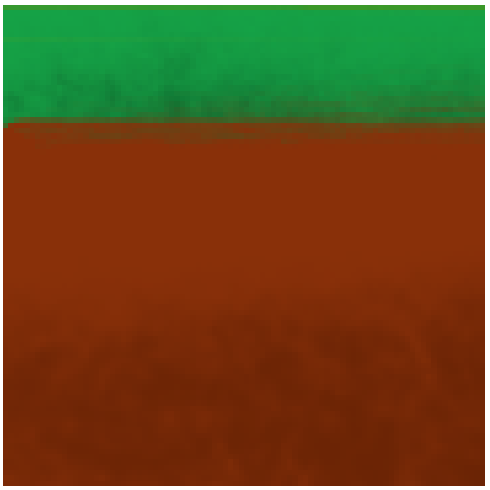
CLEAR THE DEBT ADVENTURE!!



Created by

Jirapon Jiratnanun 6531307521

Piyawudhi Sawaidee 6531326421

2110215 Programing Methodology

Semester 2 Year 2022

Chulalongkorn University

CLEAR THE DEBT ADVENTURE!!

Introduction

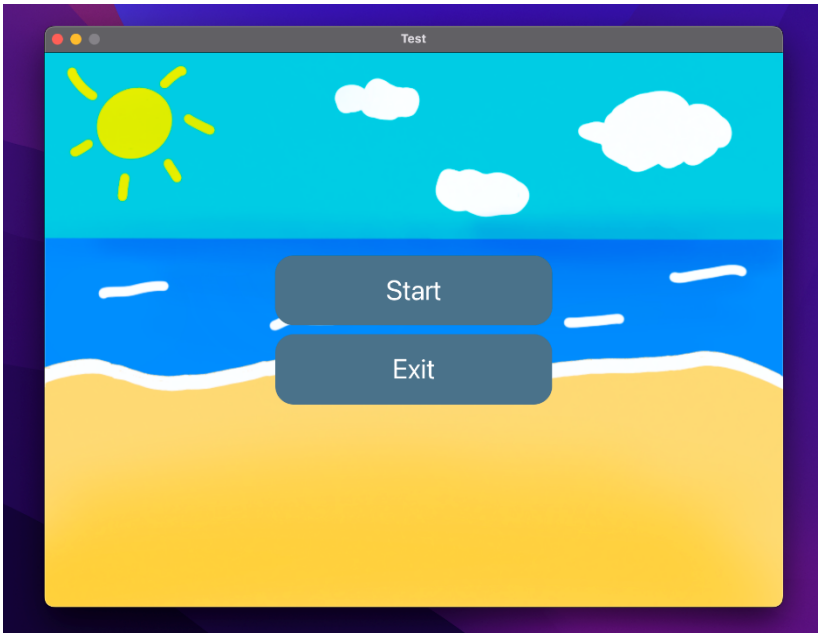
"Clear the Debt Adventure!!" This game will transport you to a mining place where you become a slime character tasked with amassing 1 million bath to repay a debt. Your primary objective is to collect valuable ores by mining and sell them at shops to accumulate the required funds. Be cautious, as time is of the essence! If you exceed the deadline, the unforgiving creditors will hunt you down. Prepare yourself for an exhilarating journey as you strive to clear your debt before it is too late!

Rule

The goal is to gather a total of 10,000 bath within 10 minute. You must achieve this by collecting various ores. Each with their respective values : stone (10 bath), iron (200 bath), gold (400 bath), ruby (800 bath) and diamond(1500 bath)

To aid your stamina, you can boost your stamina by purchasing stamina potions from in-game shop. Be careful, though, as allowing your stamina to deplete entirely will result in a game over.

