My Miner Documentation

Created by

Bodeesorn Kosullawrit 6330292321

Tikhamporn Tepsut 6330203521

2110215 Programming Methodology

Semester 1 Year 2020

Chulalongkorn University

My Miner

Introduction

My Miner is inspired by a game named “Mega Miner”. The objective of the game is to collect minerals in the soil and sell them in the store.

Rules

On each move, the player can control the truck to move by

- "Press W" to move the truck upward

- "Press S" to move the truck downward

- "Press A" to the truck move left

- "Press D" to the truck move right.

Note: Different resources sell for different amounts. (

COAL -> 20$, IRON -> 50$, SILVER -> 80$

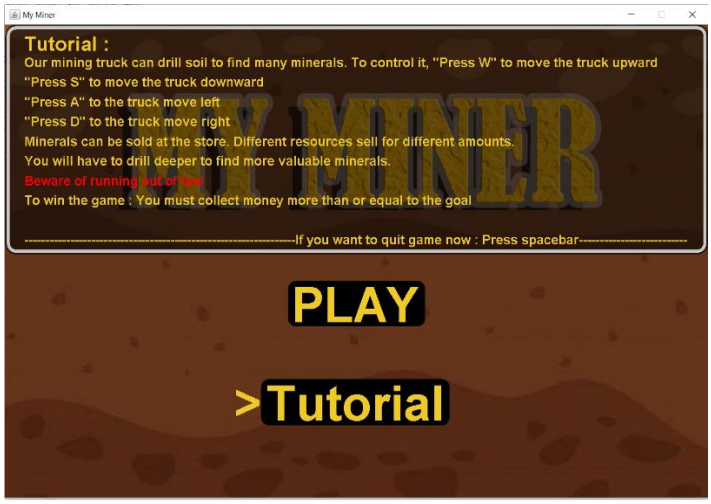
GOLD -> 150$, SAPHIRE -> 250$, EMERALD -> 400$

RUBY -> 700$, DIAMOND -> 1000$)

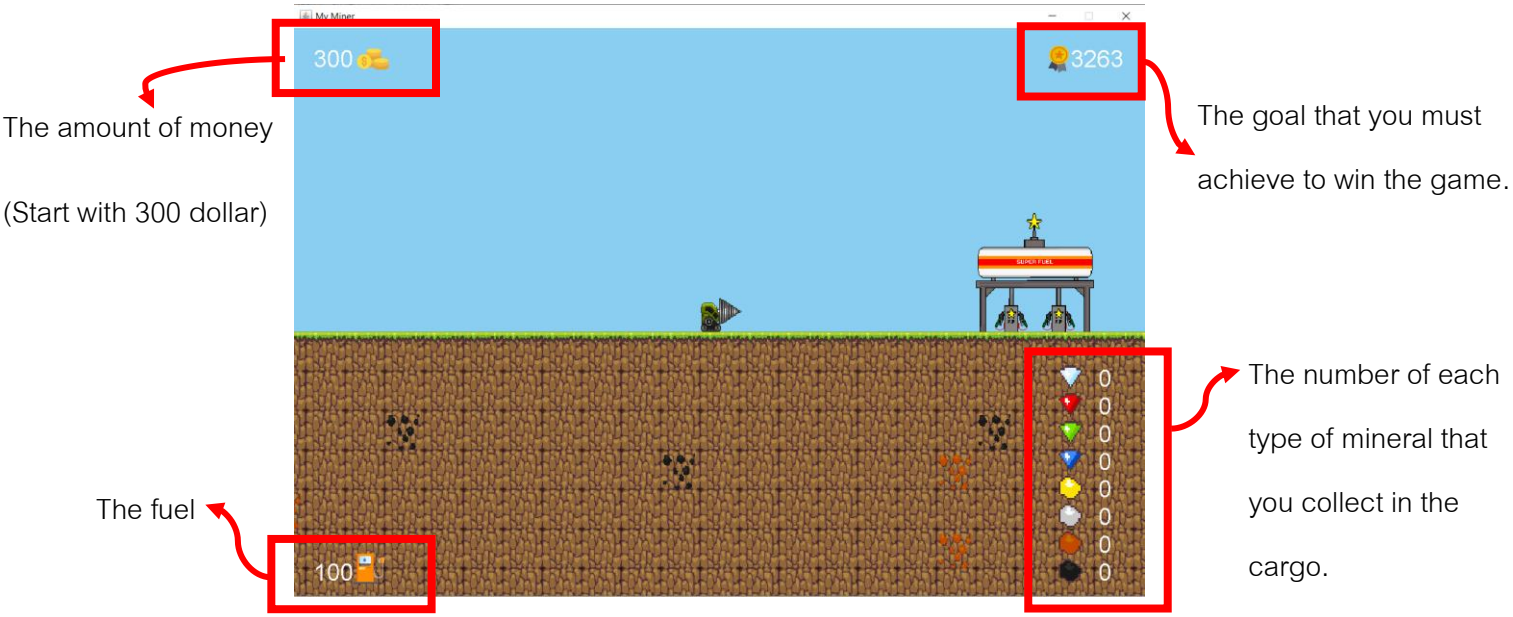
To win this game, you must collect money more than or equal to the goal. (The goal will be random.) However, the truck cannot drill some type of rock. If you try to drill them, you will lose your fuel in vain. Each type of mineral or soil or rock has a different hardness to drill. The more valuable the minerals it is, the harder it is to drill. More valuable minerals usually stay deeper, but you must be aware of running out of fuel. (If you run out of fuel, you will lose the game.) The fuel will be decreased every time you press the keys W, S, A, and D (Because these keys refer to the working of the machine of the truck)

