Raft 2D Simultor

Created by

Ponrawoot Khamkahnongngam 6330349521
Nanthicha Makjinda 6330282021

2110215 Programming Methodology
Semester 1 Year 2021 Chulalongkorn University

Raft 2D Simulator

Introduction

Raft 2D Simulator is the survival game inspired by Raft with 2-dimensional design. The concept of this game is to fix the shipwreck by collecting the essential items. The Shipwreck is the only hope you can get out of this island.During the game, you have to get items and be careful of your life point. The more lifepoints you lose, The slower you win.

I can tell you some essential information, there is a shipwreck located in a cave!!!

Rules

Map

There are mainly 5 zones you need to know.

- middle island zone: you will start in this zone, if you have 0 life point, it always respawn. If you feel annoyed, don't forget to consume.
- metal zone; metals are located in the top-left island you need to sail to there, but watch out for the eagle that can attack you.
- cultivated zone: There are 2 areas in the game. You can plant mango trees or pine trees only in this zone.
- shipwreck zone: This is the island with the huge cave containing shipwreck, you can fix shipwreck in the zone
- Fish zone: this is a normal sea area, but you can find fish only in this zone.

Objects

In this game, most objects are refreshed when they are clear for each object type in game, excepting palm trees (be always same position). they will random the numbers and positons.

- Palm tree : It requires any axe to be cutted, giving 3 woods and 3 leaves, Don't need to plant (wait for growing).
- Mango tree : It requires stone axe or metal axe, giving 2 woods, 2 leaves, 1 fruit, and 2 mango seeds. You need to plant in the cultivated area.
- Pine tree : It requires only a metal axe, giving 5 woods, 2 leaves, 1 fruit, and pinecones. You need to plant in the cultivated area.
- Bird : It requires any spear to be hunted, giving 1 bird and 2 feathers.
- Fish : It requires stone spear or metal spear. giving 1 fish.
- Eagle : It requires only a metal spear. giving 2 birds. 1 eagle head, 3 feathers, It always attacks you if you get to the top-left island.
- Plastic : No requirement. giving 1 plastic
- Stone : No requirement. giving 1 stone.
- Scrap : It requires any shovel. giving 1 scrap.
- Metal : It requires only a metal shovel. giving 1 metal.

Basic Control

You can control the player by 4 keys like most pc games (W→ turn/move up, A→ turn/move left, S→turn/move down, D→ turn/move right). There are a few conditions checking your movement. First,you can't move to the rocks or the cave (located in the bottom right-side of the map). Absolutely, you can't move to an area which isn't blank. In the whole game, you can move to land (including the cultivated area) without any conditions, but sea area is prohibited if you don't have a raft (must craft first).

First, you will have 20 life points, you can lose one by cutting trees, hunting animals, and eagle attacking. You can get life points from consuming.moreover, you can get more max life point from doing mission

In the beginning, you will get the stone axe to use,the current weapon will be reduced by 1 if you remove the object. you can get the other from crafting in inventory. don't forget to check your current weapon in the top bar.

Inventory

There are 2 tabs in this window. The first one shows the amount of each item you have, if you have no number of something, it shows only one name of the item, but the items from market trading (there is only one for each item) show the image in slot if you have. Next, there is managing weapon slots to choose your weapon in different types and material or to set free in weapon slots.

The last one tab, there is a list of craftable things. you should check the amount of items to be enough before crafting.

Mission

The objective of the mission is to give a reward if you collect the required items sufficiently. you can do the same mission more one times, but the number will be increased sequentially

Market

This gives the essential things to fix shipwreck by trading like a mission you can do only one time per one type-item.

Shipwreck

You can win this game by doing this, However, you can still play after winning. you need to get enough items to fix the shipwreck. The important condition is to stay around the cave where there is shipwreck inside (any cells that are adjacent to the cave).

Guide

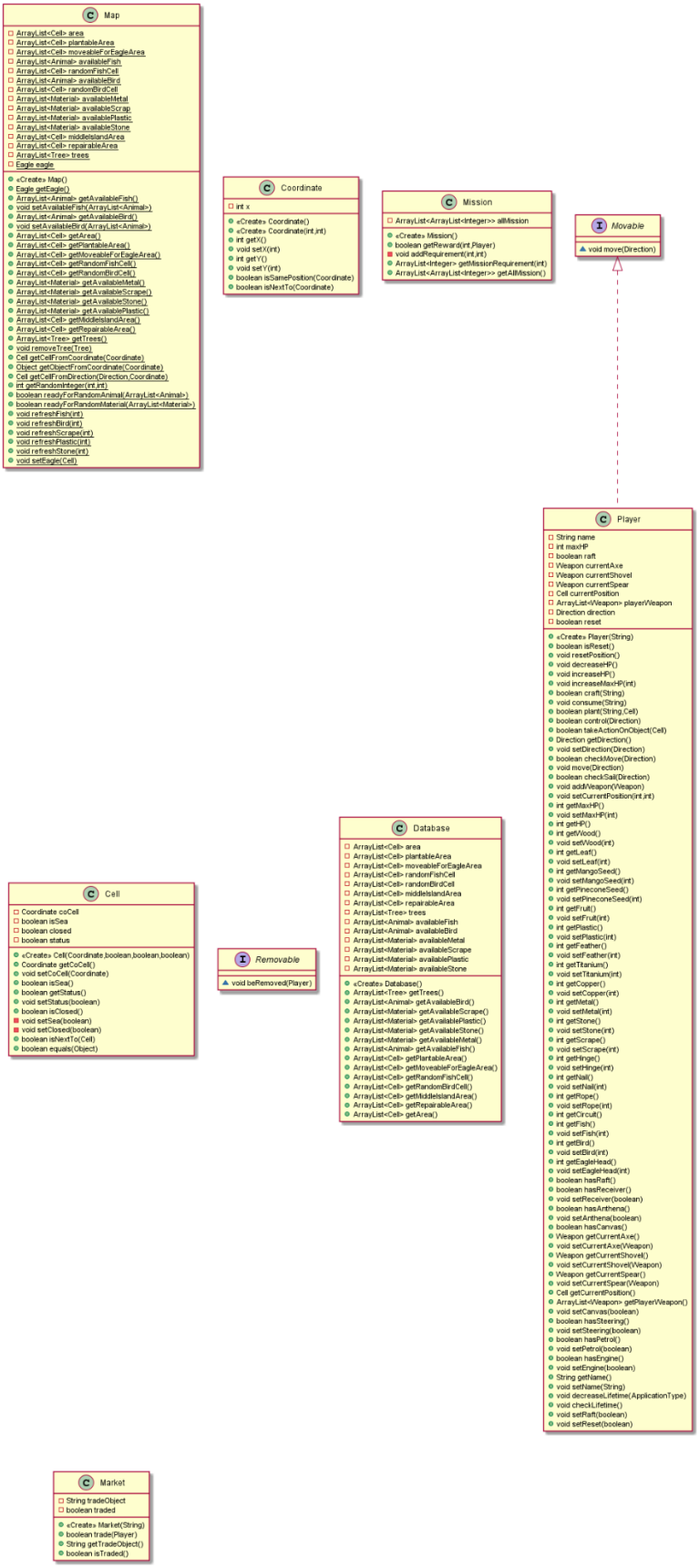
You can read the full information by this window.

