Part 1:

1. What will your program look like at the end?
    a. This program is our own take on a video game combining both aspects of the platformer and roguelike genres. The theme of the game revolves around Schrodinger's Cat thought experiment, where the player plays as a cat named "Milton" trapped inside of a cardboard box. The player will have to traverse through randomly generated rooms while in constant danger from poisonous gas filling each room, along with enemies trying to kill them at every turn. One unique aspect of this game is the ability to switch between a state of dead/alive, both having their own advantages and disadvantages. As the player continues their epic adventure, they may find extra lives as a way to recover. The ultimate goal of the game is to survive through fighting enemies and reaching the next room.
2. How will the user input work? (e.g., clicking, keys on the keyboard, specifying a file, what have you)
    a. Keys on the keyboard
3. How will the program respond?
    a. 'A' - left
    b. 'D' - right
    c. ' ' - jump
    d. 'J' - attack
    e. 'K' - switch state (dead/alive)
4. What purpose does it serve? (e.g., is a game, a productivity tool, a screen saver?)
    a. Game

Part 2:

List of Classes: Cat Class, Enemy Class, Projectile Class, Platform Class, Pair Class, Hitbox Class, Gas Class, Room Class, World Class, Collectable Class, Button Class, Door Class, Main Class, Runner Class

5. Describe the semantics and use of the class. What does it represent? When your program is run, does one instance exist? or a few? or many?
    a. Main Class - used to run World
        i. Runner Class - used to update World
    b. Cat Class - used for the main character that the player plays as; one instance exists only
    c. Room Class - used to represent the different areas the player can move to; many instances
        i. Enemy Class - used to represent the 'dust bunnies' that attack the player; many instances may be created depending on room type
        ii. Platform Class - used to represent the solid objects the player and enemies can be on top of; many instances may be created depending on the room type
        iii. Gas Class - used to represent the danger zone that is moving up by sections on the screen in every room; only one instance per room
        iv. Collectable Class - used to represent health recovery items; multiple instances created according to room
        v. Button Class - used to represent a button in the room activating ventilation (removal of gas) and access to the next room; one instance per room
        vi. Door Class - used to represent the door the player came from, and the player can access; one or more instance per room
    d. Projectile Class - used to represent attacks available to the player or enemy
    e. World Class - used to represent the world that encompasses all of the elements of the game, including all of the rooms; one instance
    f. Pair Class - used to represent the position of every component in the room; many instances
    g. Hitbox Class - used to represent the hitbox of each component for boundaries and damage; many instances
6. What are the member variables (names and types)? What does each represent semantically? Why are they public/private? Why are they the type they are?
    a. Main Class;
        - Public static final int WIDTH
            - Never-changing width of the game screen
        - Public static final int HEIGHT
            - Never-changing height of the game screen
        - Public static final int FPS

        - Never-changing update speed of the game
    - Private static JFrame frame
        - Never-changing window to view the game
    - Private BufferedImage startScreen
        - Does not need to be accessed by other classes
    - World world
        - An instance of a world class that can be accessed globally in all methods and other classes

b. Cat Class:
    - Public double lives
        - The lifespan of the cat
    - Public boolean transformState
        - The dead/alive state of the cat. Value determines when it can switch states
    - Public boolean isTransformed
        - Value helps determine attack and damage.
    - Public double transformStateCD
        - Stores the time when the cat switched states to determine the cooldown time
    - Public int timer
        - Value helps determine when the cat can switch states
    - Public boolean attackState
        - Value helps determine whether the cat can attack
    - Public double attackStateCD
        - Stores the time when the cat attacked to prevent spamming of attacks. Activates a cooldown period.
    - Public int attackRange
        - Value determines how far the attack goes
    - Public int score
        - Keeps track of score throughout the game, public to be accessed in other classes to print on end screen.
    - Public Pair catPosition
        - Stores x and y value of the cat's position. Accessed by hitbox class
    - Public Pair catVelocity
        - Stores x and y values of the cat's velocity
    - Public Pair catAcceleration
        - Stores x and y values of cat's acceleration when jumping
    - Public Pair catDimensions
        - Stores x and y value of the cat size. Accessed by hitbox class