Tip: you can press P to pause the game and press ESCAPE to exit the game

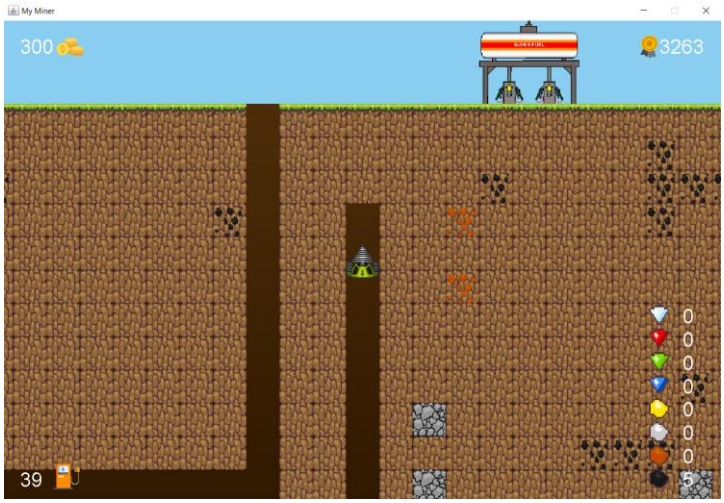
Example



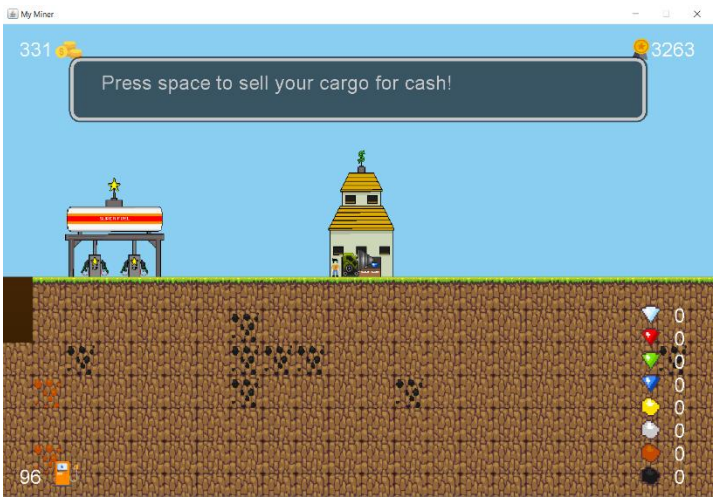
- This is the title scene when you run the game. You can read the tutorial (like in the right picture). If you select PLAY, you will go to the screen game.



- The play screen will show as the above picture.

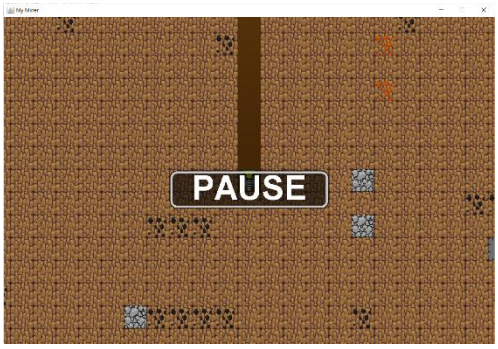


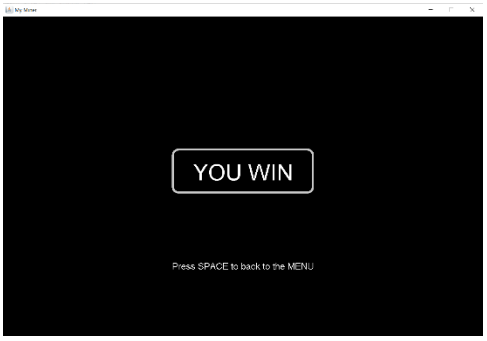
- The example when you drill the soil to find minerals.

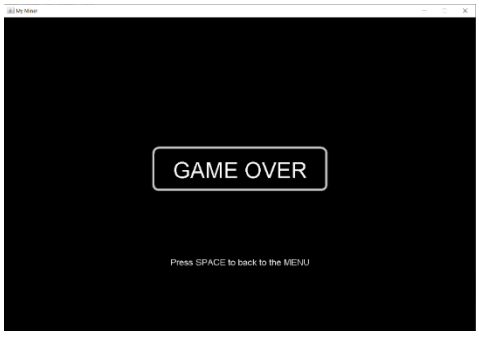


- There is the gas station to refuel and the store to sell the minerals in your cargo.

- The rock in this picture is the one that you can’t drill. If you try to drill it you will lose your fuel in vain.

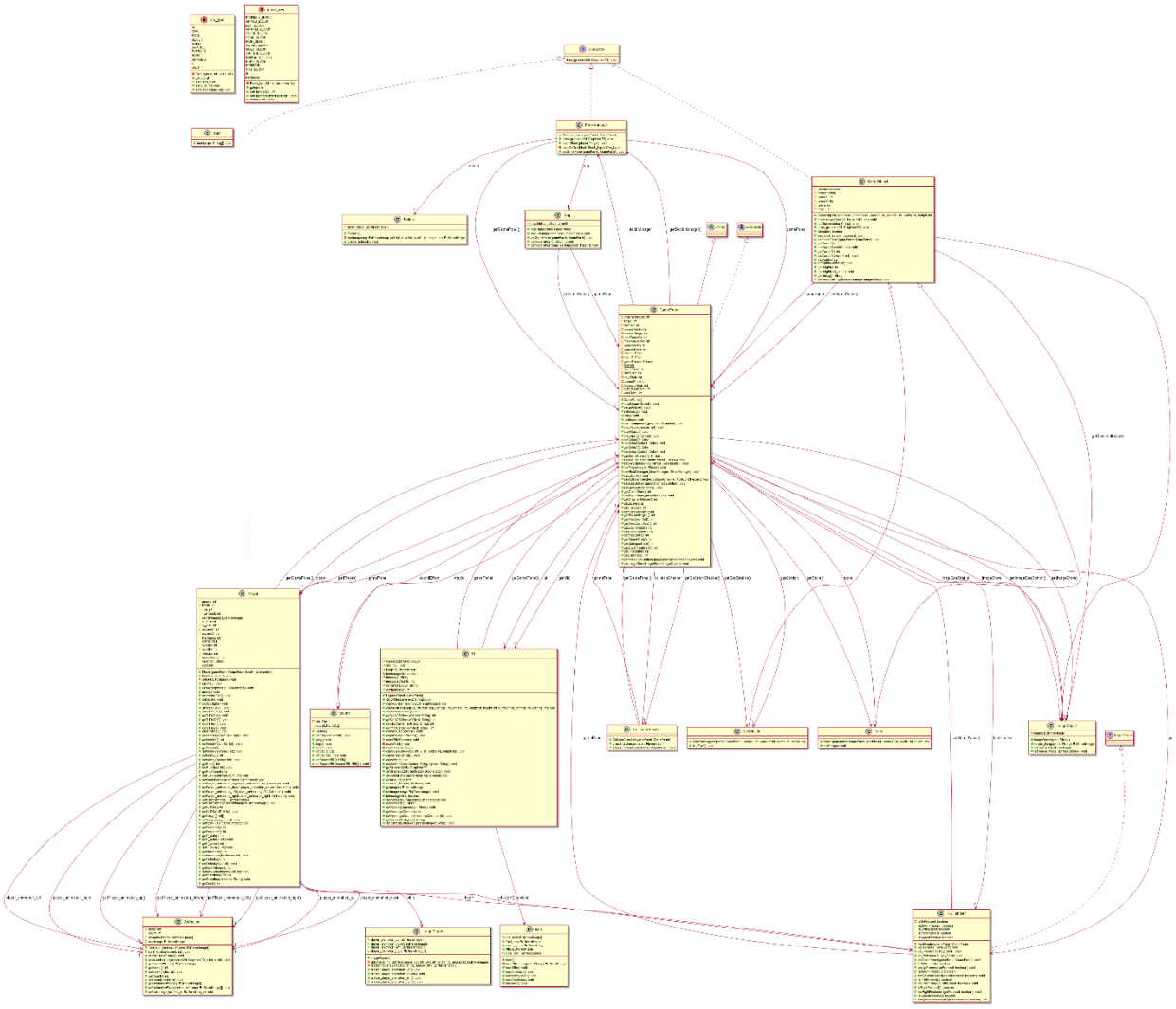






- If you want to stop the game, you can press P (press P again to continue the game). The scene when you lost or win like the two pictures on the right and you can press the spacebar to back to the menu.

