les commandes du jeu avant le début de la partie
- `paintGameOverMenu()` qui affiche le menu de fin de partie
- `paintScore()` qui affiche le score du / des joueur(s)

Quand le joueur met la partie sur pause, la méthode `paintPause()` est appelée. Celle-ci affiche le texte « Pause » et montre les commandes du jeu avec `paintCommands()`.