Don’t forget, if you get doubt during the game, reading from information that will pop up on the right top bar every time you take action, Have fun :).

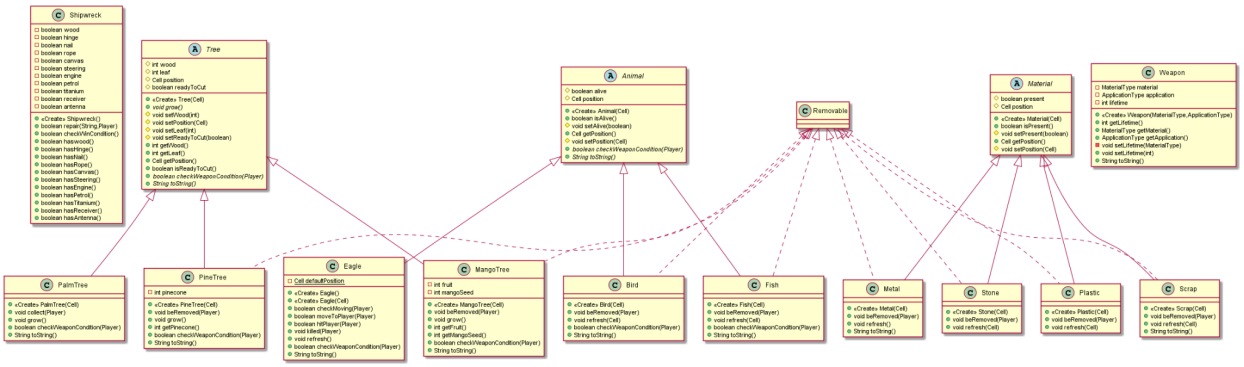
Details

Class Diagram

- Package game



- **Package object**



1. Package game.base

1.1. Enum Direction

This enum represents direction. It contains the following values: LEFT, RIGHT, UP, and DOWN.

1.2. Interface Movable

This interface defines a method for Object that can move.

+ void move(Direction direction)	This method is called when the object moves in a *direction*.

1.3. Interface Removable

This interface defines a method for Object that can be removed by the player.

+ void beRemoved(Player player)	This method is called when the player wants to remove an object.

1.4. Class Coordinate

This class represent a coordinate (x, y)

1.4.1. Fields

- int x	the position of axis X
- int y	the position of axis Y

1.4.2. Constructors

+ Coordinate()	set x to 9 and set y to 12
+ Coordinate(int x, int y)	set related field with given parameters *x* and *y*

1.4.3. Methods

+ boolean isSamePosition(Coordinate other)	Return true if *other* and this is the same position. Otherwise, return false.
+ boolean isNextTo(Coordinate other)	Return true if *other* is next to this position.
+ getters/setters	Getters/Setters for all fields.

1.5. Class Database

This class generates all cells in game map.

2. Package game

2.1. Class Cell

This class represents a cell in game map.

2.1.1. Fields

- Coordinate coCell	coordinate of this cell
- boolean isSea	a boolean to keep whether this cell is sea or not
- boolean closed	a boolean can indicate whether a player can move to or not.
- boolean status	a boolean can indicate whether a player can move to or not, but it can change unlike closed

2.1.2. Constructor

+ Cell(Coordinate coCell, boolean isSea, boolean closed, boolean status)	set related field when given parameters

2.1.3. Methods

+ boolean isNextTo(Cell cell)	Return true if *cell* is next to this. Otherwise, return false. Note: use isNextTo method from class Coordinate
+ boolean equals(Object obj)	Override method equals. Return true if *obj* coordinate is the same coordinate with this. Otherwise, return false.
+ void setStatus(boolean status)	Setters for status field If closed field is true, status would be false. Otherwise, set with a given parameter.
+ getters/setters	Getters/Setters for remaining fields

2.2. Class Map

This class represents the map of the game.

2.2.1 Fields

- ArrayList<Cell> area	list of all cells in the game.
- ArrayList<Cell> plantableArea	list of all cells that can be cultivated
- ArrayList<Animal> availableFish	list of all fishes in the game.
- ArrayList<Cell> randomFishCell	list of all cells which are used for refreshing fishes in game.
- ArrayList<Animal> availableBird	list of all birds in the game.
- ArrayList<Cell> randomFishCell	list of all cells which are used for refreshing birds in game.
- ArrayList<Material> availableMetal	list of all metals in the game.
- ArrayList<Material> availableScrap	list of all scraps in the game.
- ArrayList<Material> availablePlastic	list of all plastics in the game.
- ArrayList<Material> availableStone	list of all stones in the game.
- ArrayList<Cell> middleIslandArea	list of all cells that are in the middle island zone.
- ArrayList<Cell> repairableArea	list of all cells that are around the shipwreck zone.
- ArrayList<Tree> trees	list of all trees in the game.
- Eagle eagle	Eagle initialized in the game.

2.2.2. Constructor

+ Map()	Initialize all fields Note: Initialize by using the getter in class Database except eagle.

