Team Zoms: Ryan Greene, Jack Napor, Richard Huffman, Danny Toback

CSCI 205

From a technical viewpoint, our system has one “master” class Board, which has three helpers, DrawBoard, DrawBullet, and DrawZombies, which put everything onto the board. Everything on the board is a Piece. Piece is one abstract parent class with many child classes such as Wall, Hole, and Bullet; as well as classes with classes that extend from it, such as how Player extends from abstract class Character, which extends from abstract class Piece. From there, the ZombieView, and ZombieController graphically interpret the Board, and show it to the user visually. The ShopController and ShopView, both interact with the ZombieController and ZombieView, making it the more essential of the two MVC set ups in our program.

User Stories:

-As a user, I can move my player throughout the board

    -I can see my player rotate when it changes directions

    -I can control which direction, and when my player moves

-As a user, I can attack a zombie

    -I can kill zombies by depleting their health

    -I can fire bullets from the direction I am facing

    -I will be chased by zombies that know my location

-As a user, I can buy items from the store in between rounds

    -I will gain points from killing zombies

-I will be able to upgrade my armor, to defend against zombies

-I will be able to upgrade my gun, to deal more damage to zombies

-As a user, I can see all of my player's important information

-I can check the round

-I can check how many zombies I've killed

-I can see how much ammo I have

-I can see how much health I have

-I can see how many points I have

-As a user, I can see a neatly designed board and user interface

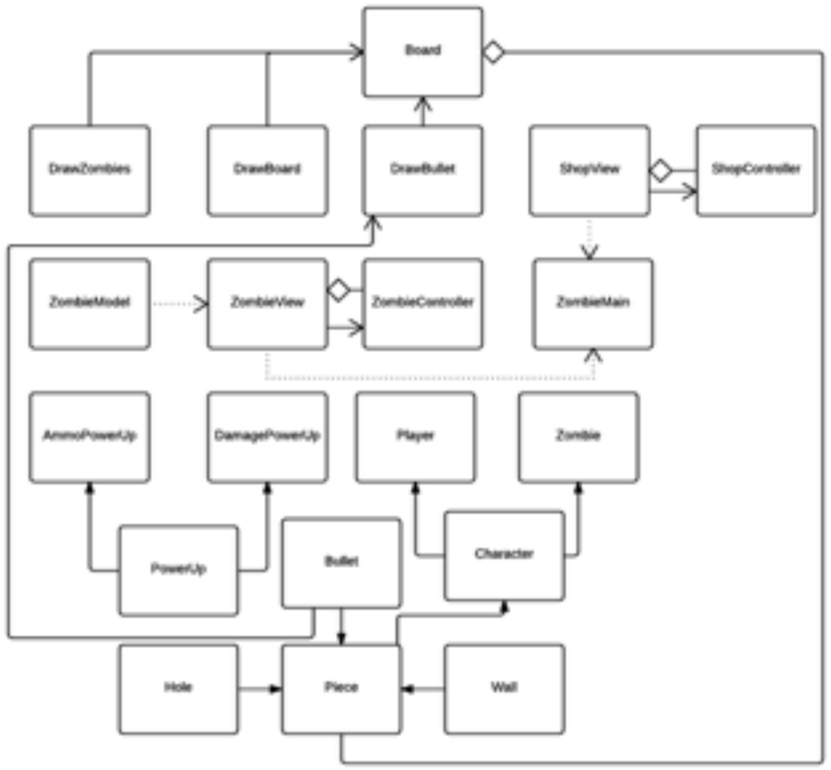
-Design the board

-Create the buttons, and functionality

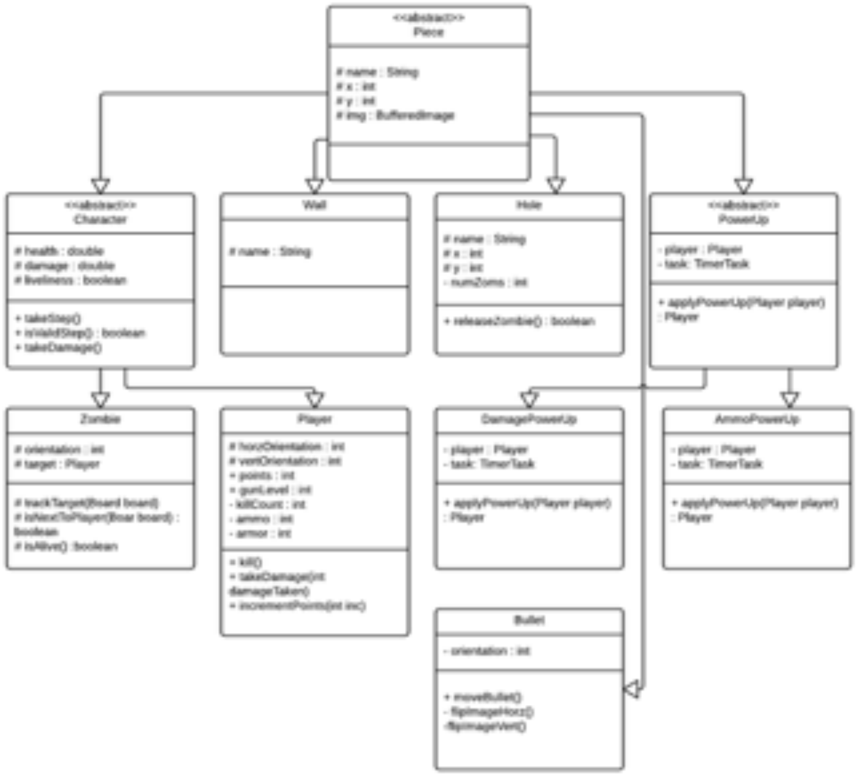
The first, and most fundamental, user story we have listed is that there is a player on the board, and the player can move. The player was able to freely move throughout the board, thanks to the Player DrawBoard classes' interactions with the Board class, and their interaction with ZombieView. The player class would update the X and Y value of the player, while DrawBoard would put that into reality on Board, and finally ZombieView would display the movement to the user. The next user story deals with attacking zombies, which is a collaboration of the Player, DrawBullet, Bullet, and Zombie class, with the usual interactions with Board and ZombieView. Player holds a value of how much damage it's going to deal, as well as which direction it is facing, and location on the board. The bullet is shot across the board, with DrawBullet, which has access to both Bullet and Player, and does the specified amount of damage to the

