Group Project Journal
COSC 2440, Software Architecture and Design Implementation
2nd Semester 2016

This document records the journey of the group as it analyzes, designs and implements the software projects. Here is our Course Journal.

Group Name: ____________

Group Members' Names and S-number:

- _____________ s________
- _____________ s________
- _____________ s________

Week 1

As a group, come up with a list of three possible games that you may want to pursue. Include a short description of each:

1. Game Title: ___________
   Description: ____________

2. Game Title: ___________
   Description: ____________

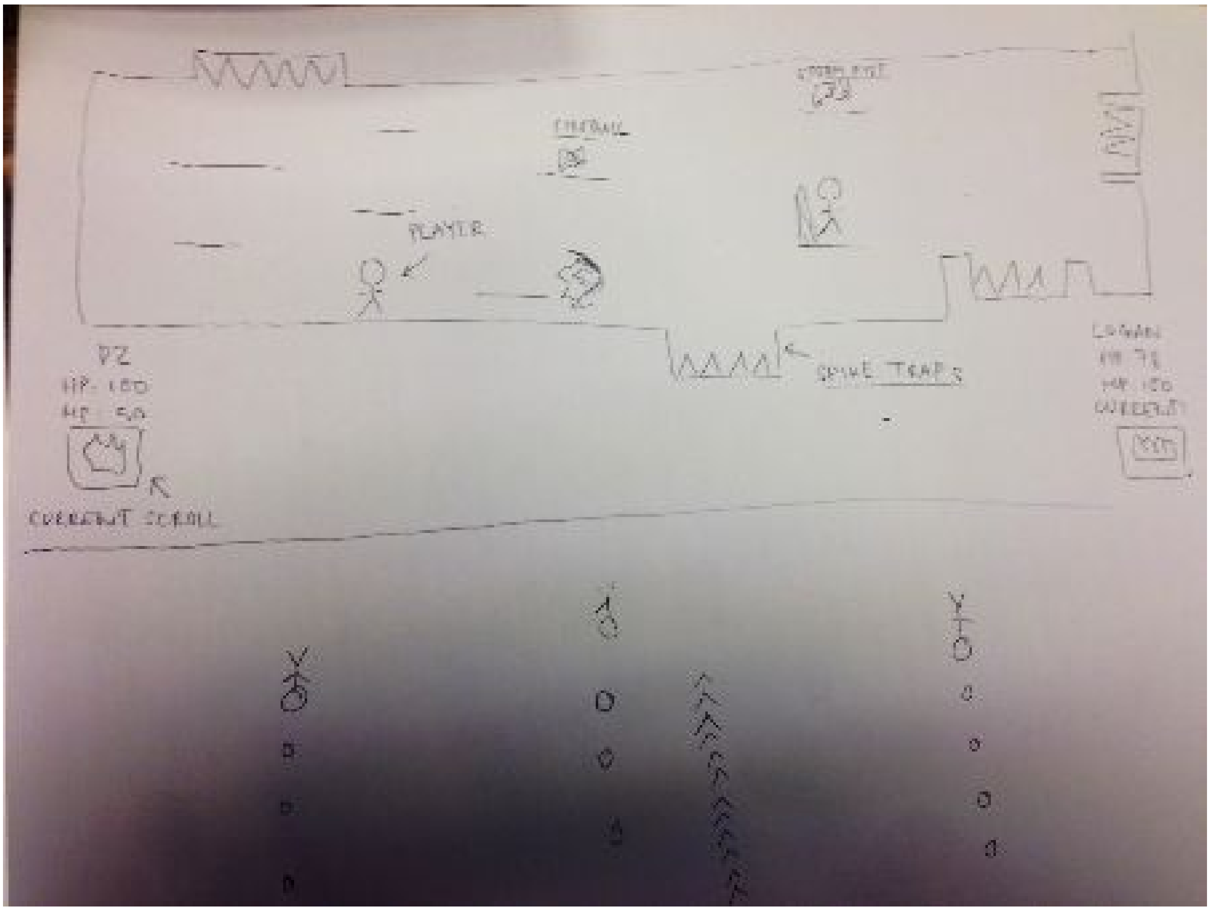
3. Game Title: ___________
   Description: ____________

Week 2

Game Name: Arcane Arena

A 2D Arena multiplayer game. Each player controls a character in a map. By default, characters can only use fists, but they can pick up scrolls to gain powerful spells. Players attempt to eliminate the other by using these scrolls. Maps will be designed with scrolls, platforms and pitfalls.

Scrolls range from basic attacks such as Fireballs, to scrolls that grant effects such as Galeforce (pushes the enemy), Frost Nova (creates a wall that blocks enemy movement) and Storm Fist (increases player's movement speed, makes default fist do more damage).



Week 3

User Stories:

- As a player, I want to see how much HP and Mana I have left so I can adjust the way I play.
- As a player, I want to see what scrolls are there on the board so I can adjust my strategies.
- As a player, I would like my game to be smooth and nice so that I don’t get frustrated.
- As a player, I would like my game to be low on requirements so I can play it on bad computers.
- As a player, I would like my game to not disconnect when I am playing a match so I can enjoy a smooth experience.
- As a player, I want to have a balanced game so I can have fun.
- As a player, I want the game to look good so I can enjoy the aesthetics of the game.
- As a player, I want diversity in scrolls so I don’t get bored.

- As a player, I want constant updates to the game so that it always feel fresh.
- As a player, I want to easily install and uninstall the game.