2.2.3. Methods

+ Cell getCellFromCoordinate (Coordinate other)	Return Cell that have the **same coordinate** to other.
+ Object getObjectFromCoordinate	Return Object that have the **same**

(Coordinate coordinate)	**coordinate** to coordinate.
+ Cell getCellFromDirection (Direction direction,Coordinate coordinate)	Return Cell that have the **same coordinate** to the coordinate changed by direction.
+ int getRandomInteger (int maximum, int minimum)	Return random integer from range between
+ boolean readyForRandomAnimal (ArrayList<Animal> animals)	Return true if all animals in the list are clear,Otherwise, return false.
+ boolean readyForRandomMaterial (ArrayList<Material> material)	Return true if all materials in the list are clear,Otherwise, return false.
+ void refreshFish(int random)	Check readyForRandomAnimal(), If true refresh fishes by random.
+ void refreshBird(int random)	Check readyForRandomAnimal(), If true refresh birds by random.
+ void refreshScrap(int random)	Check readyForRandomMaterial(), If true refresh scraps by random.
+ void refreshPlastic(int random)	Check readyForRandomMaterial(), If true refresh plastics by random.
+ void refreshStone(int random)	Check readyForRandomMaterial(), If true refresh stones by random.
+ void setEagle(Cell next)	set position of the eagle to next.
+ void removeTree(Tree tree)	remove tree from ArrayList<Tree>
+ setters	Setters for availableFish and availableBird fields
+ getters	Getters for all fields

2.3. Class Player implements Movable

This class represents the player.

2.3.1. Fields

- String name	player name
- int maxHp	maxHP
- int HP	player HP
- int wood	amount of wood

- int leaf	amount of leaf
- int mangoSeed	amount of mango seed
- int pinecone	amount of pinecone
- int fruit	amount of fruit
- int plastic	amount of plastic
- int feather	amount of feather
- int titanium	amount of titanium
- int copper	amount of copper
- int metal	amount of metal
- int stone	amount of stone
- int scrap	amount of scrap
- int hinge	amount of hinge
- int nail	amount of nail
- int rope	amount of rope
- int circuit	amount of circuit
- int fish	amount of fish
- int bird	amount of bird
- int eagleHead	amount of eagle head
- boolean raft	boolean keep whether player have raft or not
- boolean receiver	boolean keep whether player have receiver or not
- boolean antenna	boolean keep whether player have antenna or not
- boolean canvas	boolean keep whether player have canvas or not
- boolean steering	boolean keep whether player have steering or not
- boolean petrol	boolean keep whether player have petrol or not

- boolean engine	boolean keep whether player have engine or not
- Weapon currentAxe	player current axe
- Weapon currentShovel	player current shovel
- Weapon currentSpear	player current spear
- Cell currentPosition	player current position
- ArrayList<Weapon> playerWeapon	list of player weapon
- Direction direction	direction of player
- boolean reset	a boolean that check player ready to ready or not.

2.3.2. Constructor

+ Player(String name)	set - maxHP to 10 and HP to maxHP - int fields to 0 - boolean fields to false - current weapon to null except current axe set to STONE AXE - init playerWeapon and add current axe to playerWeapon - set currentPosition to cell with coordinate (9,12) - set direction to DOWN

2.3.3. Methods

+ void resetPosition()	set player current position to cell with coordinate(9, 12)
+ void decreaseHP()	decrease player HP if HP is more than 0. Otherwise, show warning. Note: use method showHPWarning from TopBar
+ void increaseHP()	increase player HP only if HP is less than maxHp.
+ void increaseMaxHP(int n)	increase maxHp

+ boolean craft(String object)	craft the *object*
+ void consume(String object)	consume the *object* then increaseHP
+ boolean plant(String object, Cell position)	plant *object* in the *position* and set the position status to false.
+ boolean control(Direction direction)	check direction of player with direction, If they are the same, return false. Otherwise, change player direction to this parameter, and then return true.
+ boolean takeActionOnObject(Cell cell)	call method to do an object on this cell varies by class of object.
+ boolean checkMove(Direction direction)	check the cell that is in front of the player whether can move or not. If the player can move, return true. Otherwise, return false.
+ void move(Direction direction)	if checkMove(direction) or checkSail(direction) are true, change position of player according to direction.
+ boolean checkSail(Direction direction)	check the raft and check the cell that is in front of the player whether can move or not.In addition,If can, return true. Otherwise, return false.
+ void addWeapon(Weapon weapon)	add *weapon* to playerWeapon
+ void decreaseLifetime(ApplicationType type)	decrease lifetime of current weapon which has ApplicationType same as given parameter
+ void checkLifetime()	check current weapon lifetime. If lifetime equal to 0, remove it from playerWeapon and set current weapon to null.
+ void setMaxHP(int maxHP)	if *maxHP* is more than 0 set this maxHP to *maxHP*
+ getters/setters	Getters/Setters for remaining fields

2.4. Class Market

This class represents a market for trading objects where each object can be traded only once.

2.4.1. Field

- String tradeObject	name if trade object
- boolean traded	boolean keeps whether tradeObject has been traded or not

2.4.2. Constructor

+ Market (String tradeObject)	set related field to given parameter

2.4.3. Methods

+ boolean trade(Player player)	return true if can trade tradeObject with *player*, and set traded to true. Otherwise, return false.

2.5. Class Mission

This class represents a mission for player.

2.5.1. Field

- ArrayList<ArrayList<Integer>> allMission	arraylist that keep all requirement for each mission

2.5.2. Constructor

+ Mission()	init allMission and add start requirement

2.5.3. Methods

+ boolean getReward(int i, Player player)	return true if *player* can get the reward from mission *i* and addRequirement to mission *i*. Otherwise, return false.
- void addRequirement(int missionNumber, int amount)	add requirements to the mission with given *amount*.

3. Package object.base

3.1. Enum ApplicationType

This enum represents the application of the weapon. It contains the following values: AXE, SHOVEL, and SPEAR.

3.2. Enum MaterialType

This enum represents the material of the weapon. It contains the following values: WOOD, STONE, and METAL.

4. Package object

4.1. Class Weapon

4.1.1. Fields

- MaterialType material	material of this weapon
- ApplicationType application	application of this weapon
- int lifetime	lifetime of this weapon

4.1.2. Constructor

+ Weapon(MaterialType material, ApplicationType application)	set material and application fields with given parameter and use setter to set lifetime from material

4.1.3. Methods

+ void setLifetime(MaterialType material)	If material is WOOD, set lifetime to 5. If material is STONE, set lifetime to 10. If material is METAL, set lifetime to 20.
+ getters/setters	Getters/Setters for all remaining fields
+ String toString()	return string material + " " + application

4.2. Abstract Animal

This class is base class of all animals. Each animal has its own amount of result after remove and weapon condition.

4.2.1. Fields

# boolean alive	boolean keeps whether this animal is alive or not.
# Cell position	position of this animal.

4.2.2. Constructor

+ Animal(Cell position)	set alive to true, set position to given *position* and set position status to false.