Main menu scene

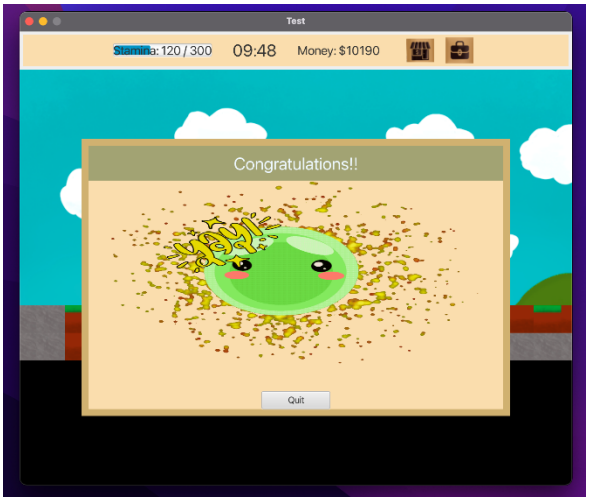


When you click “Start” button it will proceed you to game scene and “Exit” button to close the game.

Game Scene

Shop Scene

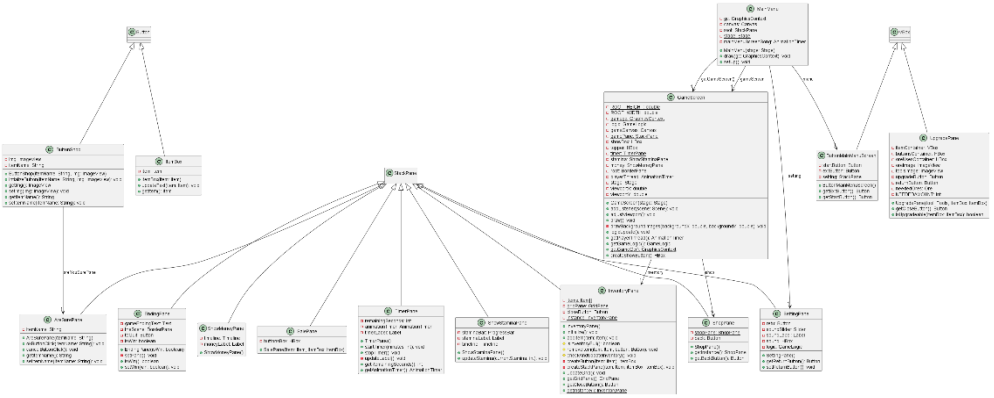
End Scene



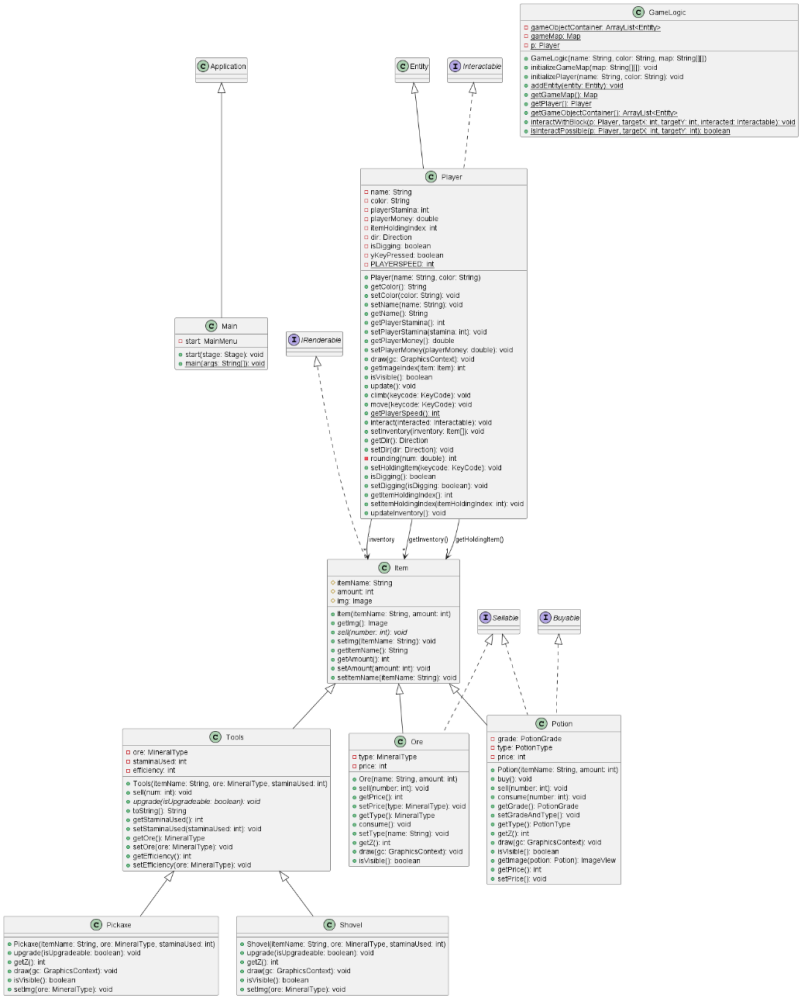
Key control

Key	Explanation
W	Up
A	Left
S	Right
D	Down
Y	Dig
1	Change to pickaxe
2	Change to shovel

UML diagrams



Gui UML



Game logic UML

Class Detail

1. Package application

1.1 class **Main** extends Application

This class is provided to launch JAvaFx applications.

1.1.1 Field

Name	**Description**
- MainMenu start	Instance variable

1.1.2 Method

Name	**Description**
+ void start(Stage stage)	Create a new MainMenu, ser stage title and show.
+ void main()	Launch the application.

2. drawing

2.1 class **AreSurePane** extends StackPane

This class represents a stackpane pop up when you click an item That you want to buy in a shop.

2.1.1 Field

Name	**Description**
- String itemName	Name of item

2.1.2 Constructor

Name	Description
+ AreSurePane(String ItemName)	Set scene with title of stackpane text and button box

2.1.3 Method

Name	Description
+ void okButtonClick(String itemName)	Create action method when ok button clicked
+ void cancelButtonClick()	Create action method when cancel button clicked
+ GETTER & SETTER for each field	

2.2 class ButtonMainMenuScreen extends VBox

This class represents a pane contains start button and exit button in MainMenu screen

2.2.1 Field

Name	Description
- Button startButton	Start button
- Button exitButton	Exit button

2.2.2 Constructor

Name	Description
+ ButtonMainMenuScreen()	Set scene size and spacing. Add start button and exit button

2.2.3 Method

Name	Description
+ GETTER & SETTER for each field	

2.3 class ButtonShop extends Button

This class represents button contains item image and set on for buying item in shop pane

2.3.1 Field

Name	Description
- ImageView img	Image of item
- String itemName	Name of item
- AreSurePane areYouSurePane	Represent AreSure pane

