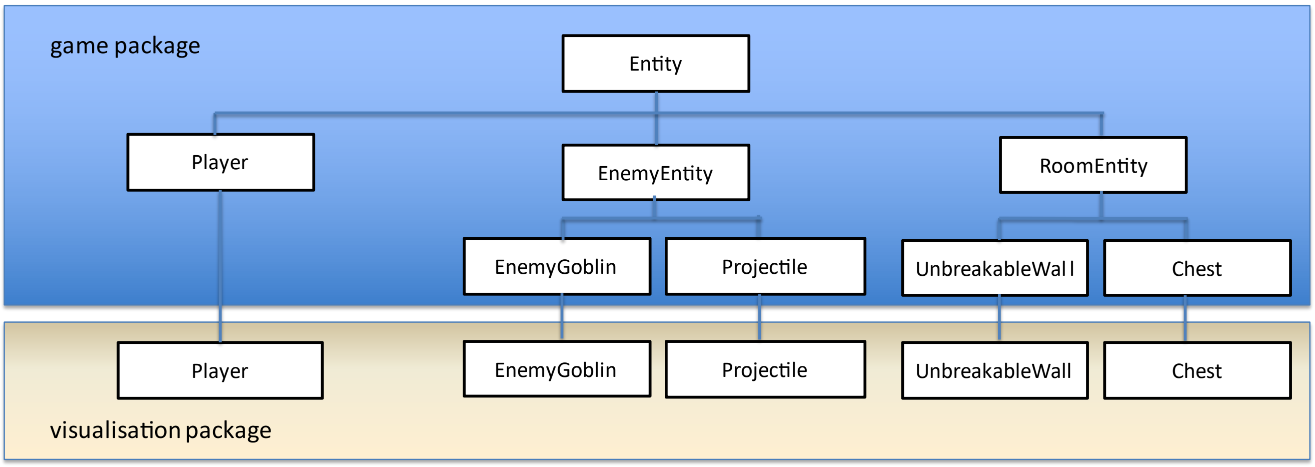
Java Project 2024-2025

Goal of this project:

Create a 2D Top-Down Dungeon Crawler game in Java and follow these guidelines:

Game logic:

- Provide a clear **separation between game logic and game presentation**. Do this by encapsulating all game objects and logic, except their presentation and user-interaction, in a separate **game package** (e.g.: be.uantwerpen.fti.ei.geavanceerde.dungeoncrawler).
  The goal of this separation is to provide a very simple way of writing a completely new visual presentation based on the same logic and structure. So you might have two completely different looking games, based on the same game logic package.
- Design a hierarchy of game entities (obstacles, bonuses, projectiles, etc.) and their interactions (collision control, position control, etc.). Be as creative & fancy as you like. A basic entity class hierarchy might be, for instance, implemented as in this figure:



Specifications:

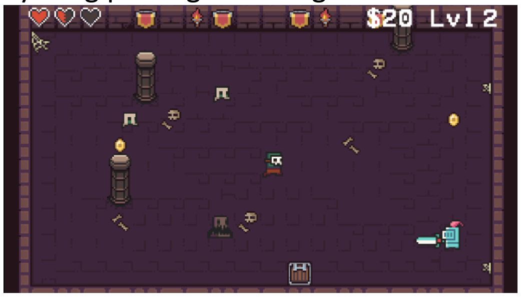
- Use one single object (might be a “singleton” design pattern) to represent **the game**: a self-contained object with a list of game’s entities, a main loop to check for collisions, interactions and states, a facility to keep scores, creation and deletion of game’s entities depending on the needs, etc.
- A “game object” **must not** contain any calls to visualization libraries, interactive keyboard control, etc.
- Use the **“abstract factory”** pattern to create entities inside the game object to achieve the separation between game logic and presentation.
- Create a simple way to time events: a basic “Stopwatch” class for keeping time (in milliseconds) between two “ticks” might be a good idea. (Because not every computer runs at the same speed and yet, the visualized entities should move with the same speed on all computers.)
- The game field is a 2D world which is independent from the visualization (i.e. different coordinate systems!). Use a coordinate system with ‘Double’-type coordinates for your game. The “visible” game world is a window that is shown on the screen.
- A score and health indication should be visible during the game.
- Your game should contain **at least one system** designed in a **Data-Oriented** fashion, using an **Entity-Component-System architecture**.
- In the game logic, the usage of ‘null’ values needs to be **avoided**.
- **At least one** useful implementation of the **Java Streams API** should be implemented.