-

Title: Jumping on platform
Actor: player
Secondary actor: player 2
Scenario: jumping on platform, player stands on platform
Description: jumping on the platform to avoid all traps, get object skill, avoid shot from player 2
Scope:
Level:
Extensions:
Precondition: valid x,y coordinates
Postcondition: platform coordinates
Stakeholders: player 1
Technology list: collision detection
. . .

-

Title: shooting enemies
Actor : player 1
Secondary actor: player 2
Scenario: shooting each other
Description: by getting each shot from the enemies, the player will get damaged
Scope:
Level:
Extensions:
Precondition: any states except dead.

Postcondition: affected coordinates.
Stakeholders: player 1, player 2
Technology list: Collision detection, projectile detection
. . .

-

Title: Spike trap death
Actor: Any players
Secondary actor: Spike trap
Scenario: Step on the spike trap
Description: Stepping on the spike trap will lead to immediate elimination.
Scope:
Level:
Extensions:
Precondition: Collision with the spike
Postcondition: Death
Stakeholders: Players
Technology list: Collision detection
. . .

-

Title: Power up
Actor: Players
Secondary actor: Power scrolls
Scenario: Pick up a random power scroll
Description: Grant users power to shoot at the enemy or shield themselves from threads
Scope:
Level:
Extensions:

Precondition: Don’t have any power
Postcondition: Have a power
Stakeholders: Power
Technology list: Collision detection
. . .

-

Title: Winning
Actor: Players
Secondary actor:
Scenario: End the game when we have one winner
Description: If one player eliminate all other players -> win
Scope:
Level:
Extensions:
Precondition: No precondition
Postcondition: End the game
Stakeholders: All players
Technology list: No specification
. . .

Week 4
Use case

-

Title: Move the character

Actor: Character
Scenario: The player will press Space, A, S, D to move their character throughout the map. <System checks if the specific movement is possible>. Character should be jumping, move left (A), move down (S) and move right (D) accordingly. Collision cases Space -- change the velocity

-

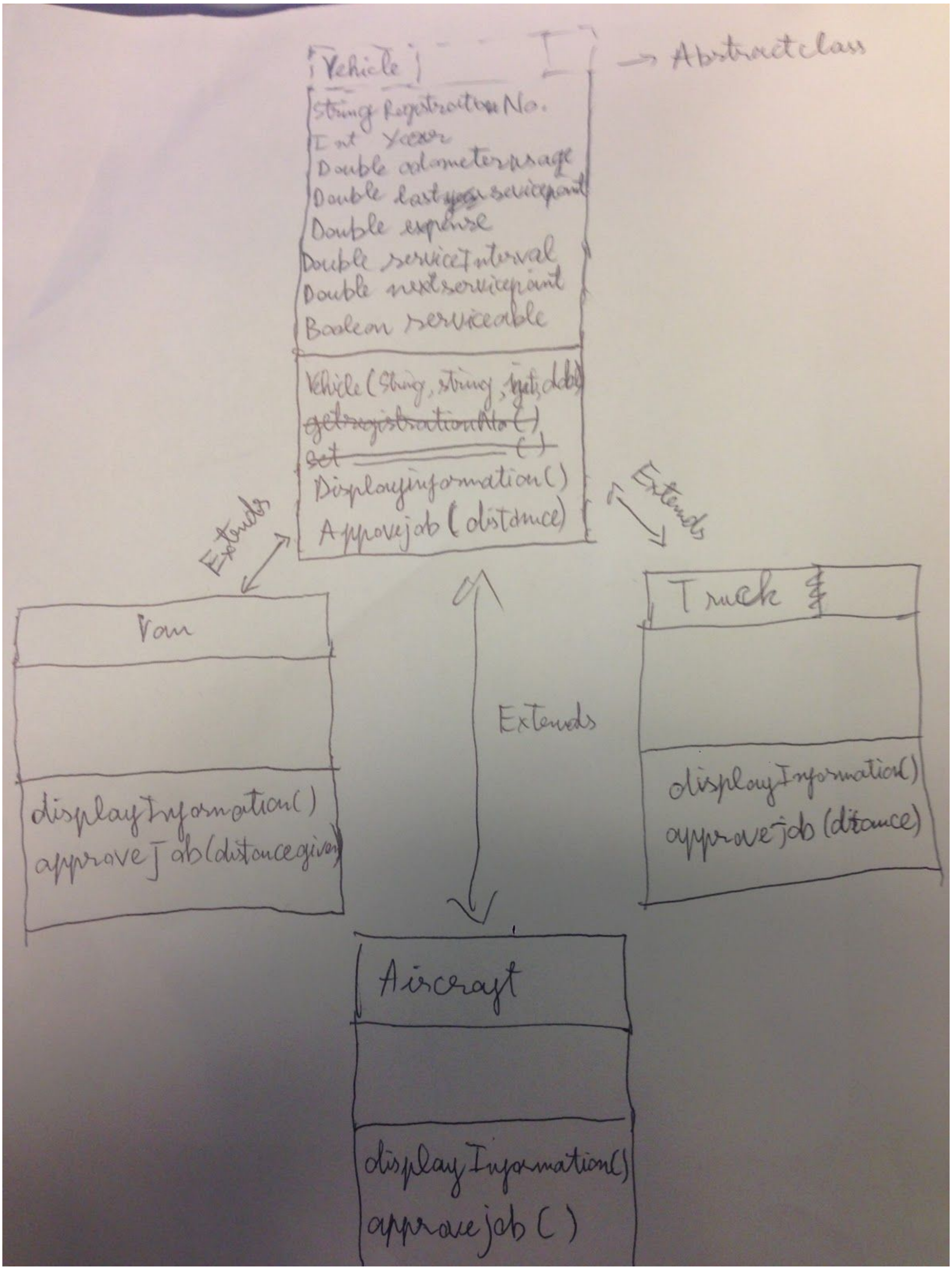
Title: Jump on the platform
Actor: Character
Scenario: The player will control the character who are standing on a ground or on another platform. Character will jump and move its location to the targeted platform and standing on a new platform