Class Diagram



*Noted that Access Modifier Notations are listed below

\+ (public)

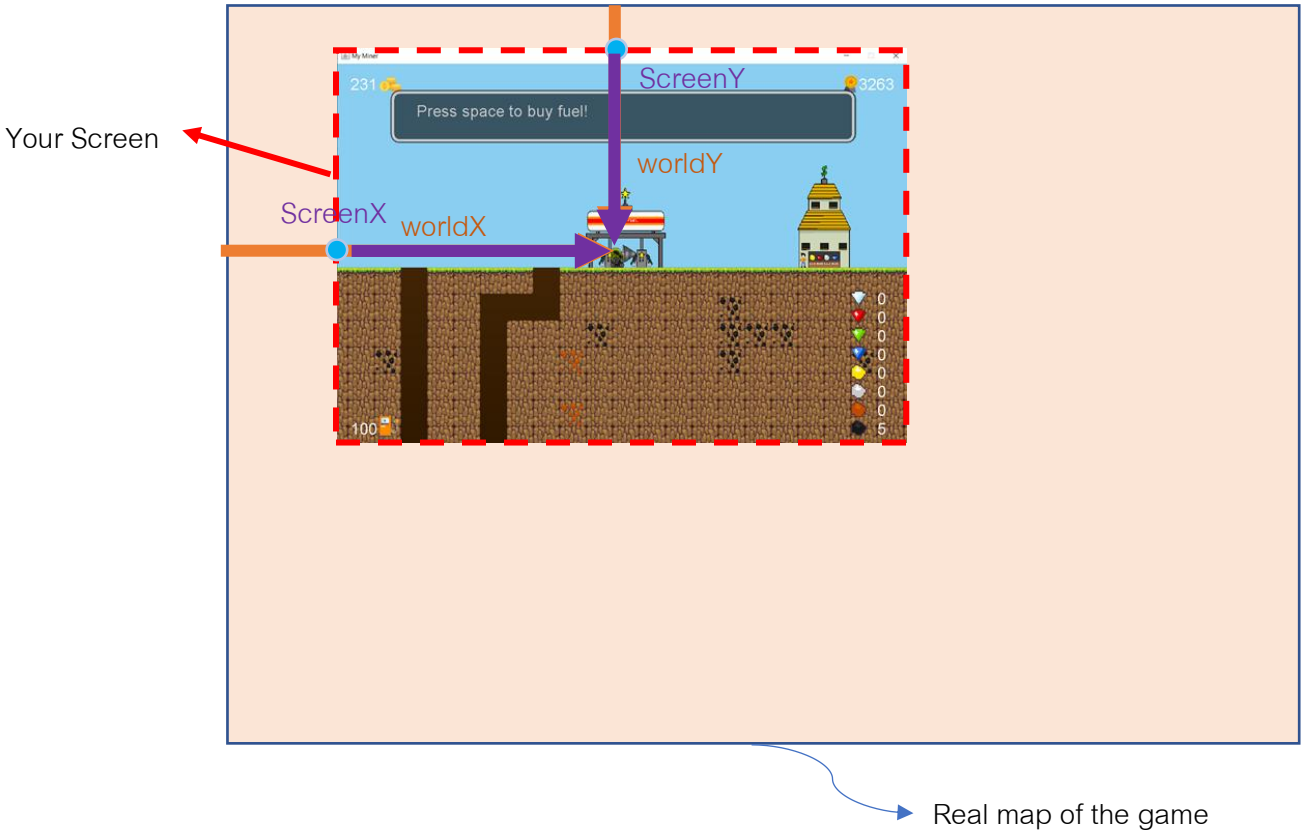
\# (protected)

\- (private)

<u>Underline</u> (static)

Italic (abstract)

Note: In this game, the screen will follow the player (the truck), so it has two types of coordinates as shown in the picture. One is the real coordinate when compare with the real map (We will call worldX and worldY). The another is the coordinate when compare with the screen (called ScreenX and ScreenY).



1. Package block

1.1 Enum Block_type

INVISIBLE_BLOCK(0, 0), GRASS_BLOCK(1, 3), DIRT_BLOCK(2, 3), GRAVEL_BLOCK(3, 4), STONE_BLOCK(4, 10), COAL_BLOCK(5, 4), IRON_BLOCK(6, 6), SILVER_BLOCK(7, 7), GOLD_BLOCK(8, 8), SAPHIRE_BLOCK(9, 8), EMERALD_BLOCK(10, 9), RUBY_BLOCK(11, 9), DIAMOND(12, 9), SKY_BLOCK(13, 0)

This enum is about the kinds of blocks that will show on the screen.

Note: The first value in parenthesis represents the ID of each kind of block and the second is its hardness. For example, Ruby's ID is 11, and Ruby's hardness is 9.

1.1.1 Fields

- int id	The ID of each block.
- int hardness	The hardness of each block.

1.1.2 Constructor

- Block_type(int id, int hardness)	Initialize all fields with the given value.

1.1.3 Methods

+ Getter/ Setter	Getter and setter method of all fields

1.2 class BlockManager implements Drawable

This class is used for managing every block that will show on the screen.

1.2.1 Fields

- GamePanel gamePanel	A game panel of this game
- Texture texture	Containing texture of every type of block
+ Map map	Contain map of this game

1.2.2 Constructor

+ BlockManager(GamePanel gamePanel)	Initialize game panel with the given value. Initialize map and texture.

