This is a written diary of each time I implemented changes into the game, it tracks most changes and issues encountered along the way:  
  
-Scavenging ideas, thinking of a vampire survivors clone

-So far, found sprites for main character that may be of use, will implement these for main character and find some for enemies

-Using our class's code to implement movement in a top down 2d movement, sprites will involve walking sprites

--------

I implemented 2d movement on the player as instructed in class, along with sprite movement.

I had issues with flicker, this was due to me creating a sprite array containing an incorrect number of sprites, which as fixed shortly after.

Next, I was implementing the same kind of directional sprites onto an enemy character that would use the chase(object) method found in the rect class.

------------

I implemented a chase onto the enemy towards the player, but I was crashing due to an outOfBounds error when setting a pose for the enemy.

This was due to the cardinal directions such as "UP" being drawn from the GameBase class instead of the sprite class, I had to specify where it should be coming from.

-------------------------

So far everything works. I have implemented a basic tilemap using a dungeon tileset, placed a large floor in order to lay a foundation to the background while coding, I noticed I need to make quire a large double array to cover the floor, might need to find a way to shorten it.

-----------------------------

Next, I'm implementing damage mechanics into the game, I gave the player an Int representing health, and an Int to the enemy representing strength. I coded it so anytime the enemy overlaps with the player using the method in the rect class, the player would lose the enemy's amount of strength from their health.

I also implemented a timer into the game that allows the game to use the current in the system, useful for creating times. I created a timer into the player that is set to 0, and is reset to currentTime anytime the player is hit. This allows me to control how often the player can be hit within a time window.

---------------------

I implemented a flashing red overlay over the character, this is enabled any time the character is hit and disappears once currentTime + a chosen amount of milliseconds have elapsed.   
  
I had issues making the red overlay not go away, this was due to the reset for the red flash being placed in the same if statement that triggers the red overlay, not outside.

-------------

I decided to not use a flashing red overlay on the character. Instead, as it is visually better, I grabbed a sprite for a "wound" that would be drawn over the player on hit, which looks much better than a semi-transparent red overlay.

The wound cycles between 3 images, and so a random number generator is used on each trigger to select those images. It seems that the speed and order at which they trig

ger has kind of tangled and randomized how they appear, but I like how they look.

------------------

I added a sprite onto the player that symbolizes a sword slash, with it’s own rectangle and strength values which I yet have to implement.

I changed the logic behind managing the skeleton enemy to cater to a wider array of enemy times with differing stats. Making them all children to the Enemy class. These enemies are added to an arraylist of enemies which then allows me to do things such as damage calculation to all enemy types rather than specifying each one. There were a couple of issues:

The creation of enemies would lead to a concurrentModification crash from creating an enemy added into the arrayList while paiting said enemy, this made java do both at the same time in the AWT thread, which lead to a crash. The solution to this was so synchronize both methods.

The other issue is seeing how poses will work with the new logic, at the moment, the only skeleton spawned is leaving a trail. Finding a fix…

--------------------------

I was able to set the positions correctly for spawned enemies, switching the reference from skeleton to every enemy in the arrayList was good enough.

Now, I implemented damage interaction and a health bar into the enemies. The health bar is made by drawing a gray bar background with a red bar on top that is the ratio of the current enemy health over it's maximum health, multiplied by the width of the gray bar. This allowed the red bar representing current health to overlay the gray bar, representing max health.

An issue arose, damage calculation towards enemies works, however, they wont die at 0 health despite having 0 health, finding a fix…

-------------------------------  
  
I fixed the issue with enemies not dying, this was due to setting the enemies to null and not removing them to the list.

However, removing them from the list is not safe without null checks and will lead to a crash, the solution was to use an iterator and use .hasNext() to safety remove these enemies. Now enemies die and disappear without a crash.

-----------------------------

A lot of changes happened.

First, I switched from using the previous sword logic, meaning drawing the drames and having all the sword logic within the game class, to separating it into different classes like the player class. Sword, and now weapons, have parent classes of WeaponSprite that works with WeaponAnimation. These are versions that are somewhat similar to the original classes but with differences:

These only have one sequence of frames, which are reused in all four directions by mirroring them across each axis.