- Private BufferedImage ___
    - The images do not have to be accessed by other classes.
- Public Hitbox catHitbox
    - A hitbox class for the cat to determine it’s boundaries and if it receives damage
- Public String orientation
    - Keeps track of which way the cat is facing

c. Hitbox Class:
- Public int hitboxWidth
    - Width of an element
- Public int hitboxHeight
    - Height of an element
- Public int hitboxTop
    - Position of the top of the element
- Public int hitboxLeft
    - Position of the left side of the element
- Public int hitboxRight
    - Position of the right side of the element
- Public in hitBoxBot
    - Position of the bottom of the element
- Public int buffer
    - Smooths collision

d. Pair Class:
- Public double x
- Public double y
    - Accessed by many classes and stores the x and y values of selected elements

e. Room Class:
- Public ArrayList<Platform> platforms
    - Keeps an arraylist of all of the platforms generated
- Public ArrayList<Door> doors
    - Keeps an arraylist of all of the doors generated
- Public ArrayList<Projectile> projectiles
    - Keeps an arraylist of all of the projectiles currently in the room
- Public ArrayList<Enemy> enemies
    - Keeps an arraylist of all of the enemies generated
- Public Button button
    - Creates a button for a room
- Public Gas gas
    - Generates gas for the room

- Public Room prev
- Public Room next1
- Public Room next2
- Private BufferedImage __
    - Images of the wall and gas level. Do not have to be accessed by other classes
- Public int difficulty
    - Helps determine how many enemies to spawn
- Public boolean isRoomZero
    - Helps differentiate the first room

i. Enemy Class:
    - Public int xPos
        - X-value position of enemy
    - Public int yPos
        - Y-value position of enemy
    - Private BufferedImage __
        - Images for the enemy
    - Public Hitbox enemyHitbox
        - Hitbox to determine damage from an attack
    - Public String orientation
        - Determine which way the enemy faces when generated
    - Public int health
        - Determine when enemies should disappear

ii. Platform Class:
    - Public Pair platformDimensions
        - Stores the length, width, and height of the platform
    - Public Pair platformPosition
        - Stores the location of the platform in the room
    - Public Hitbox platformHitbox
        - Indicates where the cat can and cannot be
    - Public int repeat
        - Determines how many times the platform image should be repeated
    - Private BufferedImage platform
        - Holds the image of the platform

iii. Gas Class:
    - Public int height
        - Indicates where the gas is
    - Public int lineHeight
        - Indicates how high the gas will rise

- Private BufferedImage __
    - Holds the images of the gas and line
- Public boolean exists
    - Indicates whether the button has been pressed or not
- Public int counter
    - Increments of time that help indicate when the gas should rise

iv. Collectable Class:

- Public Pair collPosition
    - X and Y of the collectable
- Public Pair collDimensions
    - Length, width, and height of collectable
- Public Hitbox collHitbox
    - Hitbox for when the player collides with collectable
- Public BufferedImage collSprite
    - Image of the collectable
- Public boolean used
    - Determine when to draw the collectable

v. Door Class:

- Public Pair doorPosition
    - Stores the x and y values of the position of the door
- Public int doorType
    - Indicates where the door is in the room
- Private BufferedImage __
    - Holds the images of the open and closed door
- Public Pair doorDimensions
    - Stores the length, width, and height of the door
- Public boolean unlocked
    - Statement that shows whether the button has been pressed and which door to draw
- Public Hitbox doorHitbox
    - Hitbox to determine whether the cat entered/exited the room

vi. Button Class:

- Public Pair buttonPosition
    - Stores the x and y value of the location of the button
- Public boolean pressed
    - Helps determine which image to draw
- Public BufferedImage __
    - Stores the image of the button

