Snake Game

Gameplay:

The player controls a snake. The objective is to collect as many points as possible, this is done by eating red apples. If a player eats a yellow apple their snake is reversed in direction (the head becomes the tail and vice versa). If the player eats a black apple the snake dies.

Core features:

1. **FPS clock (Featured)**
   Demo: the snake would move with a consistent speed of space per second. Implementation dependent on Swing convention.
2. **Score counter (Featured)**
   Demo: In the top right corner there is a score counter. Upon eating a red or yellow apple the score is increased. Upon death the score counter is reset and starts from 0.
3. **Arrow key controls (Featured)**
   Demo: The snake changes direction depending on the corresponding arrow key pressed.
4. **Tutorial Panel (Featured)**
   Demo: At the first start of the game, there would be a tutorial panel. This panel disappears and the game starts, when the player presses any key. The panel will have a brief explanation of how the game works – use arrow keys to move; press P to pause; press H to open the high score; what the apples do.
5. **Hit detection on objects (Featured)**
   Demo: If a snake eats a red apple, it will increase its size and score by one;
   If it eats a yellow apple, it will increase its size and score by one, but it also invert its direction (the rear will become the head, and vice versa);
   If it eats a black apple, it will end the gameplay, and a death screen will show up.
6. **Hit detection on self (Featured)**
   Demo: If the snake collides with itself, it will die.
7. **Hit detection on game border (Featured)**
   Demo: if the snake collides with a border, it will die
8. **Random apples (Featured)**
   Demo: The game always starts with one red apple in a specific space. Whenever an apple is eaten another will spawn. The black apples change color after certain time and thus can become either red or yellow. The apples spawn under the following prerequisites:
   1. Algorithm iterates through each point from (0,0) to (10,10) (as specified by grid size in the config), and adds each point which is *not* occupied by the snake to a HashSet representing all the available spawning spaces,

2. Each point which is currently occupied by an apple is also removed from the hashset,
3. If the apple spawning method is set to strict (so strict = true) , if there amount of available spaces is less than specified in the config (currently 8) then the apple will not spawn. This is done to prevent possibly game-ending apples (eg. black apples) from spawning when they might not be avoidable,
4. If there are more than 3 available spaces (amount of ideal available spaces around the snake's head) then every point around the head is also removed from the hashset,
5. A random index from the hashset is chosen, and an apple is spawned at that point

9. **State-by-state updates (Featured)**

   Demo: every time before the snake moves a space, the program will check if the snake will collide with anything (itself, border, apple) if so, it will calculate the appropriate outcome and on the next frame the effect will take place
10. **Increase length (Featured)**

    Demo: When a red or a yellow apple is eaten by the snake, the length of the snake is increased by adding a cell to snake on the rear end.

High priority:

1. **Input queueing (Featured)**

   Demo: the player can put up to 4 different directions in queuing. Meaning the snake would remember them and execute each one of them on the next available space. If the amount of elements is smaller or equal to the max queue size (4), the head of the queue (so the first added element) is removed, and the next one is added.
2. **Pause menu (Featured)**

   Demo: In the top right corner, there is a button with a pause icon, which when pressed pauses the game and shows a pause panel. Upon repress it will un-pause the game show the original game panel.
3. **Animation (Removed)**

   Demo: 'half-step' between cells: instead of the snake teleporting from one cell to the next, it does a half-step to be between the two cells, and then fully occupies the next cell. Inputs are accepted between each half-step.

Low priority:

1. **Death tip messages (Featured)**

   Demo: When a player dies, a death panel will show up. This includes a death tip message, which is chosen randomly from a pool, depending on how the

player died. Example: if the player died by hitting one of the borders, the game could show the following text *"Try avoiding borders."*; if the player eats a black apple, it could show : *"The color black does usually mean death."; etc.*
The panel also includes a retry and an exit button.

2. **High score board (Removed)**
   Demo: whenever the player dies it will allow them to see their high score. If the new score that they have achieved is higher than the previous high score, the high score will be automatically updated.
3. **Sound effects (Removed)**
   Demo: whenever a collision is executed a sound effect will accompany it. This provides for a smoother user experience
4. **Volume (Removed)**
   Demo: enables the user to increase or decrease the sound of the game
5. **'Users' for high score (Removed)**
   Demo: the user will have to enter their name in order to "log in". This enables them to keep track of their own personal high score. Thus, multiple players can use the game and have different high scores.
6. **Un-pause timer (Removed)**
   Demo: when the player un-pauses the game, there will be a short timer, before the game starts. This enables the user to be prepared when to start playing.
7. **Settings button and a menu (Removed)**
   Demo: it will have the volume settings there + optional switch of control from arrow key binds to WASD

Topics of Choice:

1. **Version control**
   - Done via GitHub
   - Branches for main/dev + possibly for features
   - Sensible commits based on features
2. **Game development**
   - Clocks
   - Game Design; add sources for why these features improve the game experience
     - Rendering sprites
     - Animations
     - Key-bind overlay
     - Controls
       - 'P' for pause
       - 'R' for retry
     - Information on what apples do what

        - Have the player figure out what the apples do, which would be based on their color (ex: red apples would imply that they are good; black apples would imply they are bad)
        - Players would then receive information about the apples they have discovered throughout their gameplay at the beginning of they new attempt
    - Feedback system
        - Death tip messages