Their delay has been changed to follow milliseconds instead of frames, allows me to more consistently control the time between the sprites.

The mirroring got broken, fixing now…

------------------------

It took a lot of work to get mirroring fixing, but I had to use Graphics2D to use it's translation and rotation capabilities in order to mirror the sprites.

The sprites now mirror correctly, but their hitboxes do not reflect this change, I had to make changes in the orientation of the rectangles drawn by the sword depending on the direction the character is facing as to be able to reflect the change.

I had to edit some of the code to implement camX and camY changes to enable scrolling through the map, so I had to go back and adjust everything drawn ever to reflect this.

---------------------------------------

Next change implemented more enemy types, I customized the spawns to allow for a few things:

I used a random number generator to generate numbers between 1 and 3, the result will be weighted and will lead to spawning an enemy of a particular type.

These enemies, given the flexibility of the skeleton class, will have different stats, these can still be scaled as the game progresses.

---------------------------------

Next, I'm switching the current intent of the game from being a time based survival, to a round based survival. Where I implemented some code that tracks the amount of enemies killed in a round and, checks if they're all dead to increment the roundNumber. Stat scaling changes will be implemented through the current round number.

---------------------------------

Lots of changes happened this time around again. I implement the round system, which works when x amount of enemies are killed with a set number of spawns. I added an xp stat to each enemy which will grant your character xp based on the amount declared on the enemy object. At a threshold, you will level up. This ss implemented through a level up screen that is drawn, pulling out a string and a function from an array of rewards (which will be randomized later) and using a key listener to select one of the rewards. The reward selected, will be called as a runnable which will be able to be executed at selection, running that function and granting the reward. I read up about runnables and this was a very neat way to write the rewards with lambda functions.

I also added an exp bar just like the health bars but above the screen, along with more things such as a lv counter and a round in seconds timer (change from the previous System current time timer).

------------------------------------------------

More changes. I added collisioin detection for the enemies, but in a way to resolve clumping, so they push each other back when too close. This was some math that I had to look up and it was implemented in the code. Some of it dealt with diagonal stretching and adjusting the magnitude of the pushback between enemies. This, however, because it’s a double loop, it checks for collisions in a O^2 speed which makes it laggy. Onto this later…

I implement a new weapon, a bow. This weapon is functionally the same as a sword but it spawns an arrow in front of the player that moves at vx and vy speeds and disappears after a certain amount of time has passed. Various aspects of drawing are also controlled on whether the arrow has been fired or not with multiple flags. While the speed was set at vx 1 or vy 1 at most, the arrows moved too fast. A quick fix to this was making a frame counter that would increment every time the speed was applied per frame, so the arrows would only move every third frame.

This leads me to the following, I can apply the same frame delay logic to make the enemy collision detection less taxing. This is, however, before I implement fancier methods such as "quad trees", which I will read up on later on….

-----------------------------------

More changes, too many:

I implemented an axe, using the exact same logic as the bow but with a negative movement speed that changes signs in order for it to travel into an arc.

I implemented walls, using a different tile "W" that makes itself into a Rect object on creation in order for me to add collision to it and prevent the player from going off boundaries.

I gave the enemies the same hurt hit indicators as the player, same logic.

I overhauled the leveling system, used different logic that uses an array containing a reward and a random number generator that randomizes the rewards picked, while checking that the rewards have not been chosen already.

I also added sound using java's default sound library, added game music and sound effects for weapons and hits, these two are managed by two sound objects as one needs to be constantly playing music on a loop with no overlapping sound.

I think functionally, the game as of right now is good enough, but there's quite a lot of work ahead to do, there are many mechanics missing, menus and the game is very barebones. However, it is a good baseline to what the future could be built on. There are, however, a couple of issues that can be tackled on:

There is still some white artifacts on the tiles that sometimes goes away in the background, this could probably be fixed by filling the background with black, or it might be frame related, I can’t seem to figure it out.

There are some performance issues, while some of it might be related to my hardware, there is room for optimization, especially during collision checking.

The arrow direction changes when the player moves but at least keeps the correct travel direction.

These are changes that I need to fix in the future, but for now, the game is fully playable.