- Public Hitbox buttonHitbox
  - Hitbox to determine whether the cat “pressed” the button

vii. Projectile Class:
  - Public Hitbox projHitbox
    - Hitbox to determine whether the projectile hit player/enemy
  - Public BufferedImage projSprite
    - Image of the projectile
  - Public Pair projVelocity
    - Stores x and y value of the projectile’s velocity
  - Public Pair projPosition
    - Stores x and y value of the projectile’s position
  - Public Pair projDimensions
    - Stores length, width, and height of projectile
  - Public int projRange
    - Indicates how far the projectile can go
  - Public Pair initProjPosition
    - Stores the x and y values of where the projectile was first generated
  - Public boolean hitSomething
    - States whether to disappear or not

f. World Class:
  - Public int width
    - The width of the game
  - Public int height
    - The height of the game
  - Public double time
    - The time since the game began
  - Public boolean status
    - Whether the game has ended
  - Private BufferedImage gameOver
    - Image of the game over screen
  - Public Room firstRoom
    - Generates the first room type
  - Public Room currentRoom
    - Creates a new room
  - Public Cat player
    - The player. Needs to be accessed by other classes
  - Public ArrayList<Integer> roomTypes
    - Stores the different kinds of room

        - Public final Random RNG
            - Pseudorandom number that helps generate the rooms

7. What the constructor(s)? If there is only one, why? If there is more than one, why? and how do they differ?
    a. Public Cat()
        i. Only one instance made
    b. Public Room(Random RNG, int prevRooms)
        i. Public Room(int roomType)
            1. Beginner difficulty to let the player learn how to play
    c. Public Enemy(int x, int y, String orientation)
    d. Public Door(double x, double y, int type)
    e. Public Button(double x, double y)
    f. Public Platform(int x, int y, int repeat)
    g. Public Gas()
    h. Public Collectable(int x, int y)
    i. Public Projectile(Pair pos, Pair vel, BufferedImage sprite, int range)
    j. Public World(int width, int height)
    k. Public Main()
    l. Public Hitbox(Pair dimensions, Pair position)
    m. Public Pair(double initX, double initY)
    n. One for everything else because those are the only specifications the constructor absolutely needs
8. What are the methods (return type, name, argument types)? What do they do (in words, not code)? Why are they in public/private/static/not-static? Are they recursive?
    a. Cat Class & Enemy Class:
        i. Public void update(World w, double time)
            - Updates position of cat/enemy based on velocity, etc.
        ii. Public void shoot()
            - Creates projectile at the position of the entity
    b. Gas Class:
        i. Public void reset()
            - New room -> restart at given level
        ii. Public void vanish()
            - Button pressed -> disappear
    c. World Class:
    d.
        i. Public void updateWorld(double time)
            - Changes the position of all objects on screen based on their velocity (calls all objects' update methods if they have one)
        ii. Public void drawWorld(Graphics g)