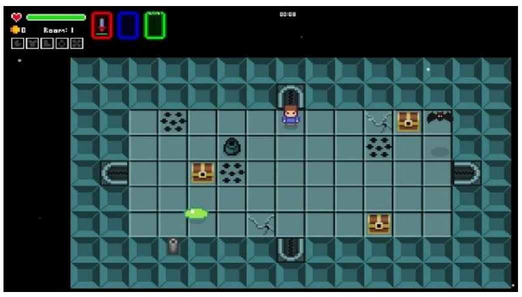
- Your game logic should include **at least one game script** that is written in **Lua** (e.g. enemy AI, physics, player movements, game events, etc.).
- A configuration file with game parameters should be included (e.g. window size, selected level, lives, etc.).
- The game should be playable on other systems **without any code modifications**.
- The player character starts in a room/maze in which he/she needs to navigate from room to room. The game ends when the player finds the exit of the maze, or is out of life points. This exit is locked until the player finds the key or clears the objective of the room. Each room crawls with enemies that needs to be avoided or destroyed in order to navigate the maze. The player can elevate its score by destroying enemies or clearing objectives. Players can be locked in a room until all enemies are destroyed or when the key has been found to open the door. Different type of objects can be present in the room (e.g. unbreakable walls, chests, keys, pits, spikes, etc.) that blocks player movement and/or the player can interact with. Enemies will try to hit the player by touch or with projectiles to lower the player's life points. The behavior of the enemies should happen in a possible non-trivial way (use your imagination but don't spend hours on "artificial intelligence", this is not the right course. Inspiration can be found in existing examples of the game on the internet). Bonuses could be placed in the room to increase or decrease the players' health, gain powerful weapons, gain movement speed, obtain invincibility to enemies, etc. The goal is to exit the level as fast as possible (reward points) or within a predefined time duration.
  A simple description can be found on Wikipedia: https://en.wikipedia.org/wiki/Dungeon_crawl
  Create your own version of the game with at least two levels (configurable or predefined). Inspiration can be found in existing examples of games on the internet.

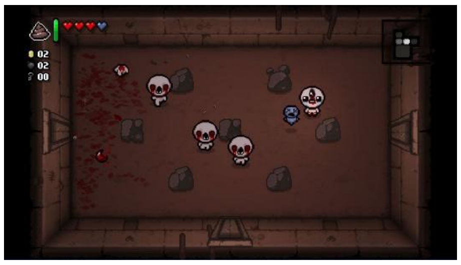
Visual & interactive implementation:

- Use the Java2d library (java.awt.Graphics2D) in your "visual" Implementation of the game's objects. Use the Java event library for "interactive" keyboard input (java.awt.KeyEvent, etc.).
  Some information can be found at:
    - https://docs.oracle.com/javase/tutorial/2d/index.html
    - https://www.iitk.ac.in/esc101/05Aug/tutorial/2d/index.html
    - https://zetcode.com/gfx/java2d/

  Use separate packages to encapsulate the "visual" and "interactive" implementations of your game objects (e.g.: J2d).
- In your visual implementation, you can use your own pictures or you can find some online. Restrict yourself to a 2D representation. You can use transparency or animations.
- Everything put together might look like:







Practical:

- A basic **working** implementation of the package with basic game logic (and more importantly: following clear design considerations and required design patterns, i.e. abstract factory and singleton) & visual representation in Java2D are required to get a passing grade, e.g. proper code commenting & documentation (javadoc) of your API, logical structure of the code & code base (directories), proper class construction that proves you “understand” Java are obviously implied! Including a configuration file (see xml or Properties library java.util.Properties) and Lua game script.
- Extensions are **up to you**:
    - More enemies or other enemies
    - Different game-modes
    - Extra levels (stored in files or generated)
    - More advanced interactions
    - Advanced power-ups
    - Moving obstacles
    - Collectables
    - ...
- A portfolio that logs the usage of generated code from LLMs in your project is mandatory.
- Good luck! If you have any questions, remarks or comments feel free to ask during course hours.