1.2.3 Methods

+ void draw(Graphics2D graphics2d)	Create the new frame that will show -Set paint of graphics2d with Gradient Paint by using GradientPaint(0, 0, gamePanel.getColor1(), 0, height, gamePanel.getColor2()) -Fill the rectangle with that gradient (width and height is equal to width and height of game panel) -Using while-loop to draw the image and set the coordinate of the block in block_type in every row and every column (block_type is a 2-dimensional array in class Block_type) -Let's worldX = worldCol * gamePanel.getTileSize() worldY = worldRow * gamePanel.getTileSize() (TileSize is a size of one block)

	This step is to change row and column in block_type into the real coordinate of this map game. screenX = worldX - gamePanel.getPlayer().getWorldX() + gamePanel.getPlayer().getScreenX() screenY = worldY - gamePanel.getPlayer().getWorldY() + gamePanel.getPlayer().getScreenY() This is the coordinate of the block in the screen. -If the player is left most of the screen, set screenX = worldX. If the player is on the top (or on the ground in the screen), set screenY = worldY -If the player is now the rightmost of the screen, the screen will stop. (not follow the player anymore) Set the screenX = gamePanel.getScreenWidth() - (gamePanel.getWorldWidth() - worldX) and if the player is at the bottom of the map, then set screenY = gamePanel.getScreenHeight() - (gamePanel.getWorldHeight() - worldY) -Finally, calling method drawImage from graphics2d to set the proper image. We divide it into two cases. The first is a normal case that the screen follows the player. The second is the case that the screen won't follow the player anymore.
+ void deleteBlock(Player player)	When the player arrives at the block, it will destroy that block that can destroy. We must consider 5 cases (Player's direction is UP, DOWN, LEFT, RIGHT, STAND) Let's int entityLeftWorldX = player.getWorldX() int entityRightWorldX = player.getWorldX() + gamePanel.getTileSize() int entityTopWorldY = player.getWorldY();

	int entityButtomWorldY = player.getWorldY() + gamePanel.getTileSize() int entityLeftCol = entityLeftWorldX / gamePanel.getTileSize() int entityRightCol = entityRightWorldX / gamePanel.getTileSize() int entityTopRow = entityTopWorldY / gamePanel.getTileSize() int entityButtomRow = entityButtomWorldY / gamePanel.getTileSize() These are the position of this player. (We assume that if the player goes into the block for 30 pixels, the block will be destroyed.) - If the Player's direction is UP, set entityTopRow = (entityTopWorldY + 30) / gamePanel.getTileSize() and if this block is not INVISIBLE_BLOCK and not SKY_BLOCK. Then adding the ore in this block (if this block has ore inside) in the cargo. Another player's direction is similar to UP, but the player's direction makes the position of this player is different. -If the Player's direction is DOWN, entityButtomRow = (entityButtomWorldY - 30) / gamePanel.getTileSize() and if this block is not INVISIBLE_BLOCK and not SKY_BLOCK. Then adding the ore in this block (if this block has ore inside) in the cargo. -If the Player's direction is LEFT, entityLeftCol = (entityLeftWorldX + 30) / gamePanel.getTileSize() and if this block is not INVISIBLE_BLOCK and not SKY_BLOCK. Then

	adding the ore in this block (if this block has ore inside) in the cargo. -If the Player's direction is RIGHT, entityRightCol = (entityRightWorldX - 30) / gamePanel.getTileSize() and if this block is not INVISIBLE_BLOCK and not SKY_BLOCK. Then adding the ore in this block (if this block has ore inside) in the cargo. -If the Player's direction is STAND and if this block is not INVISIBLE_BLOCK and not SKY_BLOCK. Then adding the ore in this block (if this block has ore inside) in the cargo.
- Ore_type blockToOre(Block_type block)	Change from block type to ore type For example, COAL_BLOCK can change to ore type by calling Ore_type.COAL
+ Getter/setter	Getter and setter method for all fields

2. Package entity

2.1 class Player implements Drawable

The player represents the truck that you can control to collect minerals.

2.1.1 Fields

- int money	The money that the player has. Initialize it as 300 dollars.
- int lvlDrill = 9	It is the hardest of block types that you can drill. (If that block is a rock that you can't drill, it means that the hardness of that rock is more than lvlDrill)
- int fuel = 100	The fuel of the truck that you control

- int fuelCount = 0	Used for counting how many frames to reduce the fuel
- GamePanel gamePanel	The game panel of this games.
+ ImagePlayer skin	Image that uses in animation of this player Initialize it by calling new ImagePlayer()
- Animation player_animation_up	Animation for the player to go up
- Animation player_animation_down	Animation for the player to go down
- Animation player_animation_left	Animation for the player to go left
- Animation player_animation_right	Animation for the player to go right
- BufferedImage currentImage	Current Image of the player
- int x_axis	Position in X-axis of the player on the screen.
- int y_axis	Position in Y-axis of the player on the screen.
- final int screenX	It equals the center of the screen in X-axis (Using as coordinate of the player in the condition that the screen follow the player)
- final int screenY	It equals the center of the screen in Y-axis (Using as coordinate of the player in the condition that the screen follow the player)
- int hardness	The hardness of the block that we want to drill
- int[] cargo	The cargo that can store the minerals. Initialize it with an array with a capacity equal to eight. (Because this game has 8 minerals)
- KeyHandler keyH	Store the value that you press on the keyboard to control the player to move
- int worldX	The coordinate in the X-axis in the real map of the game
- int worldY	The coordinate in the Y-axis in the real map of the game

- int velocity	The velocity of the player
- int maxVelocity	Maximum velocity of the player
- String direction	The direction of the player
- int goal	The achievement to win the game (You must collect money more than or equal the goal to win this game)

2.1.2 Constructor

+ Player(GamePanel gamePanel, KeyHandler keyH)	Initialize gamePanel and keyH with the given value. Calling method loadAnimation() to load the animation Set screenX = gamePanel.getScreenWidth() / 2 - (gamePanel.getTileSize() / 2) Set screenY = gamePanel.getScreenHeight() / 2 - (gamePanel.getTileSize() / 2) Calling method clearCargo() ,setDefaultValues() and setGoal() to initialize the other fields.

2.1.3 Method

+ void loadAnimation()	Initialize player_animation_up, player_animation_down, player_animation_left and player_animation_right (Using method in class ImagePlayer to get the animation for each movement)
- void setDefaultValues()	Set the default value when starting games by the following: worldX = 800

	worldY = 750 vel = 2 direction = "STAND" Set the currentImage equals to player_animation_right.getCurrentFrame() (We require that initially, the car will always turn to the right.)
+ void update()	Calling method move() and decreasefuel() because when the player moves, the fuel will be decreased. Calling method goalUpdate() to random the goal's number If the player is at Gas Station and keyH is Space Pressed, it will call method buyFuel() to buy fuel. If the player is at Store and keyH is Space Pressed, it will call method sellCargo() to sell minerals.
+ void draw(Graphics2D graphics2d)	Similar to the method draw in class Block Manager. Instead of setting the block's coordinate, this method in this class will set the coordinate of the player. However, we need to change the image of the player in each direction. (Using method drawAnimation from class Animation to get the proper image) In the player's direction STAND, we use the method drawImage from graphics2d and use currentImage to show the same image.
+ void move()	This method is about controlling how the player moves. The player's velocity is slower when attacking the soil or rocks. Because we can move the player freely. When we stop to move the player, we have to "snap" the player to the correct position.