2.3.2 Constructor

Name	Description
+ ButtonShop(String itemName, ImageView img)	Initialize AreSurePane and set background scene

2.3.3 Method

Name	Description
+ void initialzeButton(String itemName, ImageView img)	Create button stand for item in game shop
+ GETTER & SETTER for each field	

2.4 class EndingPane extends StackPane

This class represents a stackpane pop up when the game ends.

2.4.1 Field

Name	Description
- Text gameEndingText	Text that show when game end
- BorderPane theScene	The container of all the children in this pane.
- Button toQuit	Quite button of the game
- Boolean isWin	State of wining of the game

2.4.2 Constructor

Name	Description
+ EndingPane(Boolean isWin)	Set scene size. Add background

2.4.3 Method

Name	Description
+ void setPane()	Initialize BorderPane, gameEndingText and toQuite button. Set scene size.
+ GETTER & SETTER for each cell	

2.5 class GameScreen

This class represents a screen of the game which contains every pane in this game.

2.5.1 Field

Name	Description
- final double ROOT_HEIGHT	Height of scene = 600
- final double ROOT_WIDTH	Width of scene = 800
- GrapicsContext gamegc	the GraphicsContext of gameCanvas
- GameLogic logic	Initialize the game logic.
- Canvas gameCanvas	the canvas used for display game.
- StackPane gamePane	Pane stand for game scene
- ShopPane shop	Pane stand for shop scene
- InventoryPane inventory	Pane stand for inventory scene
- Hbox showbox	Box contains shopButton and inventoryButton

- Hbox topper	Top box of game scene
- TimerPane timer	Pane shows deadline of game
- ShowStaminaPane stamina	Pane shows stamina of character
- ShowMoneyPane money	Pane shows money of character
- BorderPane root	Root of the scene
- AnimationTimer playerThread	The infinite loop of this game to update the state of this game
- Stage stage	Stage of the scene
- Double viewportX	Position X of player
- Double viewportY	Position Y of player

2.5.2 Constructor

Name	**Description**
+ GameScreen(Stage stage)	Set up shop inventory pane, game canvas, inventory button and toper.