4.2.3. Methods

+ abstract boolean checkWeaponCondition(Player player)	check whether player's current spear can be used to kill this animal. Each animal has a different condition.
+ abstract String toString()	Override method toString return string represent name of animal
+ getters/setters	Getters for all fields
# setters	Setters for all fields

4.3. Class Fish extends Animal implements Removable

This class represents fish which is an animal that can be removed by any SPEAR.

4.3.1. Constructors

+ Fish(Cell position)	initialize fish with given *position*

4.3.2. Methods

+ void beRemoved(Player player)	check player current spear, check whether player is in the position next to this position and check whether this fish is alive then add 1 fish to player, decrease player current spear lifetime, decrease player HP, set alive to false and change position status to true.
+ void refresh(Cell position)	check whether this fish is alive or not if not change it to true, set the position to the given *position* and set position status to false.
+ boolean checkWeaponCondition (Player player)	return true if player's current spear is STONE or METAL SPEAR. Otherwise, return false.
+ String toString()	return "fish"

4.4. Class Bird extends Animal implements Removable

This class represents a bird which is an animal that can be removed by STONE SPEAR or METAL SPEAR.

4.4.1. Constructors

+ Bird(Cell position)	initialize bird with given *position*

4.4.2. Methods

+ void beRemoved(Player player)	check player current spear, check whether player is in the position next to this position and check whether this bird is alive then add 1 bird and 2 feathers to player, decrease player current spear lifetime, decrease player HP, set alive to false and change position status to true.
+ void refresh(Cell position)	check whether this bird is alive or not if not change it to true, set the position to the given *position* and set position status to false.
+ boolean checkWeaponCondition (Player player)	return true if player's current spear is not null. Otherwise, return false.
+ String toString()	return "bird"

4.5. Class Eagle extends Animal

This class represents a eagle which is an animal that can be removed by METAL SPEAR.

4.5.1 Fields

- final Cell defaultPosition	the respawn position of eagle

4.5.2 Constructor

+ Eagle()	set position by defaultPosition
+ Eagle(Cell cell)	set position by cell

4.5.3 Methods

+ boolean checkMoving(Player player)	check whether player is in metal zone true or false; return this check.
+ boolean moveToPlayer(Player player)	move to player. calculate y-axis first.
+ boolean hitPlayer(Player player)	damage to player 1 life point

+ void killed(Player player)	set alive to false when condition about wanpon (METAL SPEAR) is true.
+ void refresh()	set alive to true and set position to defaultPosition
+ boolean checkWeaponCondition(Player player)	check condition about weapon (METAL SPEAR)
+ String toString()	return "eagle." + "\nYou got 2 bird, 1 eagle head, 3 feathers,"

4.6. Abstract Tree

This class is base class of all trees. Each has its own amount of result when remove and weapon condition.

4.6.1. Fields

# int wood	amount of wood that player will obtain when cut.
# int leaf	amount of leaf that player will obtain when cut.
# Cell position	position of this tree
# boolean readyToCut	boolean keeps whether this tree is ready to cut or not.

4.6.2. Constructor

+ Tree(Cell position)	set ready to cut to false, set position to given *position* and set position status to false.

4.6.3. Methods

+ abstract void grow()	called when growing tree
+ abstract boolean checkWeaponCondition(Player player)	check whether player's current axe can be used to cut this tree or not. Each tree has a different condition.
+ abstract String toString()	Override method toString return name of the tree and amount of objects that player will obtain when cut the tree.

# setters	Setters for all fields
+ getters	Getters for all fields

4.7. Class PalmTree extends Tree

This class represents a palm tree which is a tree that cannot be removed. When the player cuts a palm tree it will turn into a stump.

4.7.1. Constructor

+ PalmTree(Cell position)	initialize palm tree with given position, set wood to 3, and set leaf to 3

4.7.2. Methods

+ void collect(Player player)	Called when player want to cut the tree check player current axe, check this palm tree is ready to cut and check whether player is in the position next to this position then add amount of wood and leaf from fields to player, set readyToCut to false, set wood and leaf field to 0, decrease player HP, and decrease player current axe lifetime.
+ void grow()	check this palm tree is not ready to cut then set wood and leaf to 3 and set readyToCut to true
+ boolean checkWeaponCondition (Player player)	return true if player's current axe is not null.
+ String toString()	return "palm tree." + " You got 3 woods, 3 leaves,"

4.8. Class MangoTree extends Tree implements Removable

This class represents a mango tree which is a tree that can be removed by STONE AXE or METAL AXE. Player can plant it again by using mango seed in plantable areas.

4.8.1. Fields

- int fruit	amount of fruit that player will obtain when cut.
- int mangoSeed	amount of mango seed that player

	will obtain when cut.

4.8.2. Constructor

+ MangoTree(Cell position)	initialize mango tree with given position, set wood to 2, and set leaf to 2, set fruit to 1, and set mangoSeed to 2

4.8.3. Methods

+ void beRemoved(Player player)	Called when player want to cut the tree check player current axe, check this mango tree is ready to cut and check whether player is in the position next to this position then add amount of wood, leaf, fruit, mangoSeed from fields to player, set readyToCut to false, set wood, leaf, fruit, mangoSeed fields to 0, decrease player HP, decrease player current axe lifetime, remove this tree from Map, and set the position status to true.
+ void grow()	check this mango tree is not ready to cut then set wood and fruit to 1, set leaf and mangoSeed to 2, set readyToCut to true, and set position status to false.
+ boolean checkWeaponCondition (Player player)	return true if player's current axe is STONE or METAL AXE. Otherwise, return false.
+ String toString()	return "mango tree." + " You got 1 wood, 2 leaves, 1 fruit, 1 mango seed"
+ getters	getters for fruit and mango fields

4.9. Class PineTree extends Tree implements Removable

This class represents a pine tree which is a tree that can be removed by METAL AXE. Player can plant it again by using pinecone in plantable areas.

4.9.1. Fields

- int pinecone	amount of pinecone that player will obtain when cut.

