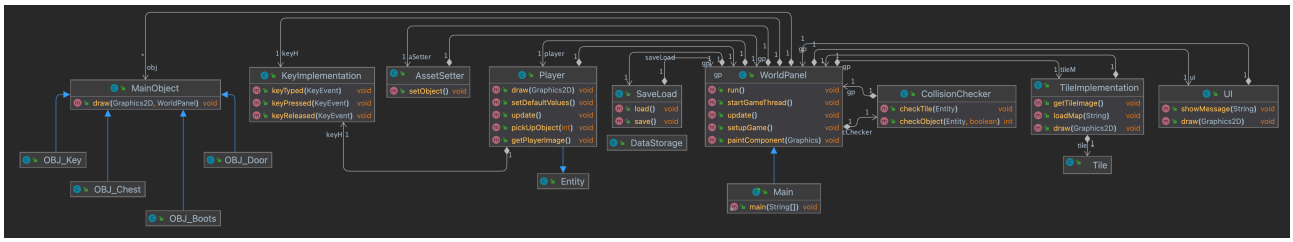
Description of the program

Information

UML Diagram



Class Explanation

.JSON

There is the **Main Class** where is going to be a final implementation of the project and an installation of the .json file (hereinafter referred to as a level.json).

The level.json file manages the starting position of the player, player inventory, enemy positions, and locations of A's and B's (special coins).

Example of the level.json:

```
{
{
  "gameOverTimeInSeconds": 60,
  "monsters": [
    {
      "worldX": 22,
      "worldY": 26,
      "name": "Youtube",
      "life": 4
    },
    {
      "worldX": 26,
```

```
      "worldY": 23,
      "name": "Youtube",
      "life": 4
    },
    {
      "worldX": 20,
      "worldY": 29,
      "name": "Professor",
      "life": 10
    }
  ],
  "inventory": [
    {
      "name": "Pen"
    },
    {
      "name": "Pen"
    },
    {
      "name": "Boots"
    },
    {
      "name": "Pen"
    },
    {
      "name": "Key"
    }
  ],
  "objects": [
    {
      "name": "Pen",
      "worldX": 24,
      "worldY": 23
    },
    {
      "name": "Boots",
      "worldX": 24,
      "worldY": 20
    },
    {
      "name": "Pen",
      "worldX": 30,
      "worldY": 23
    },
    {
      "name": "Key",
      "worldX": 20,
      "worldY": 22
    }
  ],
  "player": {
    "worldX": 28,
    "worldY": 18,
    "lifes": 6
  },
  "elapsedTime": 12000000000
}
```

Packages

- **java**
  Contains all the classes of the game.

- **resources**
  Contains all the game textures, including objects, player sprites, maps, etc. in .png/.txt format.

Player Class:

- Represents the player entity in a game. It handles the player's movement, collision detection, and sprite animation. It receives input from a `KeyImplementation` object to determine the player's direction. The class also keeps track of the player's level and life. The `Player` class is responsible for updating and drawing the player on the screen.

Entity:

- Manages game entities, including their position, movement, collision, sprites, and text. It has properties for position, speed, direction, sprite images, collision area, and dialogue. Key methods include `update()` for updating actions and collision detection, `speak()` for displaying text on a screen, and `draw()` for rendering the entity's sprite.

KeyImplementation Class:

- Is responsible for handling keyboard input events in a game. It keeps track of the state of certain keys (up, down, left, right, enter) being pressed or released. Based on the game's current state, it performs different actions when specific keys are pressed or released. This class allows for player interaction by translating keyboard input into game actions.

WorldPanel Class:

- Represents the main panel of the game. It handles the game's screen settings, world settings, and game states. The class contains various components such as
  the `TileImplementation`, `KeyImplementation`, `CollisionChecker`, and `UI`. It

also manages entities including the player, objects, and monsters using arrays. The class runs the game loop in a separate thread and updates and draws the game components accordingly.

TileImplementation Class

- Manages the tiles in the game. It loads tile images, sets up tile properties, and draws the tiles on the screen based on the game map and player's position.

SaveLoad Class:

- Saves player stats, inventory, and world data.

UI Class:

- Manages the game's user interface elements and game states. It handles displaying messages, drawing the player's life, and managing different game states such as play, pause, and text.

EventImplementation:

- Is responsible for managing events and interactions within the game. It handles collision detection between the player and event rectangles, triggers specific actions based on the collision, and updates the game state accordingly. The class also includes methods to handle specific events, such as damage tile.

Main:

- Serves as the entry point for the game. It creates an instance of the `JFrame` window, sets up the game panel, and starts the game thread. The main method is responsible for initializing the game and launching the window.