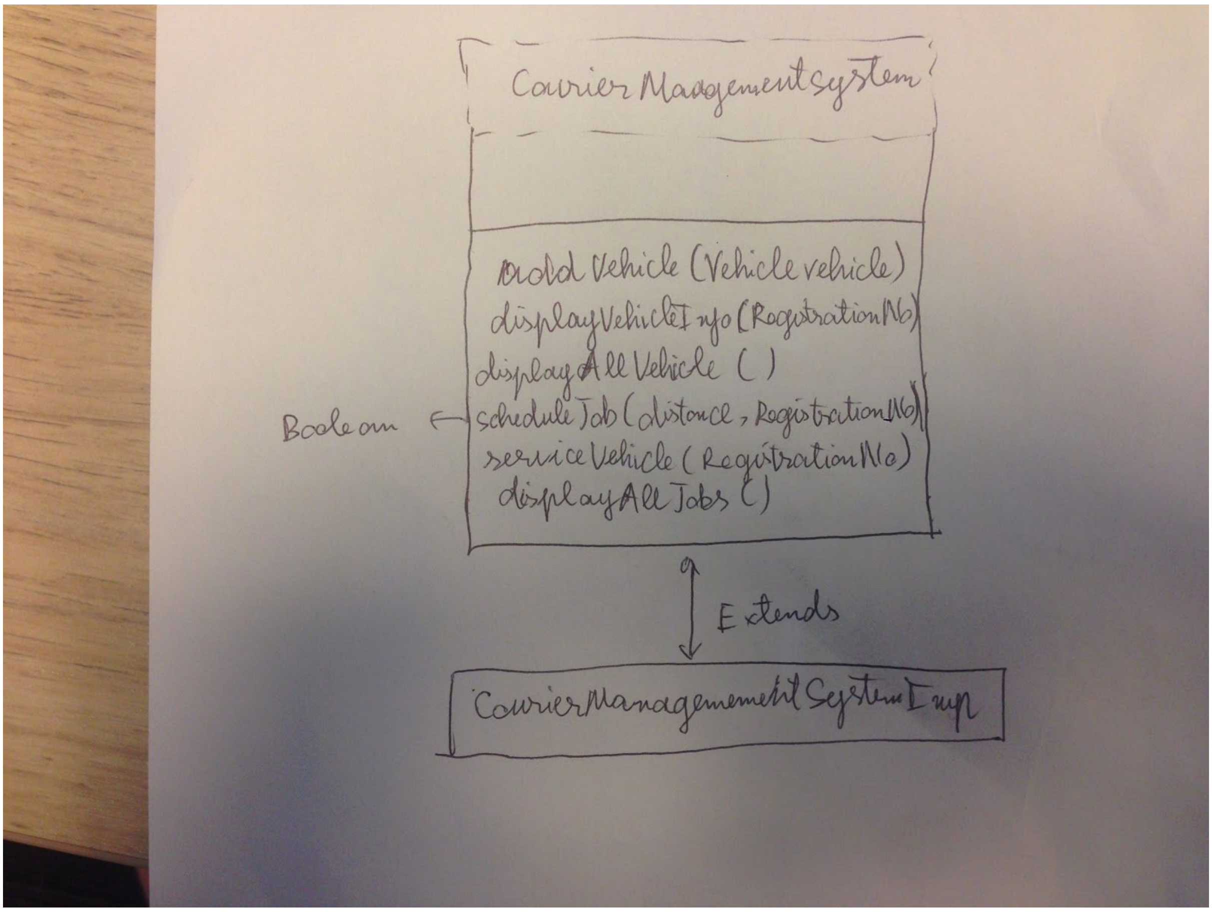
-

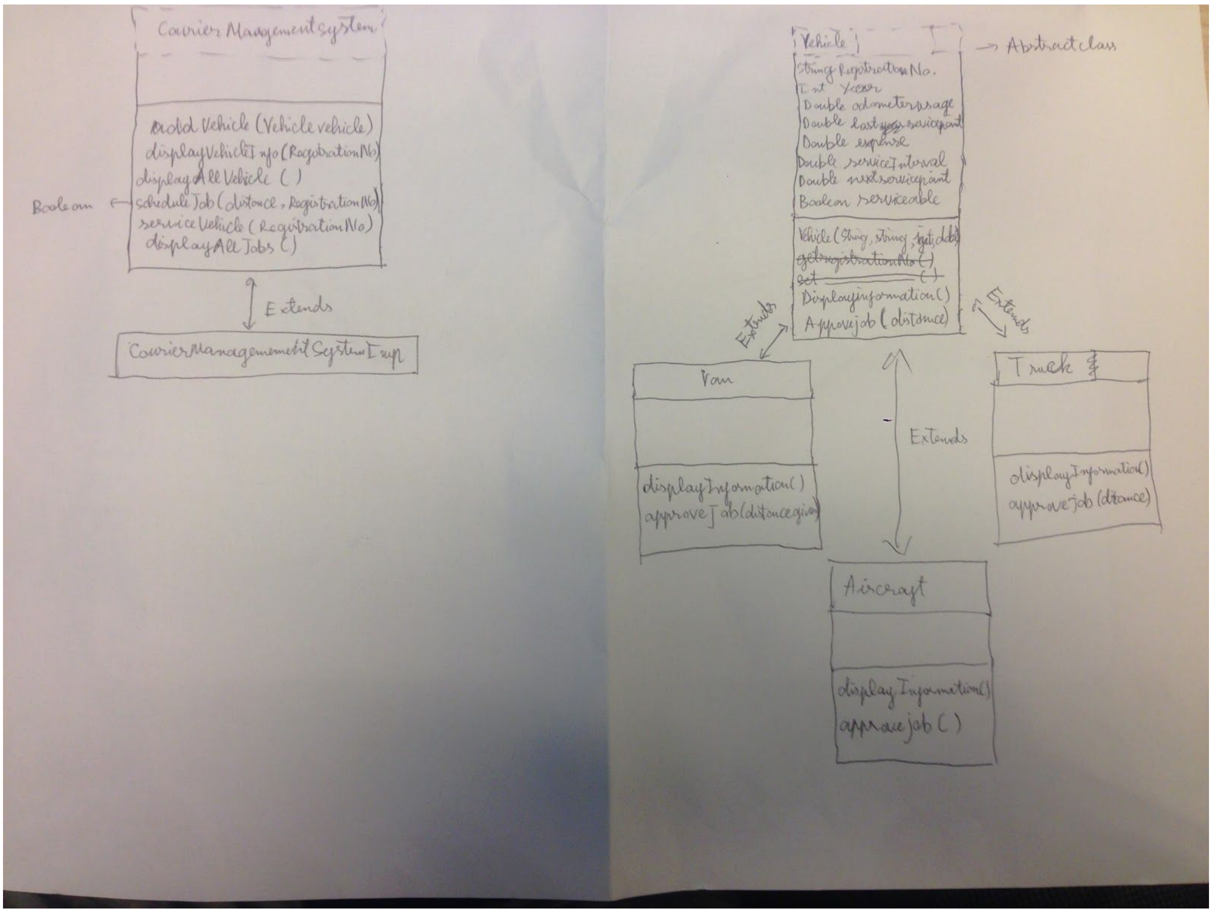
Title: Shoot the enemy
Actor: Character Secondary Character: AI or another player-controlled character
Scenario: <insert a drawing> The player will use a button (J) to shoot the enemy, each hit will take away 50hp on 250hp of the enemy. The character only able to shoot if they are holding a power scroll that pick up randomly in the map. Also, character can take damage of 50hp (except if they are having protected scroll then damage will be 30hp)

Week 5

Class Diagram for Courier Management System:




Courier Managementsystem
addVehicle (Vehicle vehicle)
displayVehicleInfo (RegistrationNo)
displayAllVehicle ( )
Booleam
scheduleJob (distonce, RegistrationNo)
serviceVehicle (RegistrationNo)
displayAllJobs ( )
Extends
CourierManagemenetSystemImp


Courier Managementsystem
addVehicle (Vehicle vehicle)
displayVehicleInfo (RegistrationNo)
displayAllVehicle ( )
Booleam
scheduleJob (distonce, RegistrationNo)
serviceVehicle (RegistrationNo)
displayAllJobs ( )
Extends
CourierManagemenetSystemImp
Vehicle
Abstractclass
String RegistrationNo.
Int Year
Double odometerusage
Double last ~~year~~ servicepoint
Double expense
Double serviceInterval
Double nextservicepoint
Boolean serviceable
Vehicle (String, string, int, dobl)
~~getregistrationNo ( )~~
~~set ( )~~
Displayinformation ( )
Approvejob (distance)
Extends
Extends
Extends
Van
displayInformation ( )
approveJob (distancegiven)
Truck
displayInformation ( )
approvejob (ditance)
Aircraft
display Information ( )
approvejob ( )

Week 6

Week 7

1.Prototype

Where to use :

-This pattern has been used to clone different sorts of wall objects and then use a map which is two dimensional arraylist to modify the location of the objects using setX() and setY() methods.

