Provide a list of the classes you will define.
For each:

1. 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?
2. What are the member variables (names and types)?
    1. What does each represent semantically?
    2. Why are they public/private?
    3. Why are they the type they are?
3. What the constructor(s)?
    1. If there is only one, why?
    2. If there is more than one, why? and how do they differ?
4. What are the methods (return type, name, argument types)?
    1. What do they do (in words, not code)?
    2. Why are they public/private/static/not-static?
    3. Are they recursive?

Classes

1. Game
    a. Main class, runs the game
    b. Run this file to run the game
    c. Data Members
        i. level
            1. static int
            2. keeps track of what level the player is currently on
        ii. player
            1. static instance of class Player
        iii. BOX_WIDTH/BOX_HEIGHT
            1. Dimensions of the jframe
            2. Public static final int
                a. Can be accessed by any class, but shouldn't be changed at any time
        iv. bkgrnd, door, door1
            1. Color variables
            2. static(only one instance of "Game")

3. determine color to use to draw different background components

v. haxorLand
    1. boolean
    2. Tells us if we are in “haxorLand” (changed physics) or not

vi. player
    1. static of class Player
    2. instance of player class that gets used as main character

vii. playerPosX and playerPosY
    1. 2 different ints
    2. Track player’s position on the screen

viii. enemies
    1. static ArrayList
    2. takes in generic <Enemy>
    3. stores enemies to print out

** This ArrayList was a solution to a problem we were stuck on for a while. The ArrayList stores the enemies that we want to print out on each level. We chose to create an ArrayList because it allows us to easily add and take away enemies for each level. ArrayLists also allowed us to manipulate how each type of enemy moves and interacts, while always being stored in one data structure..**

ix. haxItemCount
    1. static int
    2. counts how many items the player has collected

x. part
    1. static Item
    2. instance of Item that gets printed

xi. endGame
    1. boolean
    2. ends the game when player reaches the end

xii. gAmt
    1. int
    2. changing velocity for gravity physics

d. Constructors

i. One constructor
    1. Sets haxorLand equal to false
    2. calls loadLevel method corresponding to player's current level
    3. Defines new thread and runs
        a. continuous runner, runs throughout game

e. Methods
    i. Main
        1. creates the GUI
        2. adds KeyListeners
        3. runs the game
    ii. loadLevel 1, 2, 3, 4
        1. public static void
        2. all separate methods
        3. each one loads a different level in the game
    iii. loadEndScreen
        1. public static void
        2. loads end screen of the game
    iv. loadHaxorLand
        1. public static void
        2. if player has collected all 3 items, sets boolean haxorLand to true
    v. paintComponent
        1. public void
        2. takes in Graphics
        3. paints the background
        4. repainted every loop, changes player and enemy position on screen
    vi. Runner
        1. class
        2. implements Runnable
            a. run
                i. public void (returns nothing)
                ii. continuously loops through and updates positions of player and enemies

iii. checks for players touching items, enemies, walls
iv. advances game to new levels

2. Enemy
    a. Different types of enemies faced by the player
    b. Data Members
        i. Type
            1. Int
            2. Type of enemy
        ii. enemX/Y
            1. x and y position of the enemy
            2. public b/c changed by other classes
        iii. i
            1. Image
            2. used with t to determine what image to use
        iv. t
            1. Toolkit
            2. used with i to determine what image to use
        v. dir
            1. boolean
            2. initialized to true
            3. changes enemy movement direction
        vi. enHeight and enWidth
            1. public ints
            2. rough dimensions for enemy hitboxes
    c. Constructors
        i. one constructor
            1. takes in 5 ints that set enemy type and size
    d. Methods
        i. changeDirec
            1. public void
            2. changes dir to true or false depending on amount of iterations of the while(true) loop in Runner

ii. updatePos
    1. public void
    2. changes enemy movement depending on type

3. Item
    a. Data Members
        i. Type
            1. Int
            2. Gives type of item
        ii. x/yPos
            1. public int
            2. gives the x and y position of the item on the screen
            3. doesn't change
        iii. itHeight/itWidth
            1. public int
            2. determines size of item hitboxes
        iv. i
            1. Image
            2. used with t to draw images on screen
        v. t
            1. Toolkit
            2. used with i to draw images on screen
    b. Constructors
        i. one constructor
            1. takes in an int item type, x/yPos, itWidth/Height
            2. depending on type prints out the item at a certain location
4. Player
    a. The character that the user controls
    b. Data Members
        i. image
            1. Image
            2. Player image used for the main character
        ii. playerMovingLeft, Right, Up, and Down

1. 4 separate static booleans
2. initialized to false
3. cause movement while true

iii. jump

1. Boolean
2. false
3. should become true when player is jumping

iv. pWidth and pHeight

1. public static ints
2. set to 80
3. set player's hitbox

c. Constructors

i. one constructor

1. sets player image and size

d. Methods

i. checkTouchEnem

1. public static boolean - returns boolean
2. returns true if player is touching enemy
3. takes in Enemy
4. checks if player hitbox collides with enemy or not

ii. checkTouchItem

1. public static boolean - returns boolean
2. returns true if player is touching item
3. takes in Item
4. checks if player hitbox collides with item or not

iii. checkTouchWall

1. public static boolean - returns boolean
2. returns true if player is touching wall
3. takes in level
4. checks if player hitbox collides with wall or not

iv. getImage

1. public Image

2. gets an image
3. returns image

v. startMove
1. public static void
2. changes playerMoving boolean to true depending on if user pressed w, a, s, or d

vi. stopMove
1. public static void
2. changes playerMoving boolean to false depending on if user let go of w, a, s, or d

vii. movePlayer
1. public static void
2. moves the player depending on which playerMoving booleans are true

5. World
   a. Constructors
      i. one constructor
         1. placeholder if world is ever instantiated
   b. Methods
      i. polyX1, X2, X3, X4, X5 and polyY1, Y2, Y3, Y4, Y5
         1. public static int[]
         2. takes in int level
         3. returns int array
         4. fills array with numbers depending on level
         5. array numbers will be used to draw polygons onto screen
      ii. numPoints1, numPoints2, numPoints3, numPoints4, numPoints5
         1. public static int
         2. takes in int level
         3. returns int
         4. assigns int number of points in polygon depending on level