	If keyH is UpPressed, it means that the player's direction will be set to "UP". If worldX % 50 <= maxVelocity, it will call snapToTileX() to snap the player to the correct position. (Correct position means the nearest block) According to the fact that the player's velocity is slower when attacking the soil or rocks and if the hardness of the block is less than or equal lvlDrill, we set the player's speed equal to velocity + (lvlDrill / 3) – hardness. Finally, setting the player's position by the following: If player's speed is less than one, then setWorldY(getWorldY() - 1) Otherwise, setWorldY(getWorldY() - speed) Otherwise, calling method goToTileX() (it means that doesn't need to snap) The other cases are similar to UpPressed case. If keyH is DownPressed, it means that the player's direction will be set to "DOWN". If worldX % 50 <= maxVelocity, it will call snapToTileX() and if hardness of block is less than or equal lvlDrill, we will set player's speed equal to velocity + (lvlDrill / 3) – hardness. Finally, setting the player's position by the following: If player's speed is less than one, then setWorldY(getWorldY() + 1) Otherwise, setWorldY(getWorldY() + speed) Otherwise, calling method goToTileX()

	If keyH is LeftPressed, it means that the player's direction will be set to "LEFT". If worldX % 50 <= maxVelocity, it will call snapToTileY() and if hardness of block is less than or equal lvlDrill, we will set player's speed equal to velocity + (lvlDrill / 3) – hardness. Finally, setting the player's position by the following: If player's speed is less than one, then setWorldX(getWorldX() - 1) Otherwise, setWorldX(getWorldX() - speed) Otherwise, calling method goToTileY() If keyH is RightPressed, it means that the player's direction will be set to "RIGHT". If worldX % 50 <= maxVelocity, it will call snapToTileY() and if hardness of block is less than or equal lvlDrill, we will set player's speed equal to velocity + (lvlDrill / 3) – hardness. Finally, setting the player's position by the following: If player's speed is less than one, then setWorldX(getWorldX() + 1) Otherwise, setWorldX(getWorldX() + speed) Otherwise, calling method goToTileY() Otherwise (or don't press anyting) the direction will set to "STAND"
+ void decreasefuel()	If the keyH is UpPressed or DownPressed or LeftPressed or RightPressed, it will decrease the fuel. In this game, we use 60 frames in 1 second. We use fuelCount to count the frame. (When the player moves, the fuel will be decreased 1 litter in every 20 frames)

	If the fuel is less than or equal to zero, it means that you lost the game. So, set GameState of gamepanel to 4 (4 means OutOfFuel state) and then calling method clearCargo() and set WordX to 800 and WorldY to 750.
+ void setGoal()	Using random from java.util.Random Creating an object of the Random class and then calling method nextInt(bound) (bound is the number that we want to random from 0 to bound)
+ void goalUpdate()	If money is more than or equal to the goal, then set game state of the game panel to 5 (5 means win state)
+ void snapToTileX()	Let's variable named number equals to worldX % 50 If number is more than 25, then calling setWorldX((getWorldX() / 50) * 50 + 50) Otherwise, setWorldX((getWorldX() / 50) * 50)
+ void snapToTileY()	Let's variable named number equals to worldY % 50 If number is more than 25, then calling setWorldY((getWorldY() / 50) * 50 + 50) Otherwise, setWorldY((getWorldY() / 50) * 50)
+ void goToTileXX()	Let's variable named number equals to worldX % 50 If number is more than 25, then calling setWorldX(getWorldX() + 1) Else If number is more than zero, then setWorldX(getWorldX() - 1)
+ void goToTileYY()	Let's variable named number equals to worldX % 50 If number is more than 25, then calling setWorldY(getWorldY() + 1) Else If number is more than zero, then setWorldY(getWorldY() - 1)

+ void goToTileX()	The player will go to another tile on X-axis when the player's direction is LEFT or RIGHT If the player's direction is LEFT, then setWorldX(getWorldX() - 1) If the player's direction is RIGHT, then setWorldX(getWorldX() + 1) Otherwise, calling method goToTileXX()
+ void goToTileY()	The player will go to another tile on Y-axis when the player's direction is UP or DOWN If the player's direction is UP, then setWorldY(getWorldY() - 1) If the player's direction is DOWN, then setWorldY(getWorldY() + 1) Otherwise, calling method goToTileYY()
+ void clearCargo()	Using for-loop to set every member in cargo to zero
+ void addTocargo(Ore_type ore)	Calling ore. getId() to get the ID of that ore. In the cargo, increased the number of that ore if the ID of the ore is not equal to -1 (ID = -1 means the blocks is null) To add ore in the cargo, you must call method playSE(4) and getUi().showMessage("+" + ore) by the gamePanel
+ void getWorldX()	Get the real coordinate of the player on the X-axis.
+ void setWorldX(int worldX)	Set the real coordinate of the player on the X-axis. First, get the hardness of the block by calling gamePanel.getCollisionChecker().checkHardness(this) If worldX >= gamePanel.getTileSize() and worldX <= gamePanel.getWorldWidth() - 2 * gamePanel.getTileSize() and hardness <= lvlDrill (These conditions use for checking that the player doesn't stay out of the map), then set worldX with the given value.
+ void getWorldY()	Get the real coordinate of the player on the Y-axis.

+ void setWorldY(int worldY)	Set the real coordinate of the player on the Y-axis. First, get the hardness of the block by calling gamePanel.getCollisionChecker().checkHardness(this) If worldY >= 750 and worldY <= gamePanel.getWorldHeight() - 2 * gamePanel.getTileSize() and hardness <= lvlDrill (These conditions use for checking that the player doesn't stay out of the map), then set worldY with the given value
+ getter/setter	Getter and setter methods of the other fields.

3. Package load.resource

3.1 class Icon

The class is about images that show in this game.

3.1.1 Fields

+ BufferedImage[] ore_icon	Icon of each mineral
+ BufferedImage fuel_icon	The fuel's icon
+ BufferedImage money_icon	The money's icon
+ BufferedImage title	The title's icon
+ BufferedImage goal_icon	The goal's icon

3.1.2 Constructor

+ Icon()	Calling method loadTitle(), loadFuelIcon(), loadMoneyIcon(), loadGoalIcon() and loadIcon() to set all fields

3.1.3 Methods

- BufferedImage loadOtherIcon(String path)	Return the image from the path
+ void loadTitle()	Set the title's image by calling method loadOtherIcon("/image/title.jpg")
+ void loadFuelIcon()	Set the fuel's image by calling method loadOtherIcon("/image/gas-pump.png")
+ void loadMoneyIcon()	Set the money's image by calling method loadOtherIcon(/image/dollar.png)
+ void loadGoalIcon()	Set the goal's image by calling method loadOtherIcon(/image/goalicon2.png)
+ void loadIcon()	Initialize the ore_icon with the array that contains BufferedImage and its capacity of this ore_icon equals eight. (Because there are 8 types of minerals in this game and the index of this array refer to the ID of each type of minerals) Set each channel in ore_icon with an image of that type of mineral.