2.5.3 Method

Name	**Description**
+ void addListener(Scene scene)	Add event listeners to scene
+ void adjustViewport()	Adjust the viewport based on player position
+ void draw()	Render a game object to game canvas

+ void drawBackgroundImages()	Draw the repeated background images
+ void logicUpdate()	Update game logic
+ Hbox createShowButton()	Initialize shopButton, inventory Button and add to game scene
GETTER & SETTER for each field	

2.6 class InventoryPane extends StackPane

This class represents an inventory that pops up when click at inventory button in the game screen.

2.6.1 Field

Name	**Description**
- Item[ ] items	Array contains item in inventory
- GridPane gridPane	Graphic of inventory
- Button closeButton	Close button of inventory scene
- InventoryPane instance	Represent the inventory pane

2.6.2 Constructor

Name	**Description**
+ InventoryPane()	Initialize closeButton, gridPane and array of inventory. Set scene size style and background

2.6.3 Method

Name	Description
+ void initialize()	Set array of inventory by add item
+ void addItem(Item item)	Add a new item into array of inventory
# void removeItem(Item item, Button button)	Decrease item amount from inventory. If the amount of item equal 0 remove it.
# void checkAndUpdateInventory	Update inventory after any action
# boolean isInventoryFull()	If inventory full return true
- ItemBox createButton(Item item)	Initialize itemBox and set action it
- void createStackPane(Item item, Button button)	Initialize SalePane and UpgradePane base on type of item
+ void updateGrid()	update gridpane to represent for inventory after any action.
+ GETTER & SETTER	

2.7 class ItemBox extends Button

This class represents a stackpane pop up when you click an item That you want to buy in a shop.

2.7.1 Field

Name	Description
- Item item	Represent for item

2.7.2 Constructor

Name	Description
+ ItemBox(Item item)	Set scene background and size. Set item and update text.

.

2.1.3 Method

Name	Description
+ void updateText(Item item)	Set text and style.
+ GETTER & SETTER	

2.8 class MainMenu

This class represents the main menu screen of the application.

2.8.1 Field

Name	Description
- GraphicsContext gc	Initializes graphic
- Canvas canvas	Initializes canvas
- StackPane root	Initializes root
- GameScreen gameScreen	
- Stage stage	Initialize stage

- ButtonMainMenuScreen menu	
- AnimationTimer mainMenuScreenSong	

2.8.2 Constructor

Name	Description
+ MainMenu(Stage stage)	Set stage. Initialize Canvas, ButtonMainMenuScreen and SettingPane

2.8.3 Method

Name	Description
+ void draw(GraphicsContext gc)	Set up the scene and stage. Add the canvas and button.Starts the screen song
+ void setUp()	Sets up the event listeners and button
+ GETTER & SETTER	

2.9 class SalePane extends StackPane

This class represents a pane that pops up when selling anything.

2.9.1 Field

Name	Description

- Hbox buttonBox	Pane contains consume button and sell button

2.9.1 Constructor

Name	Description
+ SalePane(Item item, Button button)	Initialize text field, consume button and sell button.

2.10 class ShopPane extends StackPane

This class represents a shop pane

2.10.1 Field

Name	Description
- final ShopPane shopPane	Represent shopPane
- Button back	close shop pane button

2.10.2 Constructor

Name	Description
+ ShopPane()	Set scene size, alignment and background . Add back button

2.10.3 Method

Name	Description
+ GETTER & SETTER for each field	

2.11 class ShowMoneyPane extends StackPane

This class represents a pane that displays the player's money.

2.11.1 Field

Name	Description
- Timeline timeline	Used to update the monet display
- Label moneyLabel	Display the player money

2.11.2 Constructor

Name	Description
+ ShowMoneyPane()	Initializes moneyLabel and sets up pane.

