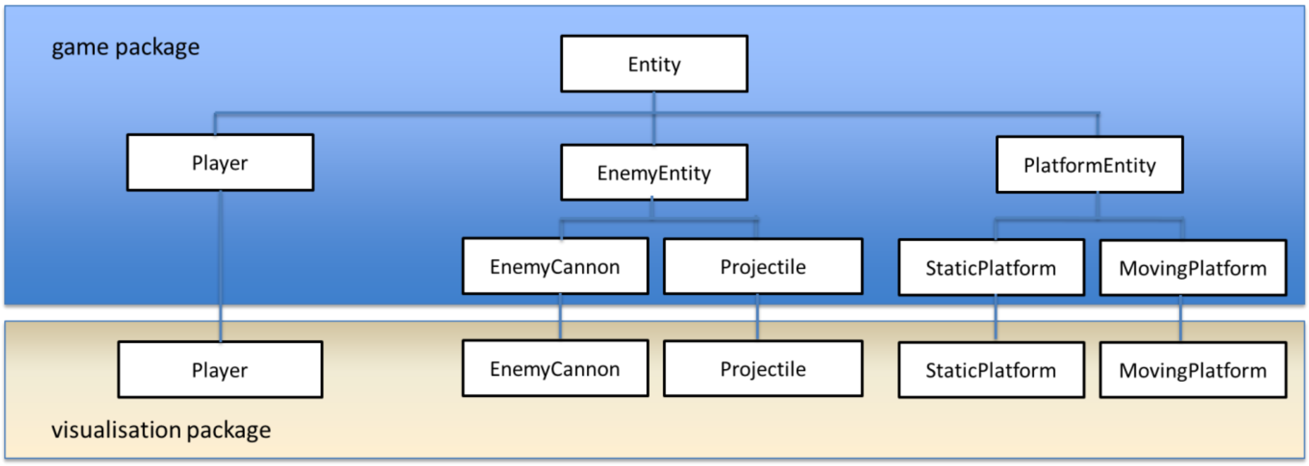
# Java Project 2023-2024

## Goal of this project:

Create a 2D Vertical Platform Jumping game (e.g. Doodle Jump) 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.jumpinggame).
  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 package.
- Design a hierarchy of game entities (platforms, bonuses, projectiles, ...) 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:



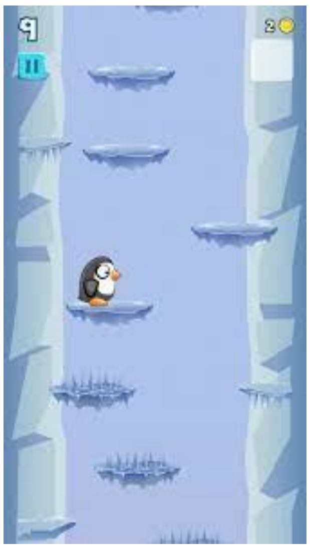
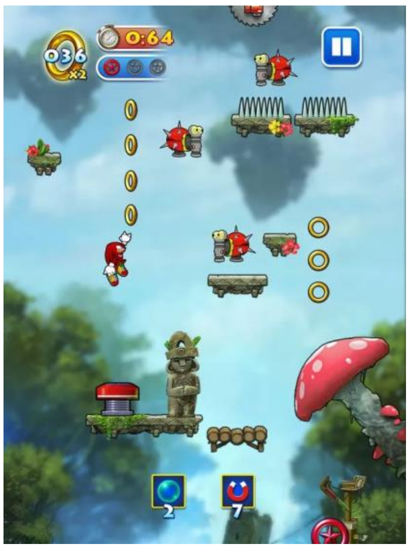
## Specifications:

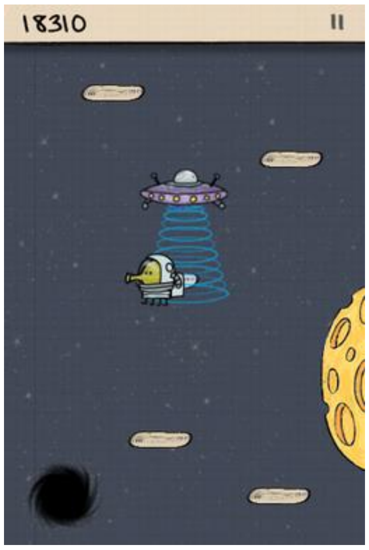
- Use one single object (might be a “singleton” design pattern) to represent **the game**: a self-contained object with a list of game’s entities, a main loop to check for collisions, interactions and states, a facility to keep scores, creation and deletion of game’s entities depending on the needs, etc.
- This “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.
- 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 has 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 (different coordinate systems!). 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**.
- Your game logic should include **at least one game script** that is written in **Lua** (e.g. enemy AI, jump physics, platform movements, etc.).
- The player character starts on the ground floor and needs to jump onto platforms to get as high as possible in an infinite vertical scrolling game. The game ends when the player falls out of the

edges of the screen. The player gets a score equal to the highest vertical point reached (+ optional bonus points). Platforms should be generated so that the player can advance in the game. Different type of platforms are possible in the game (e.g. static, moving, breaking when jumped on, etc.). Enemies will try to push you down or off platforms. The behaviour when the player reaches the sides of the screen can be decided by you (e.g. warping around – player travels to the other side of the screen, or player will fall down and lose the game – side walls could be put in place for the first X meters, etc.). All of this 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 on platforms and increase or decrease the players health, additional jump range, jump platforms, invincibility to enemies, etc. The goal is to get as high as possible without falling down. A simple description can be found on Wikipedia: https://en.wikipedia.org/wiki/Doodle_Jump
**Create your own version of the game with at least two levels.** Inspiration can be found in existing examples of the game 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) & visual representation in Java2D are sufficient to get a passing grade. (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 platforms
    - Collectables
    - ...
- Good luck! If you might have any questions, remarks or comments feel free to ask during course hours.