Zombie. Inside of Board, the Zombie's health is checked, and if it is 0 or below, it is removed from the board. Our third user story, has to deal with the Store. The Store interacts with Player, to get the Health, Ammo, Points, and Armor, StoreView and StoreController to be created and displayed, and finally ZombieController to pop up on the screen in between rounds. This user story was one of the easier ones to implement, and therefore was done much earlier in the project than many other user stories we expected to be contemporary with it. Because it was done so much earlier, it was not able to be tested until much later, and thankfully it worked relatively well on the first test. Seeing all of the vital signs of the player is a direct interaction between the Player class and the ZombieController and ZombieView. We simply used a JLabel for the bottom left of the User Interface, and from there we could get the board, the player on the board, and everything important about the board. To make sure that the information was always up to date, we used a Java API called "Utility Timer" that does things like sweep through to see if the Player is alive or dead, in addition to how much ammunition, and how many points the Player currently has access to. Finally, the user should be able to see a neat user interface, with the whole project coming together. This was accounted for in ZombieView, where we used a combination of JPanels, JLabels, and JButtons, to make, in our opinion, a gorgeous final product.

Below is our UML diagram which shows the entire program in a very basic overview. It is clear to see that the Board is composed of pieces, and all three of the "Drawing" Classes are directly related to the board. You can also see that the shop is related to the ZombieView which is related to the entire board.



Here is a more precisely made UML diagram, that highlights the inheritance of the Piece abstract class, and all of its subclasses, and their important methods.


<<abstract>>
Piece
# name : String
# x : int
# y : int
# img : BufferedImage
<<abstract>>
Character
# health : double
# damage : double
# liveliness : boolean
+ takeStep()
+ isValidStep() : boolean
+ takeDamage()
Wall
# name : String
Hole
# name : String
# x : int
# y : int
- numZoms : int
+ releaseZombie() : boolean
<<abstract>>
PowerUp
- player : Player
- task: TimerTask
+ applyPowerUp(Player player)
: Player
Zombie
# orientation : int
# target : Player
# trackTarget(Board board)
# isNextToPlayer(Boar board) :
boolean
# isAlive() :boolean
Player
# horzOrientation : int
# vertOrientation : int
+ points : int
+ gunLevel : int
- killCount : int
- ammo : int
- armor : int
+ kill()
+ takeDamage(int
damageTaken)
+ incrementPoints(int inc)
DamagePowerUp
- player : Player
- task: TimerTask
+ applyPowerUp(Player player)
: Player
AmmoPowerUp
- player : Player
- task: TimerTask
+ applyPowerUp(Player player)
: Player
Bullet
- orientation : int
+ moveBullet()
- flipImageHorz()
-flipImageVert()