2.11.3 Method

Name	Description
+ GETTER & SETTER for each field	

2.12 class ShowStaminaPane extends StackPane

This class represents a pane that displays the player stamina

2.12.1 Field

Name	Description

- ProgressBar staminaBar	Represents the current stamina
- Label staminaLabel	Displays the current stamina.
- Timeline timeline	Used to update stamina.

2.12.2 Constructor

Name	Description
+ ShowStaminnaPane()	Initializes sstaminaLabbel and staminaBar

2.12.3 Method

Name	Description
+ void updateStamina(int currentStamina)	Update staminaBar and staminaLabel.
+ GETTER & SETTER for each field	

2.13 class TimerPane extends StackPane

This class represents a pane that displays a countdown timer.

2.13.1 Field

Name	Description
- int remainingSeconds	represent the time remaining
- AnimationTimer animationTimer	Used to update countdown timer.
- Label timeLabel	Shows current time

2.13.2 Constructor

Name	Description
+ TimerPane()	Initialize tinerLable and set up pane

2.13.3 Method

Name	Description
+ void startTimer()	Initializes the remaining time
+ void stopTimer()	Stop animationTimer
+ void updateLabel()	Update timerLabel
+ GETTER & SETTER for each field	

2.14 class UpgradePane extends Vbox

This class represents a pane that displays an upgrade interface for tools.

2.14.1 Field

Name	Description
- VBox itemContainer	Box contains item in pane
- Hbox buttonsContainer	Box contains button in pane
- Hbox oreUsesContainer	Box contains ore needed to use for upgrading
- ImageView oreImage	ore image
- Image toolsImage	tools image
- Button upgradeButton	upgrade button
- Button returnButton	Component of pane

- Ore neededOres	The ore that is needed to upgrade the item.
- final int NEEDEDAMOUNT	Amount of ore need to upgrade = 4

2.14.2 Constructor

Name	Description
+ UpgradePane(Tools tool)	Set scene size, spacing, padding, Alignment and background.

3. Package Entity

3.1 abstract class Block extends Entity implements Interactable

This class represents a block in the game screen.

3.1.1 Constructor

Name	Description
+ Block()	Set hp, visible, Z position

3.1.2 Method

Name	Description
+ abstract String getName()	Implement in subclass
+ void draw(GraphicCentext gc)	Rendering the block
+ void update()	Update visible block base on player position

+ GETTER & SETTER for each field	

3.2 class Cell

This class represents a cell that contains a Block object.

3.2.1 Field

Name	Description
- Block block	Represents Block object
- Boolean isEmptied	state of cell

3.2.2 Constructor

Name	Description
+ Cell()	Initializes block to null and isEmptied is true.

3.2.3 Method

Name	Description
+ boolean setBlock(Block b)	If Block is empty set isEmptied to false and return true.
+ void setEmptied(boolean isEmptied)	If block is destroyed set isEmptied to true.
+ GETTER & SETTER for each field	

3.3 class DirtBlock extends Block

This class represents a type of block called “Dirt”.

3.3.1 Field

Name	Description
- boolean hasGrass	State of dirt has grass.

3.3.2 Constructor

Name	Description
+ DirtBlock(boolean hasGrass)	Set hasGrass.

3.3.3 Method

Name	Description
+ void draw(GraphicsContext gc)	Define how the dirt block should be draw
+ public void interact(Interactable Interacted)	Srt block can interact with other entities.
+ String getName()	Return name of dirt block
+ GETTER & SETTER for each field	

3.4 abstract class Entity implements IRenderable

This class represents a common property of the entity in the game.

3.4.1 Field

Name	**Description**
# int x	Represents X position of entity
# int y	Represents Y position of entity
# boolean isDestroyed	Represents state entity is destroy
# boolean isVisible	Represents state of entity is visible
# int z	Represents Z position of entity
# int hp	Represent hp point of entity

3.4.2 Method

Name	**Description**
+ void setHp(int hp)	If hp less than 0 set to 0
+ abstract void update()	implement in subclass
+ GETTER & SETTER for each field	

3.5 class Map

This class represents the game map.