3.2 class Map

3.2.1 Fields

- GamePanel gamePanel	The game panel of this game
- Block_type[][] mapTileNum	2-dimensional array that contain map of this game

3.2.1 Constructor

+ Map(GamePanel gamePanel)	Set gamePanel with the given value and calling method load_tilemap(gamePanel) to initialize another field.

3.2.2 Method

+ void load_tilemap(GamePanel gamePanel)	Initialize mapTileNum and use try-catch and while-loop to create the image of all blocks that have to show on the screen The map is written in the file name “yod1_new2.txt”. We need to read that file. In that file, we use the number that represents the ID of each type of blocks
+ getter/setter	Getter and setter method of all fields

3.3 class ImageObject

3.3.1 Fields

- BufferedImage image	Image of the object

3.3.2 Constructor

+ ImageObject(String path)	Calling method create_image(path) to initialize image

3.3.3 Method

+ BufferedImage create_image(String path)	Create the image from the path (path refers to the resource of the image)
+ getter/ setter	Getter and setter method of all fields

3.4 class ImagePlayer

One action of the player using four images.

3.4.1 Fields

+ BufferedImage[] player_animation_up	An array that contains image animation for the player when the player goes up
+ BufferedImage[] player_animation_down	An array that contains image animation for the player when the player goes down
+ BufferedImage[] player_animation_left	An array that contains image animation for the player when the player goes left
+ BufferedImage[] player_animation_right	An array that contains image animation for the player when the player goes right

3.4.2 Constructor

+ ImagePlayer()	Calling method create_player_animation_up(), create_player_animation_down(), create_player_animation_left() and create_player_animation_right() to initialize all fields

3.4.3 Method

- BufferedImage splitImage(BufferedImage img, int col, int row, int width, int height)	Get sub-image of the image that calls this method. Using getSubimage method from class BufferedImage
- BufferedImage[] createImagePlayer(String path, int number)	Get the image from the path and split that image with specific row number and put them into an array and return that array
+ void create_player_animation_up()	Set player_animation_up by calling method createImagePlayer which row of this action is one
+ void create_player_animation_down()	Set player_animation_down by calling method createImagePlayer which row of this action is zero
+ void create_player_animation_left()	Set player_animation_left by calling method createImagePlayer which row of this action is three
+ void create_player_animation_right()	Set player_animation_right by calling method createImagePlayer which row of this action is two

3.5 class Sound

3.5.1 Fields

- Clip clip	A sound that loads from URL in soundURL
- URL soundURL[]	This is an array. Initialize it with capacity equals 30. This array contains strings (we will call URL

	because those strings refer to the location of the sound)

3.5.2 Constructor

+ Sound()	Set the each channel of soundURL by the following: soundURL[0] = getClass().getResource("/sound/BGsong.wav") it's background voice. soundURL[1] = getClass().getResource("/sound/button.wav") it's button sound in the MENU scene. soundURL[2] = getClass().getResource("/sound/car.wav") it's car sound. soundURL[3] = getClass().getResource("/sound/fual.wav") it's sound when you refuel at the gas station soundURL[4] = getClass().getResource("/sound/pick.wav") it's sound when you pick the minerals. soundURL[5] = getClass().getResource("/sound/store.wav") it's sound when you sell the minerals.

3.5.3 Methods

+ void setFile(int number)	Open the sound by the given number. The meaning of each number same as described in Constructor
+ void play()	Play the clip
+ void loop()	Play the clip all the time. If the clip ends, this method will make it play again automatically.
+ void stop()	Stop the clip

+ getter/ setter	Getter and setter method for each field

3.6 class Texture

3.6.1 Fields

+ BufferedImage[] array_block	An array that contains every image of every type of block

3.6.2 Constructor

+ Texture ()	Calling method create_allblock()

3.6.3 Methods

- BufferedImage splitImage (BufferedImage img, int col, int row, int width, int height)	Get a sub-image of the image that calls this method. Using getSubimage method from class BufferedImage
+ void create_allblock()	Calling method splitImage to split the image for each type of block and put them into array_block The index of array_block equals to the ID of each type of block

4. Package object

4.1 abstract class GameObject implements Drawable

4.1.1 Fields

# boolean isInside	To check that Is the player is inside the block
# String dialog	The dialog that will show when the player passes the store or the gas station
# GamePanel gamePanel	The game panel of this game
# BufferedImage image	Image of the object
# ImageObject resorceImage	The resource image of the game object
# int worldX	The coordinate in X-axis in the map of the game object
# int worldY	The coordinate in Y-axis in the map of the game object
# int width	Width of the object
# int height	Height of the object

4.1.2 Constructor

+ GameObject(GamePanel gamePanel, int worldX, int worldY, int width, int height)	Initialize all the fields with the given value. For dialog, set it as “”. Calling method checkInside(worldX, worldY) to initialize isInside.

4.1.3 Methods

+ void checkInside(int worldX, int worldY)	To check that Is the player is inside the object

	If worldX - this.worldX more than or equal to zero and worldX - this.worldX <= width - (this.gamePanel.getTileSize() and worldY - this.worldY more than zero and worldY - this.worldY <= height - 0.5 * (this.gamePanel.getTileSize()) , then isInside is true. Otherwise, isInside is false.
+ void setDialog(String dialog)	Set the dialog with the given value
+ void draw(Graphics2D graphics2d)	Draw an image of this game object on the screen. First, get the position on the screen of this game object by initialize screenX with worldX - gamePanel.getPlayer().getWorldX() + gamePanel.getPlayer().getScreenX() and initialize screenY with worldY - gamePanel.getPlayer().getWorldY() + gamePanel.getPlayer().getScreenY() Then, calling drawImage method from graphics2d.
+ getter/ setter	Getter and setter method for the other fields

4.2 GasStation extends GameObject

Gas station is a GameObject.

4.2.1 Constructor

+ GasStation(GamePanel gamepanal , int worldX, int worldY, int width, int height)	Set the attribute of the gas station with the given variables. Set resorceImage = gamePanel.getImageGasStation()

4.2.2 Methods

+ void buyFuel()	Calling method playSE(3) by gamePanel to play soundtrack. While money (from class Player) is more than zero and fuel (from class Player) is less than 100, using while-loop to reduce the money and increase fuel (1 dollar can buy 1 unit of the fuel)

4.3 Store extends GameObject extends GameObject

Store is a GameObject.