4.9.2. Constructor

+ PineTree(Cell position)	initialize pine tree with given position, set wood to 5, and set leaf to 2, and set pinecone to 2

4.9.3. Methods

+ void beRemoved(Player player)	Called when player want to cut the tree check player current axe, check this pine tree is ready to cut and check whether player is in the position next to this position then add amount of wood, leaf, and pinecone from fields to player, set readyToCut to false, set wood, leaf, and pinecone fields to 0, decrease player HP, decrease player current axe lifetime, remove this tree from Map, and set the position status to true.
+ void grow()	check this pine tree is not ready to cut then set wood, leaf and pinecone to 2, set readyToCut to true, and set position status to false.
+ boolean checkWeaponCondition (Player player)	return true if player's current axe is METAL AXE. Otherwise, return false.
+ String toString()	return "pine tree." + " You got 2 woods, 2 leaves, 2 pinecone,"
+ int getPineconeSeed()	getters for pinecone field

4.10. Abstract Material

This class is base class of all materials. Each material has its own result when remove.

4.10.1. Fields

# boolean present	boolean keeps whether this material is present in Map or not.
# Cell position	position of this material

4.10.2. Constructors

+ Material(Cell position)	set position to given *position*, set position status to false, and set present to true.

4.10.3. Methods

+ getters/setters	Getters/Setters for all fields

4.11. Class Metal extends Material implements Removable

This class represents a metal which is a material that can be removed by METAL SHOVEL.

4.11.1 Constructor

+ Metal(Cell position)	initialize metal with given position

4.11.2. Methods

+ void beRemoved(Player player)	check material of player current shovel to be METAL, check whether player is in the position next to this position, and check this metal is present then add 1 metal to player, change present to false, decrease player HP, decrease player current shovel lifetime, and set position status to true.
+ void refresh()	check that this metal is not present then change present to true and set position status to false.
+ String toString()	return "metal"

4.12. Class Stone extends Material implements Removeable

This class represents a stone which is a material that can be removed.

4.12.1 Constructor

+ Stone(Cell position)	initialize stone with given position

4.12.2. Methods

+ void beRemoved(Player player)	check if this stone is present then add 1 stone to player, change present to false, decrease player HP, and set position status to true.

+ void refresh(Cell position)	check if this stone is not present then change present to true, set position to given position, and set position status to false.

4.13. Class Scrap extends Material implements Removable

This class represents a scrap which is a material that can be removed by any SHOVEL.

4.13.1 Constructor

+ Scrap(Cell position)	initialize scrap with given position

4.13.2. Methods

+ void beRemoved(Player player)	check if this scrap is present, check player current shovel is not null, and check whether player is in the position next to this position then add 1 scrap to player, change present to false, decrease player HP, decrease player current shovel lifetime, and set position status to true.
+ void refresh(Cell position)	check if this scrap is not present then change present to true, set position to given position, and set position status to false.
+ String toString()	return "scrap"

4.14. Class Plastic extends Material implements Removeable

This class represents a plastic which is a material that can be removed.

4.14.1 Constructor

+ Plastic(Cell position)	initialize plastic with given position

4.14.2. Methods

+ void beRemoved(Player player)	check if this plastic is present and check whether player is in the position next to this position then add 1 plastic to player, change present to false, decrease player HP, and set position status to true.
+ void refresh(Cell position)	check if this plastic is not present

	then change present to true, set position to given position, and set position status to false.

4.15. Class Shipwreck

This class represents a shipwreck. Player should repair shipwreck to win the game.

4.15.1. Fields

- boolean wood	boolean keeps whether wood has been repaired or not.
- boolean hinge	boolean keeps whether hinge has been repaired or not.
- boolean nail	boolean keeps whether nail has been repaired or not
- boolean rope	boolean keeps whether rope has been repaired or not
- boolean canvas	boolean keeps whether canvas has been repaired or not
- boolean steering	boolean keeps whether steering has been repaired or not
- boolean engine	boolean keeps whether engine has been repaired or not
- boolean petrol	boolean keeps whether petrol has been repaired or not
- boolean titanium	boolean keeps whether titanium has been repaired or not
- boolean receiver	boolean keeps whether receiver has been repaired or not
- boolean antenna	boolean keeps whether antenna has been repaired or not

4.15.2. Constructor

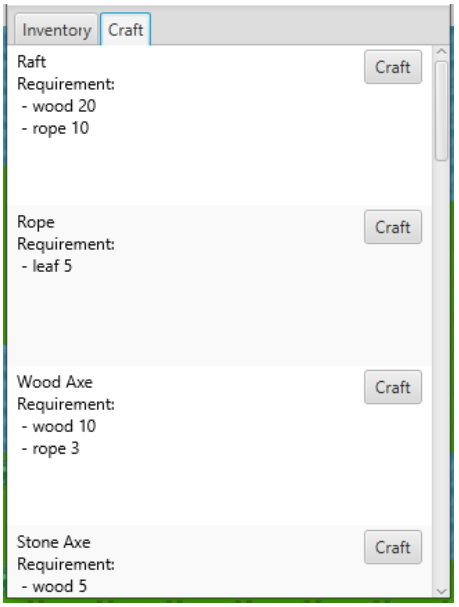
+ boolean repair(String object, Player player)	repair *object* part of shipwreck by checking *object* from player inventory
+ boolean checkWinCondition()	return true if all parts of shipwreck have been repaired. Otherwise

	return false.
+ getters	Getters for all fields

5. Package component.base

5.1. Class CraftPane extends ListView<BorderPane>

This class is ListView that represents craft pane in the inventory box in the Bottom bar.



5.1.1 Fields

- String[] craftObjects	string array that keep name of craft objects

5.1.2. Constructor

+ CraftPane (Player player, InventoryPane inventoryPane)	- initialize super class
	- initialize craftObjects field with method initCraftObjects
	- create and add BorderPane for all craftObjects with following details
	  - set left to craft details text
	  - set right to craft button that calls method craft(object) from *player* and update inventoryPane when clicked.
	  - set preferred height to 110

5.1.3. Methods

- void initCraftObjects()	initialize craft objects
- String getRequirement(int i)	generate the details of craft object that is at index *i*
+ void showWarning()	show warning information if cannot craft

5.2. Class CurrentWeaponPane extends GridPane

This class represents pane that show player current weapons in the Top bar.