3.5.1 Field

Name	Description
- int width	Width of map
- int height	Height of map
- Cell[ ][ ] cellMap	Cell contains cell block
- final int BLOCKSIZE	size of block = 40
- int BLOCKDEPTH	dept of map

3.5.2 Constructor

Name	Description
+ Map(String [][] map)	Initialize cellMap and set up it to 2D array.

3.5.3 Method

Name	Description
+ void setBlock(Block b, int x, int y)	Set a block to specific position on the map
+ void addAllBlock()	Add all non-empty block to game logic
+ boolean isMoveable	Check if a player can move to a target position
+ GETTER & SETTER for each field	

3.6 class OreBlock extends Block

This class represents a type of block called “Ore”.

3.6.1 Field

Name	**Description**
- MineralType mineralType	Mineral type

3.6.2 Constructor

Name	**Description**
+ OreBlock(MineralType (mineralType)	Set mineral type

3.6.3 Method

Name	**Description**
+ void draw(GraphicsContext gc)	Draw the ore block on graphics context
+ String getName()	return mineral type
+ GETTER & SETTER for each field	

4. Package Exception

4.1 class InteractFailedException

This class represents failed interaction.

4.1.1 Field

Name	Description
- String message	Represents the message with exception

4.1.2 Constructor

Name	Description
+ InteractFailedException (String message)	Take message for exception

4.1.3 Method

Name	Description
+ void playSound()	play an error sound when exception
+ GETTER & SETTER for each field	

5. Package Input

5.1 class InputUtility

This class provided a method for handling keyboard input.

5.1.1 Field

Name	Description
- boolean triggeredOnce	Determines whether keys can triggered
- ArrayList<KeyCode> keyPresses	Contains currently press key

5.1.2 Method

Name	Description
+ void postUpdate()	Reset the trigger after processing.
+ GETTER & SETTER for each field	

6. Package logic

6.1 class GameLogic

This class manages the game logic and entities in the game.

6.1.1 Field

Name	Description

- ArrayList<Entity> gameObjectContainer	ArrayList contains all the entities in the game.
- Map gameMap	Represents game map
- Player P	Represents player

6.1.2 Constructor

Name	Description
+ GameLogic(String name, String color, String[ ][ ] map)	Initialize game object container, game map and player.

6.1.3 Method

Name	Description
+ void initializeGameMap(String[ ][ ] Map)	Initializes game map.
+ void initializePlayer(String name, String color)	Initializes the player.
+ void addEntity(Entity entity)	Adds an entity to game object container
+ void interactWithBlock(Player p, int tergetX , int target, Interactable interacted)	Interact with block by using interact method
+ boolean isInteractPossible()Player p, int tergetX, int target	If interaction is possible return true
+ GETTER & SETTER for each field	

6.2 abstract class Item

This class represents items in the game

6.2.1 Field

Name	Description
# protected String itemName	Name of item
# int amount	Amount of item
# Image img	Image of item

6.2.2 Constructor

Name	Description
+ Item(String itemName, int amount)	Set item name and amount

6.2.3 Method

Name	Description
+ abstract void sell()	Implement in subclass.
+ void setImg(String ItemName)	Set image base on item name
+ GETTER & SETTER for each field	

6.3 class Ore extends Item implements Sellable

This class represents ore in the game.

6.3.1 Field

Name	Description
- MineralType type	Mineral type
- int price	Price of ore

6.3.2 Constructor

Name	Description
+ Ore(String name, int amount)	Initialize all fields.

6.3.3 Method

Name	Description
+ void sell(int number)	-decrease the amount of this item in the player inventory. -if the amount is <= 0 remove it from the inventory. -increase the player money by the price of this potion*number and add this item to player inventory.
+ void setPrice(MineralType type)	-set price of this item depends on type STONE - 10 IRON - 200 GOLD - 400 RUBY - 800 DIAMOND - 1500
+ void setType(String name)	-set the ore of this item depends on the name.

+ GETTER & SETTER for each field	

6.4 abstract class Tools

This class represents tools in the game.