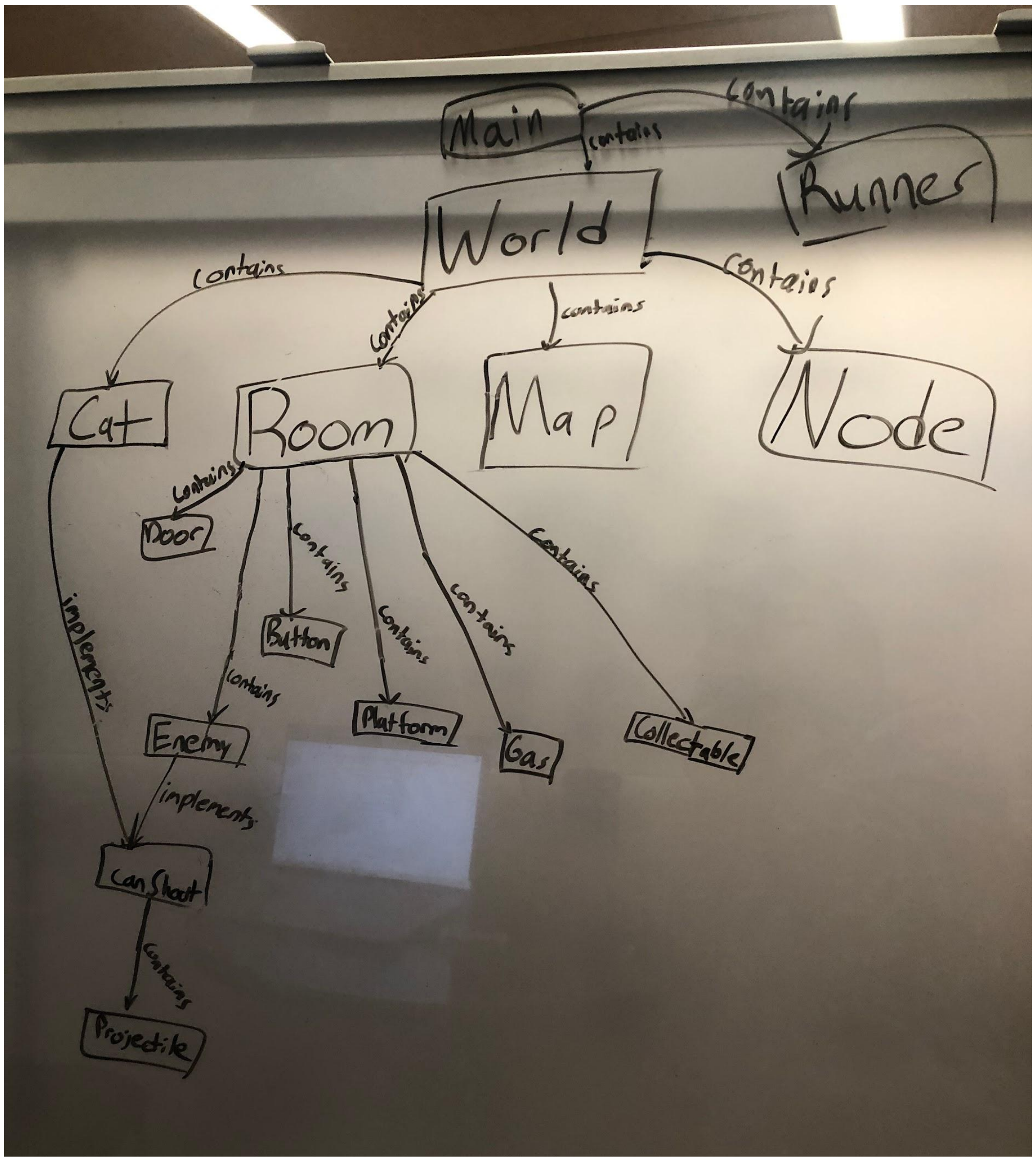
- Calls all objects' draw methods

e. Main Class:
    i. Public void keyPressed(KeyEvent e)
        - (public void keyReleased(KeyEvent e) & public void keyTyped(KeyEvent e))
        - Controls character movements or player access to menu/map
    ii. Public void mousePressed(MouseEvent e)
        - Starts game
    iii. Public void paintComponent(Graphics g)
        - Draws everything in world

f. Runner Class:
    i. Public void run()
        - Updates the world by calling World's update method

g. All Classes:
    i. Public void draw(Graphics g)
        1. Draws the object onto the screen

9. Provide a description and/or diagram of how your classes interact with each other.


Main
contains
Contains
Runner
World
contains
Contains
contains
contains
Cat
Room
Map
Node
contains
Door
contains
contains
contains
contains
implements
Button
contains
Enemy
Platform
Gas
Collectable
implements
CanShoot
contains
Projectile