- Functionality:
    - In terms of functionality, the keyListener and mouseListener function well to restrict the player from messing up the game and instead provide the player with an enjoyable experience.
- Design:
    - Whenever we encountered a problem, we discussed what we wanted the output to be and backtracked our way to a solution. We also wrote out the solution on paper first before coding it. The most troubling aspect of creating the game was randomly generating the room. In order to do this, we decided to generate the first platform randomly and use its position to generate the following ones. It was also a process of trial and error to determine the minimum and maximum amount of platforms in a room. After the platforms, the enemies, collectibles, and the position of the button and door were also randomly determined.
- Creativity:
    - We think the theme of the game is a rather new idea. The graphics were also all drawn through Piskel. We think the most unique aspect of the game is the dead and alive concept, which is possible because of the theme. It allows for playability and encourages engagement with the game.
- Sophistication:
    - The amount of time spent on this project is significantly greater than a usual assignment/lab. There were many classes needed to create the world, some to simplify matters and some to generalize elements. The system to randomly generate rooms through a seed and computation is the most sophisticated aspect of this project.
- Broadness (at least 6):
    1. Java libraries that we haven't seen in class
        a. ImageIO
    2. Subclassing ("extends" a class you created)
        a. Main extends JPanel
    3. Interfaces ("implements" an interface you created)
        a. Main implements keyListener and mouseListener
    4. User-defined data structures
        a. Linked list (rooms)
    5. Built-in data structures
        a. Arraylists
    6. File input/output
    7. Randomization
        a. World class: Random RNG
    8. Generics
- Code Quality:

- Most of the code is sectioned into their respective method, constructor, member variables, and imports. The code quality is something we could have improved upon. However, the variable names, camelCase, and structure of the code are quite consistent. Commenting and explaining certain aspects of the code would have definitely helped others understand the code much better.