6.4.1 Field

Name	Description
- MineralType ore	Represent the ore that the tool are made from.
- int staminaUsed	Represent the stamina needed for the player to use the tool.
- int efficiency	Represent the efficiency of the tool.

6.4.2 Constructor

Name	Description
+ Tools(String itemName, MineralType ore, int staminaUsed)	Initialize all fields

6.3.3 Method

Name	Description

+ void upgrade()	
+ void sell(int number)	
+ GETTER & SETTER for each field	

6.5 class Player extends Entity implements Interactable

This class represents the character in the game that the player controls.

6.5.1 Field

Name	**Description**
- String name	Represent player name
- int playerStamina	Represent player stamina
- double playerMoney	Represent player money
- Item[ ] inventory	Represent player inventory
- int itemHoldingIndex	Represent the index of items in inventory that the player holds. -always set it to 0.
- Direction dir	Represent the direction that the player is facing
- boolean yKetPressed	Represent state of digging
- int PLAYERSPEED	The player speed which is BLOCKSIZE / 10

6.5.2 Constructor

Name	**Description**
+ Player(String name)	Initialize all fields

6.5.3 Method

Name	Description
+ void draw(GraphicsContext gc)	Draw the player depending on the state of the player is.
+ boolean isVisible()	Always true
+ void update()	Update the state of the player
+ void climb(KeyCode keycode)	move the player up. Execute when keycode is W cannot go up if the player Y exceed BLOCKDEPTH
+ void move(KeyCode keycode)	Move the player to the direction based on keycode A-Left D-right S-down
+ void interact(Interactable interacted)	Interact with the interacted if the interacted not a block don’t do anything. Decrease the interacted block hp by the player tools efficiency * factor of the tools if the block hp <= 0 set the block isDestroyed to true, set the cell that contains the block to be emptied.
+ int rounding(double num)	Use for changing X,Y to be in the block unit.

+ void setHoldingItem(KeyCode keycode)	-if keycode is not 1,2 don't do anything -if it is 1 change the playerHoldingItemIndex to be 0 -if it is 1 change the playerHoldingItemIndex to be 1

6.6 class Potion extends Item implements Sellable, Buyable

This class represents potion used to restore stamina of player

6.6.1 Field

Name	**Description**
- PotionGrade grade	-represent the grade of the potion.
- int price	-represent the price of the potion.

6.6.2 Constructor

Name	**Description**
+ Potion(String itemName, int amount)	-initialize all fields

6.6.3 Method

Name	**Description**

+ void buy()	-decrease the player money by the price of this potion and add this item to player inventory
+ void sell(int number)	-decrease the amount of this item in the player inventory. -if the amount is <= 0 remove it from the player inventory. -increase the player money by 0.8*the price of this potion * number.
+ void consume(int number)	-increase the stamina of the player when used depends on the grade of the potion. LOW - 10 MID - 20 HIGH - 30
+ getImage(Potion potion)	-get image of the potion.
+ setPrice()	-set price of the potion depends on the grade LOW - 10 MID - 20 HIGH - 30
+ GETTER & SETTER for each field	

6.7 class Shovel extends Tools

This class represents a shovel item used by the player to dig the block.

6.7.2 Constructor

Name	Description
+ Shovel(int durability, String itemName, MineralType ore, int staminaUsed)	-initialize all field

6.7.3 Method

Name	Description
+ void upgrade()	upgrade the tool to the greater ore use the method getUpgrade of the MineralType enum and set the img to according to the new upgrade tool.
+ GETTER & SETTER for each field	

6.8 class Pickaxe extends Tools

This class represents a pickaxe item used by the player to dig the block.

6.8.1 Constructor

Name	Description
+ Tools(int durability, String itemName, MineralTytype ore, int staminaUsed)	-initialize all field

6.8.2 Method

Name	Description
+ abstract void upgrade()	-upgrade the tool to the greater ore use the method getUpgrade of the MineralType enum and set the img to according to the new upgrade tool.
+ String toString()	-return the String of this object in NAME made of :ORE StaminaUsed : STAMINA when NAME represent the tool name, ORE represent tool ore and stamina represent staminaUsed of the tool
+ void setEfficienct(MineralType ore)	Set the efficiency of the tool based on the tool ore STONE 10, IRON 20, GOLD 30, RUBY 40, DIAMOND 50
+ GETTER & SETTER for each field	

7. Package sharesObject

7.1 class AudioManager

This class holds all the audio used in this game.