How to implement:

-Create an object tileCache which has a hashtable map.

- this tileCache calls the loadCache method which assigns all the key of different sorts of wall to their corresponding objects.
- Use getTile method to get the correct walls which are expected and then using 2-dimensional map and setX(),setY() methods to modify their location.
- add all different types of wall objects to a Linkedlist called tile instantiated in gameModel.

Why use Prototype:

-Instead of creating these objects from scratch in each element of 2 dimensional map when looping, we could get a correct object quickly using the getTile method with the corresponding key type put in the method's parameter. This can save lots of time and more convenient.

2.Singleton

Where to use

-This design pattern is applied to create TileCache class called by the loadGraphicsAndObjects method in the GameModel class.

-This can also be applied for some sorts of objects's handlers such as EntityHandler, BulletHandler,TileHandlers and the player.

How to implement

- Create a private static TileCache object named tileCache
- Make the constructor of TileCache private so it wont be able to be instatiated
- Create a public static method which returns an object created in step 1.

Why use singleton pattern object

Because there is only one tileCache containing all different sorts of tile that is needed in the game. Singleton can benefit in avoiding the confusion of many tileCaches.

3.MVC pattern

Where to use:

MVC pattern has been used for controlling the whole game.

How to implement

-Create a gameView class for displaying all components of gamemodel

-Create a gameModel containing all linkedlist handling objects,Stage,scene and so on. These are components making the game.

