UMLdiagram for hangman game
(at least 5 use cases)

Actors
Player:: involved in guessing letters to uncover the hidden word.
Game System: Manages the game, provides the word to guess, keeps track of game state, and validates guesses.

Use Cases:
Start Game: starts a new game
Guess Letter: allows the player to guess the hidden letter.
Display Progress: shows the progress of the word, and also displays correctly guessed letters.
End Game:concludes the game to show if you won, lost or quit.

USE CASE 1: Start Game
PRECONDITIONS:
The game application is launched
The system is ready to start the game
POSTCONDITIONS:
A new game is ready for a randomly selected word to be guessed.
The player is told to start guessing.
Scenarios:
Game loop gets activated either by pressing a certain key or just by starting the game.

USE CASE 2: Guess Letter
PRECONDITIONS:
A game session is running
The player has not reached max incorrect guesses
POSTCONDITIONS:
Guessed letter is checked
If letter is present it is put into the word spaces
If not correct the system keeps count of failed attempts.
Scenarios:
Separate function that checks if a letter is available in word picked. If not, add to the hangman, else add the letter to the display word. Class Word. Word has functions that check characters against string. Returns true or false.

USE CASE 3: Display progress
PRECONDISOINs:
The game is currently running
The player has made more than one guess.
POSTCONTIONS:
The state of the word with correct guesses is displayed

The hangman display is updated with incorrect guesses.

Scenarios:

Begin printing when you start the print hangman and progress of guesses. Class Display, display the progress, by printing stuff out.

USE CASE 4: End Game

PRECONDITIONS:

The game is in two states, won, lost, or the player has quit.

POSTCONSTIONS:

The session has ended.

USE CASE 5: View High Scores

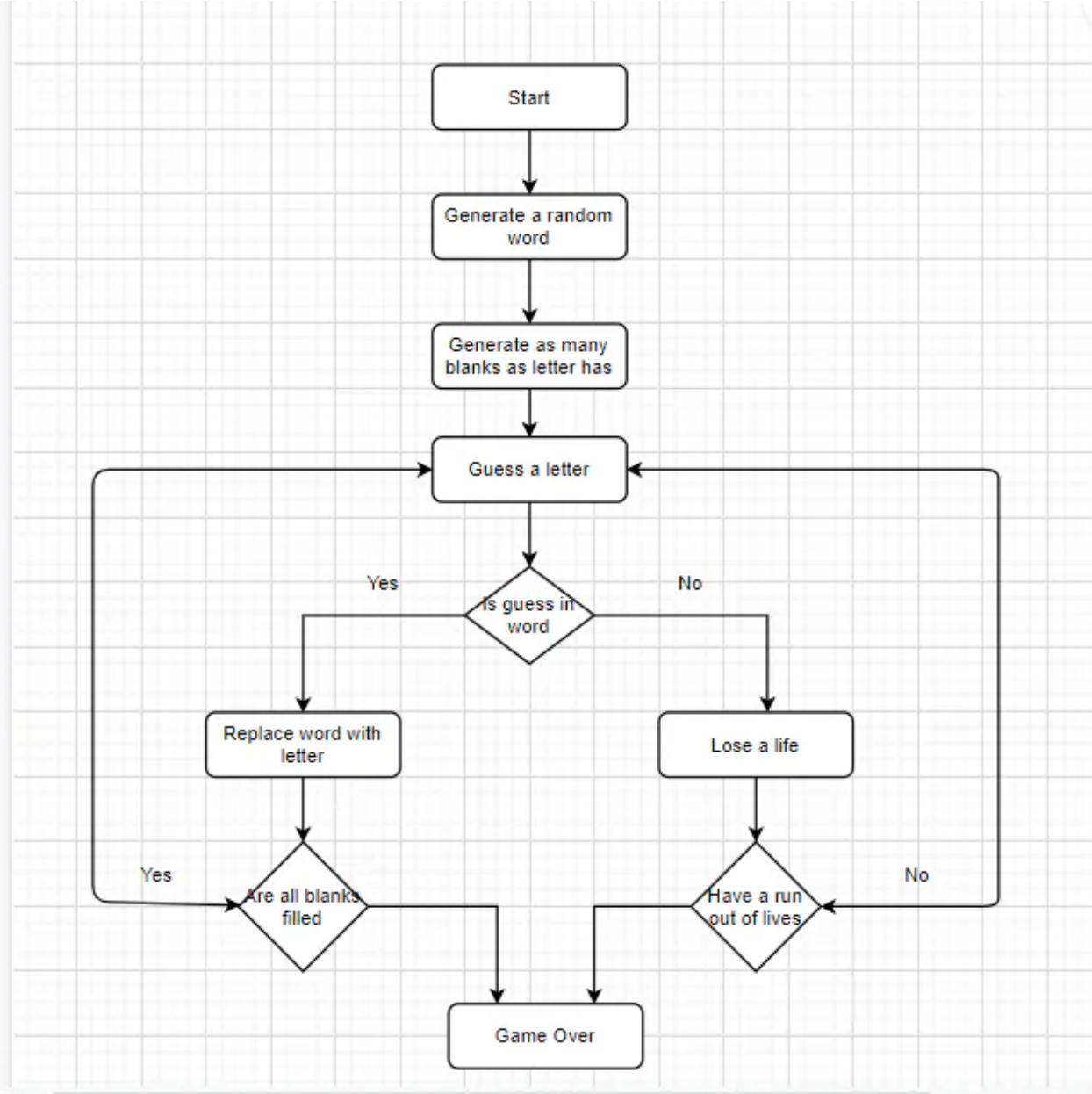
PRECONDITIONS:

The game has stored high scores

The player selects the choice to view the high scores.

POSTCONDITIONS:

The high scores are displayed to each of the player



1. **HangmanGame Class:**
    - Attributes:
        - `secretWord: String` (private): The word to be guessed by the player.
        - `guessedWord: StringBuilder` (private): The current state of the word as guessed by the player.
        - `maxAttempts: int` (private): The maximum number of incorrect guesses allowed.
        - `currentAttempts: int` (private): The current number of incorrect guesses made.

        - `guessedLetters: Set<Character>` (private): Set of letters guessed by the player.
    - Methods:
        - `HangmanGame(secretWord: String, maxAttempts: int)` (public): Constructor to initialize the game with a secret word and maximum attempts.
        - `guessLetter(letter: char)` (public): Method to guess a letter. Updates guessedWord, guessedLetters, and currentAttempts.
        - `isGameOver(): boolean` (public): Method to check if the game is over (either won or lost).
        - `isWordGuessed(): boolean` (public): Method to check if the word has been guessed correctly.
        - `isLetterAlreadyGuessed(letter: char): boolean` (public): Method to check if a letter has already been guessed.
        - `printGameStatus(): void` (public): Method to print the current state of the game (e.g., guessedWord, remaining attempts, guessed letters).
        - getCurrentAttemps(): int(public) get the current number of attempts used
2. **Base Loop Class:**
    - Attributes: None
    - Methods:
        - `main(args: String[]): void` (public static): The entry point of the Hangman game. It initializes a HangmanGame object, takes user input for guessing letters, and manages the game loop until it's over.
3. **Player Class:**
    - Attributes:
        - `name: String` (private): The name of the player.
        - `score: int` (private): The player's score.
    - Methods:
        - `Player(name: String)` (public): Constructor to initialize a player with a name.
        - `getName(): String` (public): Method to get the player's name.
        - `getScore(): int` (public): Method to get the player's score.
        - `incrementScore(points: int)`: Method to increment the player's score by a certain number of points.

4. WordBank Class:

- Attributes
    - words : String Array(private)
- Methods
    - WordBank Contructor

- getRandomWord(): String(public) method gets a random word from the String Array words.