7.1.1 Field

Name	Description
+ AudioClip ERROR	Error sound
+ AudioClip gameMusic	Game sound

7.2 interface IRenderable

This interface represents the entities that are able to be drawn.

7.2.1 Method

Name	Description
+ int getZ()	Get the Z field of the entity.
+ void draw(GraphicsContext gc)	Draw the entity to canvas.
+ boolean isVisible()	Check whether to draw or not.

7.3 class RenderableHolder

This class is used to hold references to different renderable objects or resources in a game or application and is used for holding all images in game.

7.3.1 Field

Name	Description
- RenderableHolder instance	The instance of this class
- ArrayList<IRenderable> entities	Hold all the IRenderable entities in the game.
- Comparator<IRenderable> comparator	Use for comparison before adding a new entity to entities.

+ Image background	
+ Image showInventoryButton	
+ Image showShopButton	
+ Image lose	
+ Image win	
+ Image dirt	
+ Image stone	
+Image copper	
+ Image iron	
+ Image gold	
+ Image ruby	
+ Image diamond	
+ Image grass	
+ Image digged	
+ Image ironBar	
+ Image goldBar	
+ Image rubyBar	
+ Image diamondBar	
+Image hpPotionI	
+ Image hpPotionII	
+ Image hpPotionIII	
+ Image staminaPotionI	
+ Image staminaPotionII	
+ Image staminaPotionIII	
+ Image Idle	
+ Image jump	
+ Image fall	
+ Image left	
+ Image right	

+ Image[ ] pickaxeDiggingLeft	
+ Image[ ] pickaxeDiggingRight	
+ Image[ ] shovelDiggingLeft	
+ Image[ ] shovelDiggingRight	
+ Image buttonShopBackground	
+ Image shopBackground	
+ Image backToHomeButton	
+ Image backgroundInventory	

7.3.2 Constructor

Name	Description
+ RenderableHolder()	Initialize all of the fields.

3.3.3 Method

Name	Description
+ void loadResource	Load all resources in this game.
+ void add(IRenderable entity)	Adding entity to entities and sort entities after adding it.
+ GETTER & SETTER for each field	

8. Package Util

8.1 interface Buyable

This interface represents the entities that are able to be bought.

8.1.1 Method

Name	Description
+ void buy()	Implement in sub class

8.2 class CSVParser

This class read data in .csv file and return it to String[][]

8.2.1 Method

Name	Description
+ String[][] readCSV(String filename)	return data in .csv file in form of String[][]

8.3 enum Direction

This class represents a set of all players possibly direction.

8.3.1 Field

Name	Description
NONE	
LEFT	
RIGHT	
UP	
DOWN	

8.4 interface Interactable

This interface represents entities that are able to be interacted or can interact.

8.4.1 Method

Name	Description
+ void interact(Interactable interacted)	Interact with the interacted. Effect are different depending on class.

8.5 enum MineralType

This enum represents different types of minerals found in the game.

8.5.1 Field

Name	Description
DIRT	
STONE	
IRON	
GOLD	
RUBY	
DIAMOND	

8.5.2 Method

Name	Description

+ String toString(MineralType type)	Return the string of the MineralType.
+ MineralType getUpgrade(MineralType ore)	Determine the upgraded mineral type based on the input type STONE -> IRON IRON -> GOLD GOLD -> RUBY RUBY -> DIAMOND if the input type is DIAMOND return DIAMOND and print “Invalid Ore”

8.6 enum PotionGrade

This enum represents different grades of potion.

8.6.1 Field

Name	Description
LOW	LOW potion increase stamina by 10
MID	MID potion increase stamina by 20
HIGH	HIGH potion increase stamina by 30

8.7 interface Sellable

This interface represents the entities that are able to be sold.

8.7.1 Method

Name	Description
+ void sell(int number)	Sold the entities number pieces to shop and add the money to the player by the price*number