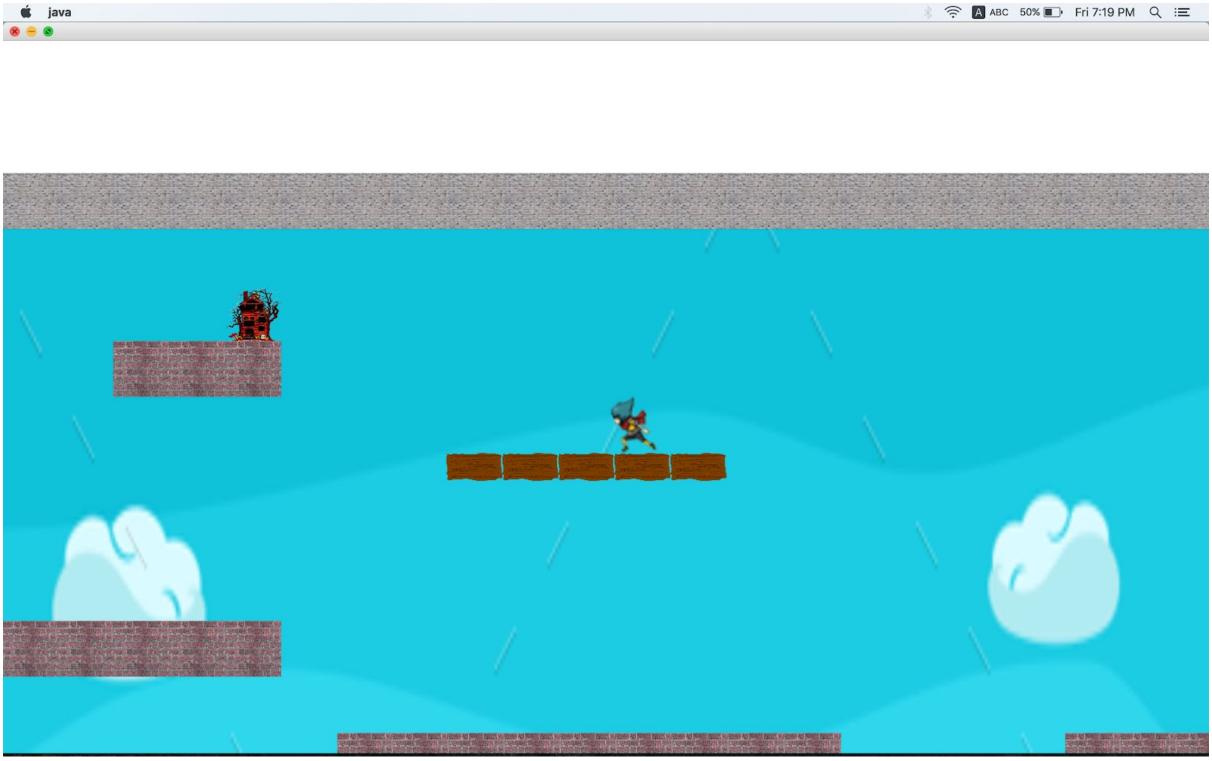
-Create a controller which takes gameModel and gameView as parameters to manipulate the game.