4.3.1 Constructor

+ Store(GamePanel gamepanal , int worldX, int worldY, int width, int height)	Set the attribute of the store with the given variables. Set resorceImage = gamePanel.getImageStore()

4.3.2 Methods

+ void sellCargo()	Calling method playSE(5) by gamePanel to play soundtrack. Increase the money (from class Player) by the sum of money that you will gain when selling every mineral in the cargo. (Using the method from Ore_type to get the value of each mineral) Calling method clearCargo() (from class Player) to clear cargo

4.4 enum Ore_type

NULL(-1, 0), COAL(0, 20), IRON(1, 50), SILVER(2, 80), GOLD(3, 150), SAPHIRE(4, 250), EMERALD(5, 400), RUBY(6, 700), DIAMOND(7, 1000)

Note: The first value in parenthesis represents the ID of each kind of mineral and the second is its selling price. For example, Ruby's ID is 6, and Ruby's price is 700.

4.4.1 Fields

- int id	The ID of each kind of mineral
- int value	The selling price of each kind of mineral

4.4.2 Constructor

- Ore_type(int id, int value)	Initialize all fields with the given value

4.4.3 Methods

+ getter/ setter	Getter and setter method of each field

5. Package main

5.1 class Animation

This class uses for making wheels of the player can roll.

5.1.1 Fields

- int index	Index of animationFrame. Let's initialize it as 0
- int count	Count the number of images in animationFrame.
- BufferedImage[] animationFrame	An array that contains images of this animation
- BufferedImage currImage	The current image of that frame

5.1.2 Constructor

+ Animation(BufferedImage[] animationFrame)	Set the animationFrame with the given value Set the currImage equals animationFrame[0]

5.1.3 Method

+ void runAnimation(int speed)	Running the animation, the speed of the running depends on the speed of the player. Let's index plus with speed*speed. If the index is more than frame per second minus with speed, then set index as 0 (go back to the first

	image of animationFrame) and calling create_nextFrame() to create animation
- void create_nextFrame()	Create the animation. Using count to get the image of animationFrame count will be increased by one every time calling this method If count is out of range of animationFrame, set count backing to zero and set currImage equals to animationFrame[count]
+ void drawAnimation(Graphics2D graphics2d, int x, int y)	Draw the animation by using drawImage (method of graphics2d) and calling runAnimation(4) (We will set the speed of the wheels equal to 4)
+ BufferedImage getCurrentFrame()	Getter method for currImage
+ getter/ setter	Getter and setter method for the other fields

5.2 class CollisionChecker

5.2.1 Fields

- GamePanel gamePanel	Game panel of this game

5.2.2 Constructor

+ CollisionChecker(GamePanel gamePanel)	Initialize the field with the given value

5.2.3 Methods

+ int checkHardness(Player player)	This method will predict the hardness of the block that the player might attack. We will divide into 5 cases (according to the player's direction): UP, DOWN, LEFT, RIGHT, STAND For example, if the player's direction is UP, then the row of the player in the game map will change and this method must return the hardness of that block in the UP direction.
+ getter/ setter	Getter and setter method for the field

5.3 class GamePanel extends JPanel implements Runnable

GamePanel is a JPanel and can be runnable. GamePanel likes a hub that initializes and calls many classes to set the game on each screen.

5.3.1 Fields

- final int originalTileSize	Size of each block. originalTileSize = 50
- final int scale	scale = 1
- final int tileSize	tileSize = originalTileSize * scale
- final int screenWidth	screenWidth = 1080 (Width of the screen)
- final int screenHeight	screenHeight = 720 (Height of screen)
- final int maxWorldCol	Max column of the real map = 92
- final int maxWorldRow	Max row of the real map = 135
- final int worldWidth	worldWidth = tileSize * maxWorldCol It's the width of the game map.
- final int worldHeight	worldHeight = tileSize * maxWorldRow It's the height of the game map.

- Color color1	Initialize it with new Color(84, 49, 10) The first color of the background
- Color color2	Initialize it with new Color(48, 25, 0) The second color of the background
- Sound soundEffect	Sound effect
- Sound music	Music in the game
- Thread gameThread	Thread of the game
- KeyHandler keyHandler	Contain data that we press on the keyboard
- Player player	Player of this game. Initialize it by calling a constructor of the player.
- BlockManager blockManager	blockManager of this game. Initialize the block manager to control the block
- UI ui	UI of this game. Initialize UI to call UI to show result in each screen
- CollisionChecker collisionChecker	collisionChecker of this game. Initialize it to check the collision that might happen.
- ImageObject imageGasStation	Image of gas station Initilaize it by calling constructor of ImageObject and set path as "/image/gas_station.png"
- ImageObject imageStore	Image of store Initilaize it by calling constructor of ImageObject and set path as "/image/store.png"
- GasStation gasStation	Gas station object
- Store store	Store object
+ final int fps	Frame per second of this game
- int gameState	The number that represents what state it is

- final int titleState	titleState = 0 It means that if the game state equals 0, then the state is now the titleState.
- final int playState	playState = 1
- final int pauseState	pauseState = 2
- final int dialogueState	dialogueState = 3
- final int outOfFuelState	OutOfFuelState = 4
- final int winState	winState = 5

5.3.2 Constructor

+ GamePanel()	Set Preferred Size with a new dimension that is created from screenWidth and screenHeight Set background with black color Set DoubleBuffered is true Adding key listener Set Focusable is true Calling method initObject() and setupGame() to init the other fields.

5.3.3 Method

+ void startGameThread()	Initialize gameThread and start it
+ void setupGame()	Set gameState = titleState and calling method playMusic(0) to play a music
+ void initObject()	Initialize gasStation with WordX = 1150, WorldY = 650, width and height = tileSize * 3

	Set dialog of gas station is “Press space to buy fuel!” Initialize store with WordX = 1550, WorldY = 600, width = tileSize * 3 and height = tileSize * 4 Set dialog of store is “Press space to sell your cargo for cash!”
+ void run()	Override method run of Thread. Let’s drawInterval = 1000000000 / fps and lastTime = System.nanoTime() and initialize two variable currentTime and delta with zero Looping the thread: Set currentTime = System.nanoTime() Set delta += (currentTime - lastTime) / drawInterval And set lastTime = currentTime If it is more than or equals 1 (It means that it’s time to change the screen), then calling method update() to update the screen and repaint() to paint the screen and minus delta with 1. The delta represents the time of this game.
+ void update()	If gameState equals playState, then calling method player.update() to update the player, blockManager.deleteBlock(player) to delete the block (if the player can), and calling method checkInside of gasStation and store to check that if the player is inside these object.