5.2.1. Fields

- Pane currentAxe	pane that show current axe
- Pane currentShovel	pane that show current shovel
- Pane currentSpear	pane that show current spear

5.2.2. Constructor

+ CurrentWeaponPane(Player player)	- initialize super class - initialize all fields with method initCurrentAxe, initCurrentShovel, initCurrentSpear - setHgap, setPadding - add text and all fields

5.2.3. Methods

- void setImage(Weapon weapon)	set background and image in the current *weapon* pane
- void initCurrentAxe(Player player)	initialize current axe
- void initCurrentShovel(Player player)	initialize current shovel
- void initCurrentSpear(Player player)	initialize current spear

- void update(Player player)	update all pane to *player* current weapon

5.3. Class GameDisplayCell extends Pane

This class represents pane of the game map cell. This class will be used in class GameDisplay.

5.3.1. Fields

- Cell cell	cell in GameDisplay
- ImageView imageView	ImageView for this cell

5.3.2. Constructor

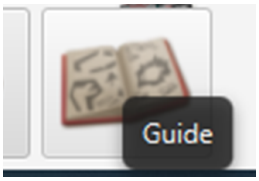
+ GameDisplayCell(int x,int y)	initialize game

5.3.3. Methods

+ void SetImageViewBlank()	clear image view of the pane
+ void SetImageView(String string)	set image view of the pane related to given parameter
+ Cell getCell()	Getter for cell field

5.4. Class GuideButton extends Button

This class is a button that will show a guide book when clicked. The guide button is represented in Bottom bar.



5.4.1. Constructor

+ GuideButton()	- set image view - set fit width and height to 60 - set cusor to HAND - set tooltip with setTooltip method - set to show the guide when

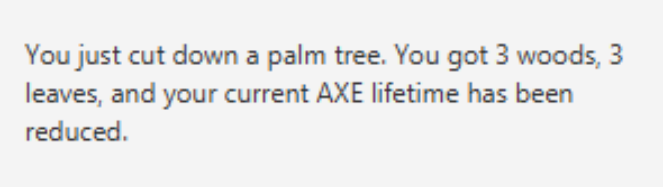
	clicked

5.4.2. Method

- void setTooltip()	set tooltip with "Guide" text

5.5. Class InformationPane extends Label

This class is a Label that shows the information when player takes action in the game.



5.5.1. Constructor

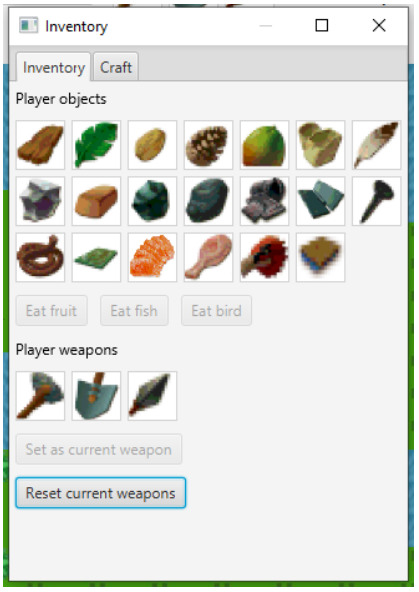
+ InformationPane(Player player)	- initialize super class - set text to "Ready !!" - set wrap text - set preferred width and set padding

5.5.2. Methods

+ void update(Player player, Direction direction)	called to update text when *player* moves with given *direction*. Note: The text will change if player moves to the direction that cannot move to.
+ void update(Player player, Object object)	called to update text when *player* takes action to a tree, an animal, or a material. Note: The text will represent the different details of each action.
+ void update(Cell cell, boolean planted, String tree)	called to update text when *player* plants a *tree*.
+ void update(boolean hit)	called to update text when player is *hit* by the eagle.
- String genInformText(boolean check, Object object, ApplicationType applicationType)	use to generate some information text.

5.6. Class InventoryButton extends Button

This class is a button that shows the inventory box TabPane. TabPane contains InventoryPane and CraftPane. This button is represented in Bottom bar.



5.6.1. Constructor

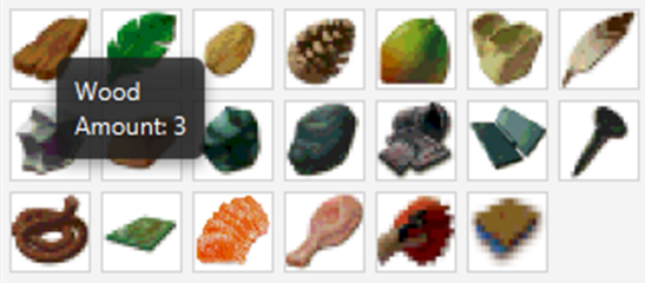
+ InventoryButton()	- set image view - set fit width and height to 60 - set cusor to HAND - set tooltip with setTooltip method - set to show TabPane that contains InventoryPane and CraftPane when clicked.

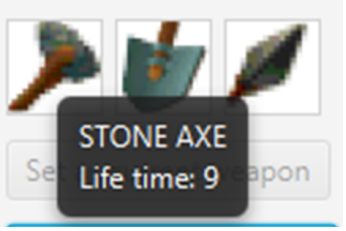
5.6.2. Method

- void setTooltip()	set tooltip with "Inventory" text

5.7. Class InventoryCell extends Pane

This class is a pane that shows player objects and their details. It is used in InventoryPane.





5.7.1. Fields

- String name	name of the object in the pane
- Object object	object that show by this pane
- int amount	amount of the object

5.7.2. Constructor

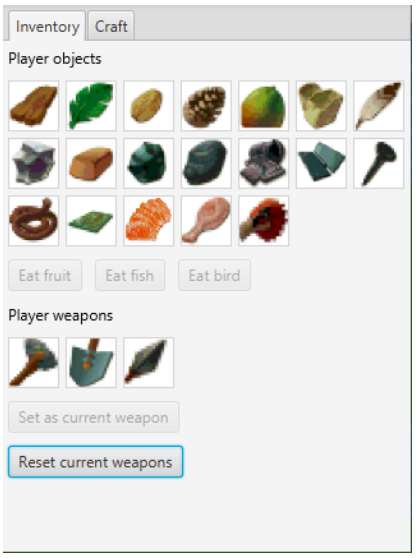
+ InventoryCell(String name, Object object, int amount)	- initialize super class - set all fields with given parameters - set height, width, and padding - set border - set tooltip - set image

5.7.3. Methods

+ void setImage()	set image of this pane
+ String toString()	return string that will use in tooltip
+ void highlight()	change border color toDARKBLUE
+ void unhighlight()	change border color to LIGHTGRAY
+ void setTooltip()	set tooltip with text from method toString
+ void setAmount	setter for amount field
+ Object getObject	getter for object field

5.8. Class InventoryPane extends VBox

This class is a VBox that contains inventory box, eat buttons, weapon box, set current weapon button, and reset current weapon button.