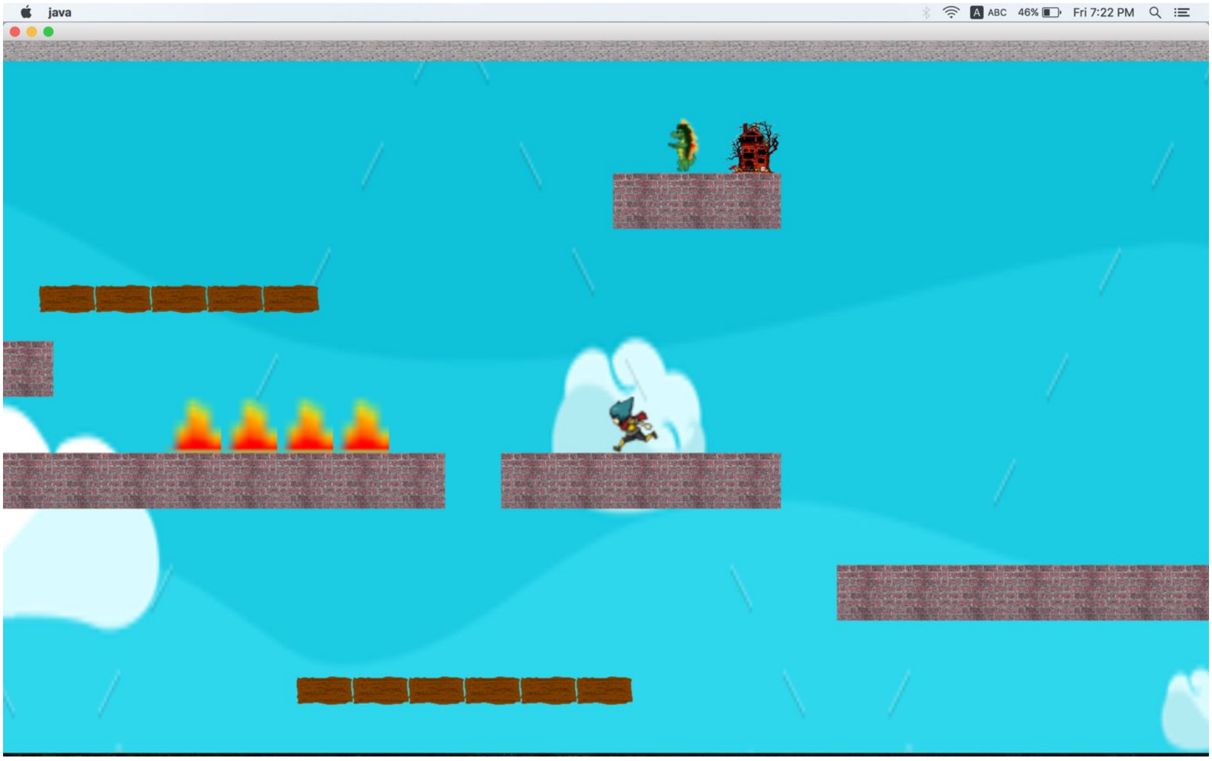
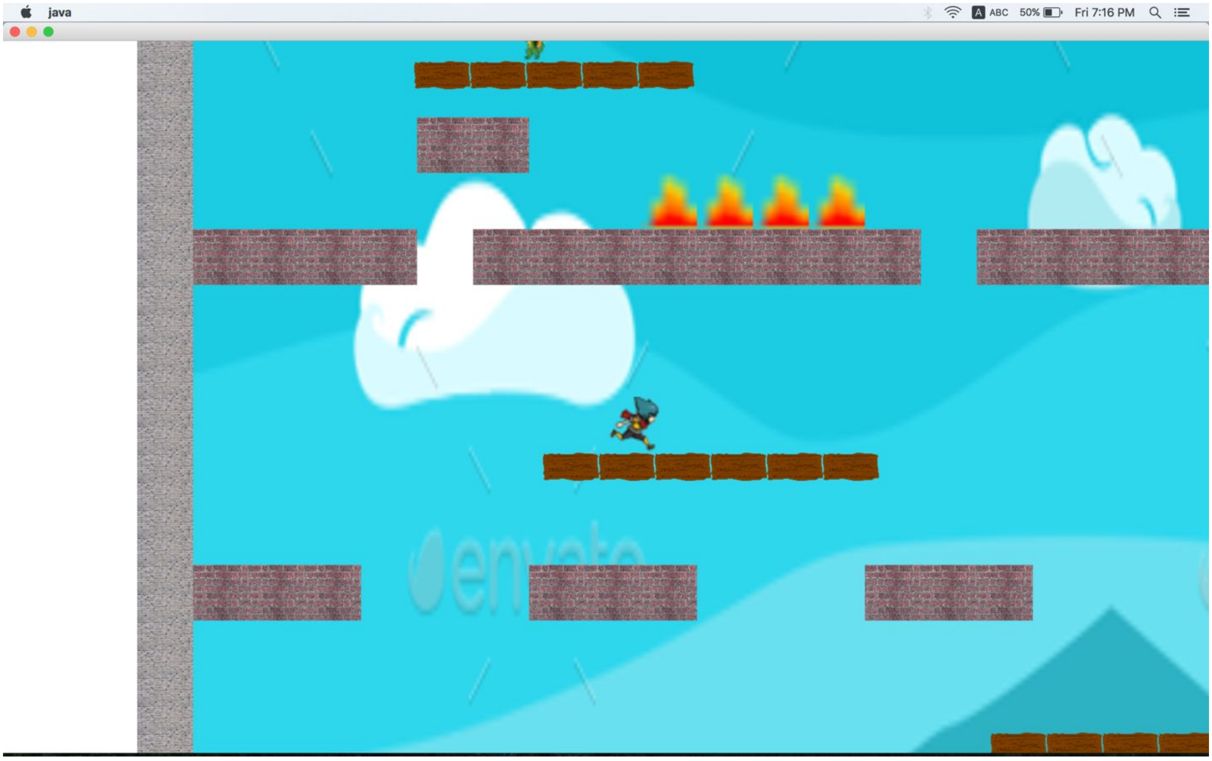
Why use it

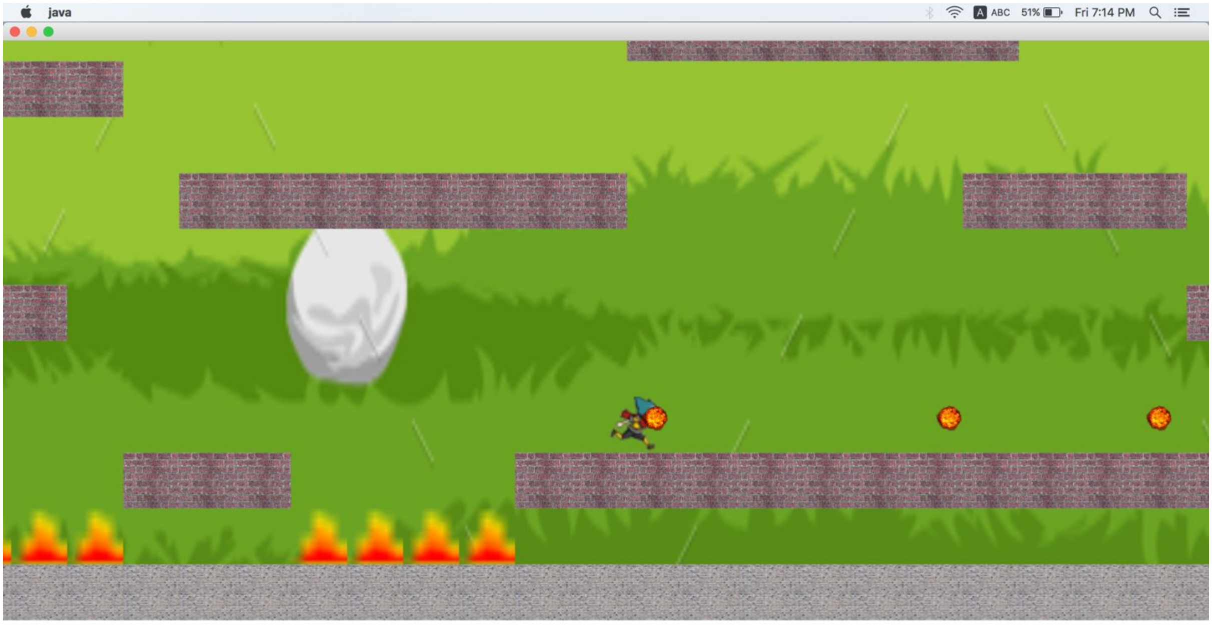
This separates the game logic and graphic of the games. This makes things much easier to maintain.