	(If it is, then the player can refuel or sell minerals)
+ void paintComponent(Graphics graphics)	Overrides javax.swing.JComponent. paintComponent In this method, it will call the “drawScreen” method of UI and method draw of other classes to show the correct screen (depends on game state) If game state is playState or pauseState, then calling method draw of class BlockManager, GasStation, Store, and Player and calling method drawScreen UI to draw the screen Otherwise, calling draw from UI to draw the correct screen
+ void playMusic(int number)	This method is about playing the music. Calling method setFile(number), play() and loop() to play music
+ void stopMusic()	This method will be called to stop the music.
+ void playSE(int number)	This method is about playing the sound effect. Calling method setFile(number), play() and loop() to play sound effect.
+ getter/setter	Getter and setter method of all fields

5.4 class KeyHandler implements KeyListener

5.4.1 Fields

- GamePanel gamePanel	The game panel of this game

- boolean isUpPressed	To check that is the key that press now is the key that controls the player moving up.
- boolean isDownPressed	To check that is the key that press now is the key that controls the player moving down.
- boolean isLeftPressed	To check that is the key that press now is the key that controls the player moving left.
- boolean isRightPressed	To check that is the key that press now is the key that controls the player moving right.
- boolean isSpacePressed	To check that is the key that press now is the spacebar.

5.4.2 Constructor

+ KeyHandler(GamePanel gamePanel)	Set the gamePanel with the given value

5.4.3 Methods

+ void keyTyped(KeyEvent e)	Do nothing
+ void keyPressed(KeyEvent e)	Get the code from the keyboard that we press If we press P if the state is playState, then playing sound effect number 1 and set gameState as pauseState if the state is pauseState, then playing sound effect number 1 and set gameState as playState If we now in the titleState If we press W, then playing sound effect number 1 and commandNum of the UI is decreased by 1 (but if

	commanNum < 0, then set it as 1) (commandNum used for moving arrow “>” up and down) If we press S, then playing sound effect number 1, but the commandNum of the UI will increase by 1 (but if commanNum > 1, then set it as 0) If we press SPACE, it means that we select that menu. If commandNum is equal to 0, then set the screen as playscreen and call the method playSE(4) (to play sound effect number 4) and set player by initializing a new player. Also initializing the blockManager and set the blockManager to the screen. Finally, set game state as play state If commandNum is equal to 1, then calling System.exit(int) to close the program If we are now in the play state Set that Press key to control the player as the following: - "Press W" = move the truck upward (Set isUpPressed as true, if we press W) - "Press S" = move the truck downward (Set isDownPressed as true, if we press S) - "Press A" = the truck move left (Set isLeftPressed as true, if we press A) - "Press D" = the truck move right. (Set isRightPressed as true, if we press D) -“Press SPACE”, then set isSpacePressed as true -“Press ESCAPE KEY”, then exit the program If we are now in state OutOfFuelState or WinState()

	If we press SPACE, then set game state as title state
+ void keyReleased(KeyEvent e)	Get the code from the keyboard that we press When we release, it will call this method. For example, if we used to press K. When we release, this method will set isUpPressed to false because key K means moving upward. For key S, A, D, and SPACE are similar to key P.
+ getter/setter	Getter and setter method of all fields

5.5 class UI

5.5.1 Fields

- Graphics2D graphics2d	Graphic of the game
- GamePanel gamePanel	Game panel of this game
- Font arial_30	Font that we use in this game Initialize it with font's name Arial, style of this type is PLAIN and size of this front is 30.
- BufferedImage image	Image that uses in the screen
- boolean isMessageOn	Boolean for checking that the object has the massage on when the player passes at it.
- String message	The message that shows when you collect the minerals in the soil
- int messageCounter	Initialize it as zero It uses for counting time that the message will display on the screen when you collect the minerals in the soil
- String currentDialogue	The string that will show above the object (gas station and store)

+ int commandNum	The arrow that shows in the title screen
+ Icon resIcon	Resource icon that will show on the screen

5.5.2 Constructor

+ UI(GamePanel gamePanel)	Set the gamePanel with the given value

5.5.3 Methods

+ void showMessage(String text)	This method is called when the minerals are collected. Set the massage as text and isMessageOn is true
+ void drawScreen(Graphics2D graphics2d)	First, set graphics2d with the given value and set the font with arial_30, and set color as WHITE Next, draw the proper screen (depending on the game state) by calling the proper method. For example, if the game state is play state, then calling method drawPlayScreen()
+ void drawPicture(BufferedImage image, int xImage, int yImage, int width, int height, int number, int xString, int yString)	Calling method drawImage (to set the image and icon on the screen) and drawstring (to set the string on the screen)
+ void drawPlayScreen()	-Drawing the play screen that shows on page 2 -Using method name drawPicture to set the picture and string on the screen -If isMessageOn is true, then it will show the message on the screen for a period of time (while messageCounter is not more than 30)

	Normally, this message will show on the right of the player. But if the player is on the rightmost on the screen, this message will be set to show on the left of the player. (The messageCounter is increased by 1 in every second) -If the player is inside the gas station or store, the screen will show a dialogue screen.
+ int getSizeOfWindowX(String text)	Return the width of string bounds by calling the method graphics2d.getFontMetrics(). getStringBounds(text, graphics2d).getWidth()
+ int getSizeOfWindowY(String text)	Return the height of string bounds by calling the method graphics2d.getFontMetrics(). getStringBounds(text, graphics2d).getHeight()
+ int getXforCenterText(String text)	Return the coordinate in the X-axis of text that will make the test stay in the center of the page
+ int getYforCenterText(String text)	Return the coordinate in the X-axis of text that will make the test stay in the center of the page
+ void drawPauseScreen()	Set the pause screen like on page 5.
+ void drawDialogueScreen()	Set the dialog that will show on page 4. It will show when the player passes the object
+ void drawTitleScreen()	Set the title screen like on page 2. Using commandNum to control the movement of the arrow If commandNum = 0, then the arrow points to PLAY and calling method drawStart() to get the string about how to select each menu and method name drawSubWindow to create sub-window.

	If commandNum = 1, then the arrow points to Tutorial and calling method drawTutorial() to get the string about how to play this game and method name drawSubWindow to create sub-window.
+ void drawStart()	Generate string about how to select each menu
+ void drawTutorial()	Generate string about how to play this game
+ void drawSubWindow(int x, int y, int width, int height)	Create sub-window with round reactangle
+ void drawRunOutFual()	Draw OutOfFuel State by calling method name drawWinorLose
+ void drawWin()	Draw Win State by calling method name drawWinorLose
+ void drawWinorLose(String status, String option)	Create Win state or OutOfFuel state (These two state same format, but have different strings to show)
+ getter/ setter	Getter and setter method for all fields

5.6 interface Drawable

5.6.1 Methods

+ void draw(Graphics2D graphics2d)	Set the image and draw the other components

5.7 class Main

5.7.1 Methods

+ void main(String[] args)	- Initialize the variable name window which types JFrame - Set DefaultCloseOperation of window as JFrame.EXIT_ON_CLOSE

	- Set resizable as false - Set title as “My Miner” - Initialize game panel - Adding game panel to window - Calling method pack() of the window - Calling method setLocationRelativeTo(NULL) of window - Set visible of the window as true - Calling method setupGame() and startGameThread() to start the game