5.8.1. Fields

- GridPane objectPane	GridPane that show all objects from player
- ObservableList<InventoryCell> objects	list that keep all InventoryCell which represent in objectPane
- GridPane weaponPane	GridPane that show all weapon from player
- ObservableList<InventoryCell> weapons	list that keep all InventoryCell which represent in weaponPane
- HBox eatButtons	HBox that contains 3 eat buttons
- Button setWeaponButton	set current weapon button
- Button resetWeaponButton	reset current weapon button
- Weapon selectedWeapon	the weapon that player want to set it as current weapon

5.8.2. Constructor

+ InventoryPane(Player player)	- initialize all fields with methods below - set spacing and padding - add all fields

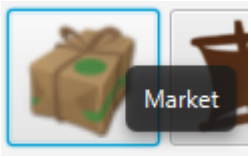
5.8.3. Methods

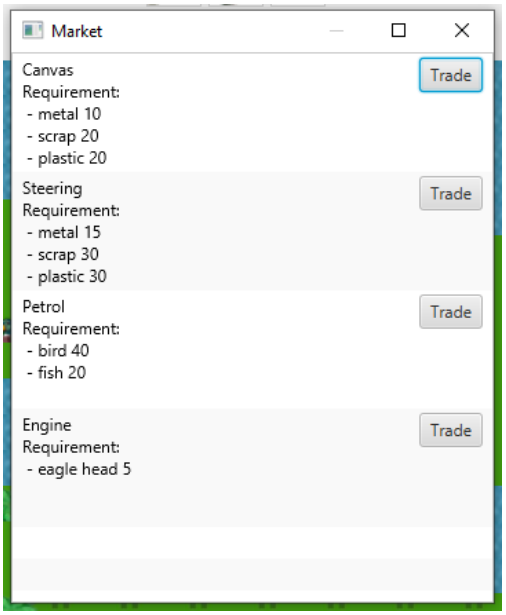
- void initObjects(Player player)	initialize objectPane and objects
- void initWeapons(Player player)	initialize weaponPane and weapons
- void resetHighlight()	set all InventoryCell in weapons to unhighlight
- void initEatButtons(Player player)	initialize eatButtons with 3 eat buttons (fish, bird, fruit)
- void initSetWeaponButton(Player player)	initialize set as current weapon button
- void initResetWeaponButton (Player player)	initialize reset current weapon button it will set all current weapons to null
+ update(Player player)	called to update pane when the

	player objects is change
- void showInform()	showing information when player cannot eat because of full HP

5.9. Class MarketButton extends Button

This class is a button that will show ListView of all markets. It is represented in Bottom bar.





5.9.1. Field

- Market[] markets	list of all markets

5.9.2. Constructor

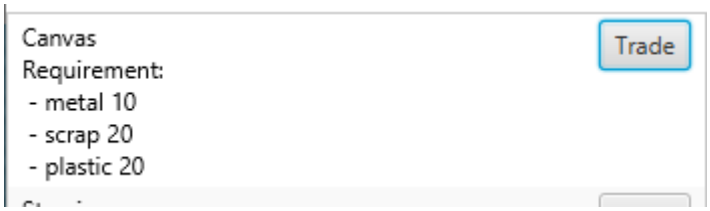
+ MarketButton()	- init all market in markets - set image view - set size - set cursor to HAND - set tooltip - set to show ListView of all MarketPane when clicked

5.9.3. Method

- void setTooltip()	set tooltip with "Market" text

5.10. Class MarketPane extends BorderPane

This class is BorderPane which is shown in ListView when clicking MarketButton.



5.10.1. Fields

- Button tradeButton	trade button
- Text text	text to show details of market

5.10.2. Constructor

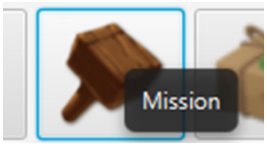
+ MarketPane(Player player, Market market)	- set text with setText method - initialize trade button, it will use the method trade in *market* when clicked. - set left to text, right to tradeButton

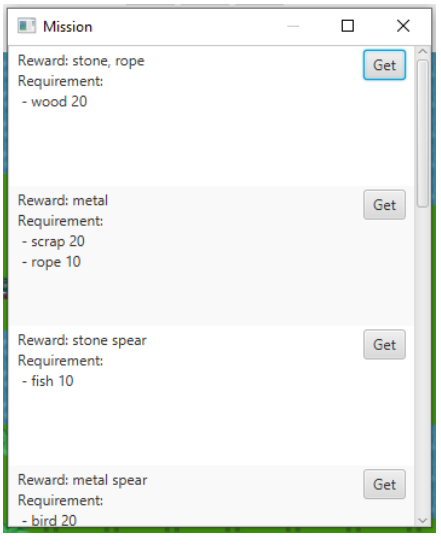
5.10.3. Methods

- void setText(String tradeObject, Player player)	set text with details of market
+ void showWarning()	show warning when player cannot trade

5.11. Class MissionButton extends Button

This class is a button that will show list of all missions when clicked. It is represented in Bottom bar.





5.11.1. Field

- Mission mission	mission

5.11.2. Constructor

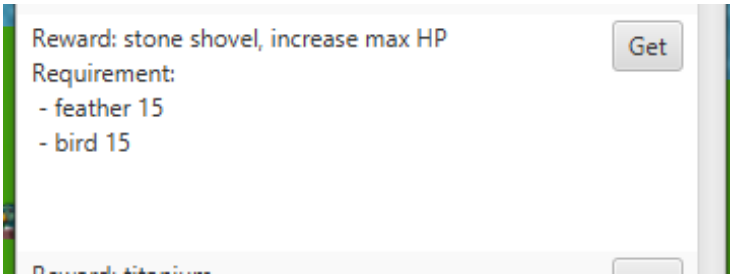
+ MissionButton()	- init mission - set image view - set size - set cursor to HAND - set tooltip - set to show ListView of all MissionPane when clicked

5.11.3. Method

- void setTooltip()	set tooltip with "Mission" text

5.12. Class MissionPane extends BorderPane

This class is BoarderPane which is shown in ListView when clicking MissionButton. It contains mission details and get button.