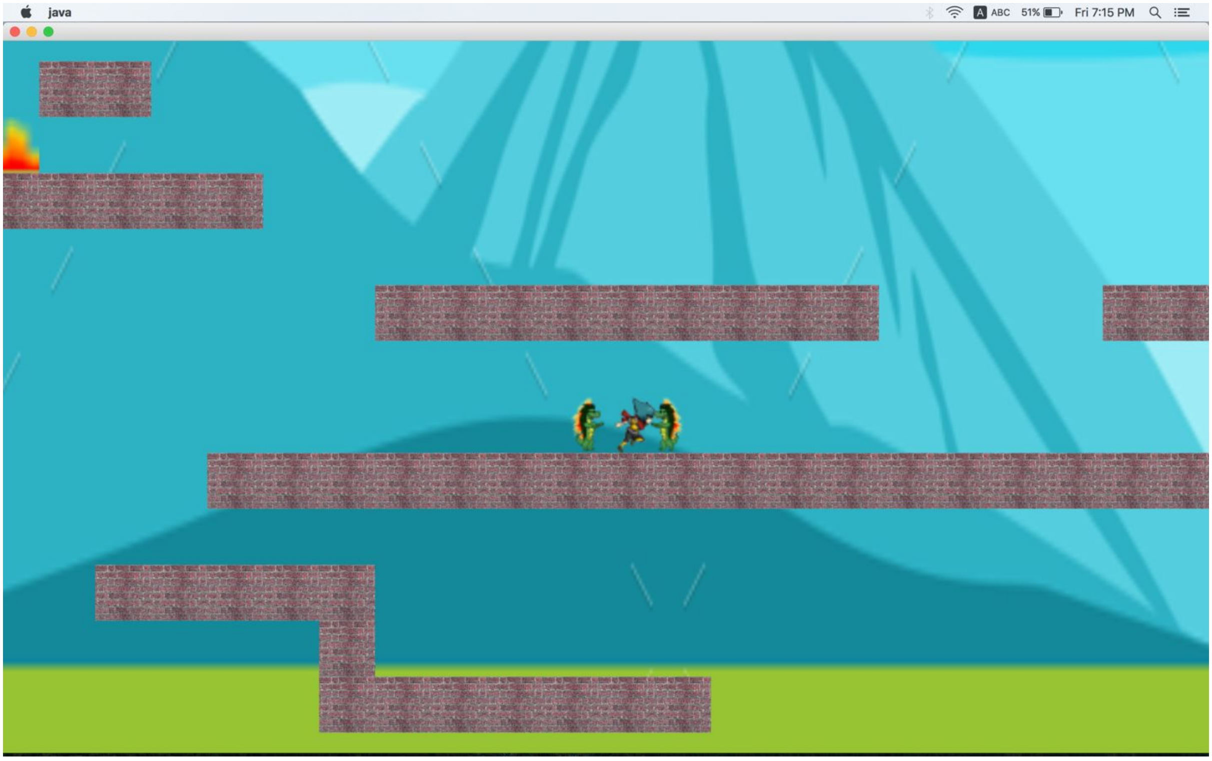
Week 8

Game Scene

- 

- 
java
ABC
46%
Fri 7:22 PM
- 
java
ABC
50%
Fri 7:16 PM
envato



- 

User Stories

- As a player, I want to see how much HP and Mana I have left so I can adjust the way I play.
- As a player, I want to be able to move my character in my desired direction such as moving left or right or jumping

- As a player, I would like my game to be smooth and nice so that I don't get frustrated.
- As a player, I would like my game to be low on requirements so I can play it on bad computers.
- As a player, I want my game is bug-free so i won't receive crash when playing the game
- As a player, I want to have a balanced game, not too hard and not too easy so i can have a wonderful experience with this game
- As a player, I want the game to look good so I can enjoy the aesthetics of the game.
- As a player, I want diversity of bullet types or skills
- As a player, i want diversity of level to enhance my replay experience
- As a player, I want constant updates to the game so that it always feel fresh.
- As a player, I want to easily install and uninstall the game.

Use Cases

Start the game
Actor: Player
Scenario: Player enter the game through the start button on the main menu

Move the character
Actor: Player
Scenario: Player can use A D Space to move left, right and jump relatively.

Attack the enemy
Actor: Player
Secondary actor: AI enemy, AI spawn house
Scenario: Player presses R to shoot the fireball, each fireball will deal damage to enemy and after 5 shots, normal AI enemy will die (disappear). Similar apply to AI house

Winning

Actor: Player
Secondary actor: AI house
Scenario: Player attempts to reach the top of the map where AI house is placed. After shooting a sufficient number of fireballs to the object AI house, the house will be destroyed (disappeared) and the game will end.

Losing
Actor: Player
Secondary actor: Spike tile, AI enemy
Scenario: If player takes too much damages by stepping/colliding with Spike tile (trap) or AI enemy, the character will die (disappear) and the game will end

Endless Enemy Spawn
Actor: AI house
Scenario: The AI house will keep spawning enemy after a set of time (5 seconds) until the house is destroyed by the player or the player is killed by trap or AI enemy

Moving platform
Actor: Moving platform
Secondary actor: Player
Scenario: The special platform will move from left to right and reverse. The player can jump to the moving platform using space key. After that, the player must move with the platform's moving direction otherwise, they will fall down

Collection

Roles:

LinkedList: Our game has 3 main classes: Entity, Bullet, Tile. Therefore, we need 3 handlers(EntityHandler,BulletHandler,TileHandler) which contain all corresponding objects. Each handler has a LinkedList as an attribute and some methods working around this LinkedList such as tick() which is just kind of checking, render(),add(),remove().

-To avoid confusion, we applied single pattern for them to ensure there are only three handlers the game is using.

Where and how to use

1/ Used in GameMap class

The game is designed by a 2 dimensional arraylist. The program will iterate through all of the elements of the array (400 elements this game). For each element, a suitable object will be created and added to its corresponding handlers.

Benefit: This actually can save lots of work as well as memory. Instead of creating 400 different variables to store these objects, the program just simply adds them the corresponding handlers.

2/ Used in tick method.

They can all be used for collision detection checking. The tick methods of some objects need to iterate

Why LinkedList:

The biggest benefit of LinkedList is removing objects with speed

Design Patterns

1.Prototype

Where to use :

-This pattern has been used to clone different sorts of wall objects and then use a map which is two dimensional arraylist to modify the location of the objects using setX() and setY() methods.

How to implement:

-Create an object tileCache which has a hashtable map.

- this tileCache calls the loadCache method which assigns all the key of different sorts of wall to their corresponding objects.
- Use getTile method to get the correct walls which are expected and then using 2-dimensional map and setX(),setY() methods to modify their location.
- add all different types of wall objects to a Linkedlist called tile instantiated in gameModel.

Why use Prototype:

-Instead of creating these objects from scratch in each element of 2 dimensional map when looping, we could get a correct object quickly using the getTile method with the corresponding key type put in the method's parameter. This can save lots of time and more convenient.

2.Singleton

Where to use

-This design pattern is applied to create TileCache class called by the loadGraphicsAndObjects method in the GameModel class.

-This can also be applied for some sorts of objects's handlers such as EntityHandler, BulletHandler,TileHandlers and the player.

How to implement

- Create a private static TileCache object named tileCache
- Make the constructor of TileCache private so it wont be able to be instatiated
- Create a public static method which returns an object created in step 1.

Why use singleton pattern object

Because there is only one tileCache containing all different sorts of tile that is needed in the game. Singleton can benefit in avoiding the confusion of many tileCaches.

3.MVC pattern

Where to use:

MVC pattern has been used for controlling the whole game.

How to implement

-Create a gameView class for displaying all components of gamemodel

-Create a gameModel containing all linkedlist handling objects,Stage,scene and so on. These are components making the game.

-Create a controller which takes gameModel and gameView as parameters to manipulate the game.

Why use it

This separates the game logic and graphic of the games. This makes things much easier to maintain.

Multi threads:

These three handlers entityHandler, bulletHandler,tileHandlers are shared resources used in many threads such as :

1/ In EntityHandler, BulletHandler, TileHandler classes

They are used in their tick and render methods which are called in ModelGame class to put inside the game loop.

2/ Some individual objects

For example: The tick method inside the fireball class makes use of entityHandler, tileHandler objects to check its collision detection with the others.

However, iterating the same linkedlist results in an error called concurrent modification exception meaning it is not permissible for one thread to modify the collection(linkedlist in this case) while another thread is iterating over it. To avoid this, all the original linkedlists should be duplicated, then the new versions could be used for iterating while the originals could be modified at the same time.

Class Diagram:

https://drive.google.com/a/rmit.edu.vn/file/d/0B6RVkfF1Q5MdWm5JbG5DaUwyNVk/view

Java code link
https://drive.google.com/open?id=0B6RVkfF1Q5MdRzYzMldZemtVZG8

Week 9

Week 10

Week 11

Week 12

Challenges:

- Did not know how a game worked. We had to learn how a game loop worked, how to handle collision detection and how to render a game with a tilemap (a 2D Array).
- We ran into problems with making an AI, so we turned to using Factory, Strategy to develop the AI and its reaction pattern.
- Handling user's score data in a consistent manner ran into some difficulties but we solved it with using a Singleton instance of the Score Data.
- Making a tile map with numbers was difficult so we decided to render the map with text instead.
- At first we struggle a little bit with map changing because all objects from previous map are still in arrays. We solve it by using create a function to clear out the old map before render the next map
- We ran into problem of dropping FPS due to the huge amount of item in the linkedlist. Also, we cannot delete the old object because linkedlist won't allow us to delete object when it is looping. We solved it by duplicating that linkedlist then operate on it
- At first, we didn't know how to create animation for the character. After a few research, we were able to set several animation state for the character. We ran to lots of bug when work with animation but we were able to handle them

Link to final code:
https://drive.google.com/open?id=0B6RVkfF1Q5MdeEZPMFQ0NENGWVE
Week 13-14
Revision:

- What is a User Story?

A User Story describes a single small scenario from a user's perspective focused on their goal rather than on the system. It's what they want to do and why do they want to do it. A user story is typically written as just one or two sentences, and they're very commonly written on index cards (PAPER).

- As a __(type of user)__, I want __(goal)__ so that __(reason)__.