5.12.1. Fields

- Mission mission	mission
- Label label	label with mission details

5.12.2. Constructor

+ MissionPane(Mission mission, int missionNumber, Player player)	- initialize super class - set this mission to given mission

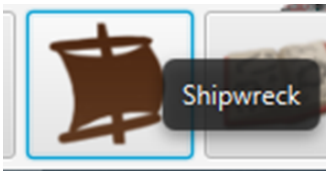
	- set preferred height to 110 - set label by using method below - create "Get" button and make it use method getReward from mission when clicked - set left with label and right with get button

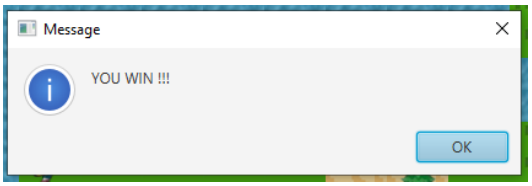
5.12.3. Methods

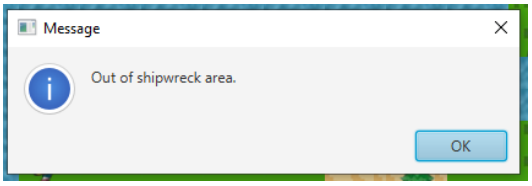
- void setLabel(int missionNumber)	set label with details of mission with *missionNumber*
+ void showWarning()	show warning when cannot get reward from mission

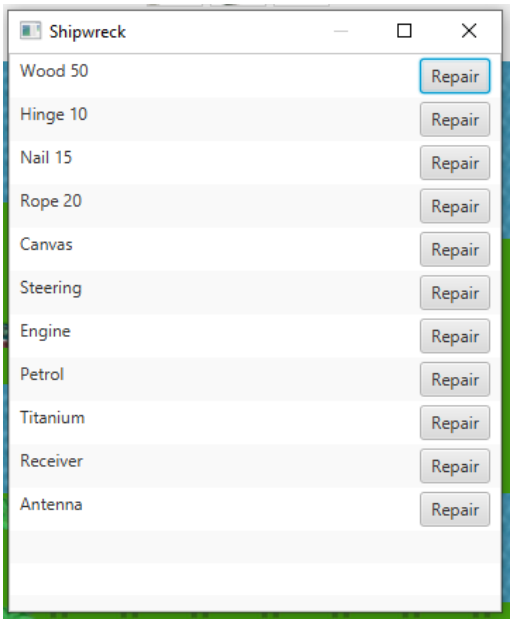
5.13. Class ShipwreckButton extends Button

This class is a button that will show list of all shipwreck parts that need to be repaired when clicked. It is represented in Bottom bar.









5.11.1. Field

- Shipwreck shipwreck	shipwreck

5.11.2. Constructor

+ ShipwreckButton()	- init shipwreck - set image view - set size

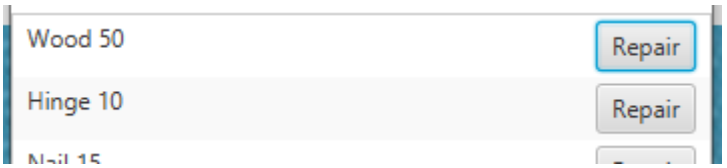
	- set cursor to HAND - set tooltip - set to show ListView of all ShipwreckPane or winning message or warning message when clicked

5.11.3. Method

- void setTooltip()	set tooltip with "Shipwreck" text

5.14. Class ShipwreckPane extends BorderPane

This class is BoarderPane which is shown in ListView when clicking ShipwreckButton. It contains the name of part and repair button.



5.14.1. Fields

- Label label	label contains name or details of part of shipwreck
- Button repairButton	repair button

5.14.2. Constructor

+ ShipwreckPane(String repairPart, Player player, Shipwreck shipwreck)	- initialize super class - init label and set label by using method below - init repair button and set button access by using method below - set repair button to call method repair from *shipwreck* when clicking - set left to label and right to repair button

5.14.3. Methods

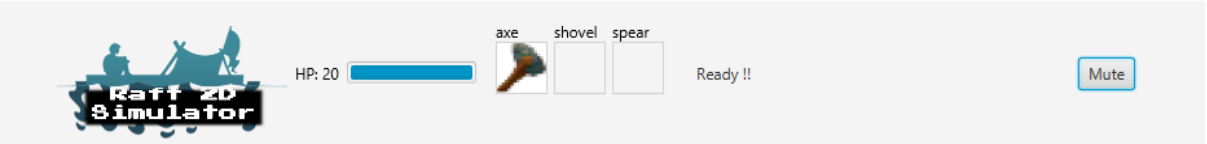
- void setLabel(String repairPart)	set label text
- void setButtonAccess(String repairPart, Shipwreck shipwreck)	set repair button to disable if player already repaired *repairPart*

+ void showWarning()	show warning when cannot repair shipwreck
+ void showWinning()	show winning if player win the game

6. Package component

6.1. Class TopBar extends FlowPane

This class is FlowPane that contains game logo, Hp text, HP progress bar, current weapon pane, information pane, and mute button.



6.1.1. Fields

- ImageView gameLogo	game logo
- ProgressBar hp	player HP progress bar
- Text hpText	HP text
- CurrentWeaponPane weaponPane	pane that show player current weapon
- InformationPane informationPane	pane that show message when player take the action
- Button muteButton	mute button

6.1.2. Constructor

+ TopBar(Player player)	- set paddinge, hgap, vgap - init all fields and add them to children

6.1.3. Methods

+ void setHp(Player player)	update hp progress bar with player hp and max hp
+ boolean showHpWarning(Player player)	return true if player hp is equal to 0 and show warning
+ getters	Getters for current weapon pane and information pane

6.2. Class GameDisplay extends GridPane

This class is a GridPane that shows the display of game map by using GameDisplayCell.

6.3. Class BottomBar extends FlowPane

This class is a FlowPane that contains inventory button, mission button, market button, shipwreck button and guide button.

6.3.1. Fields

- InventoryButton inventoryButton	inventory button
- MissionButton missionButton	mission button
- MarketButton marketButton	market button
- ShipwreckButton shipwreckButton	shipwreck button
- GuideButton guideButton	guide button

6.3.2.Constructor

+ BottomBar(Player player)	- set padding, hgap, vgap - init all fields and add them to children

6.4. Class RootPane extends VBox

This class is VBox that contains TopBar, GameDisplay, and BottomBar

6.4.1. Fields

- TopBar topBar	top bar
- GameDisplay gameDisplay	game display
- BottomBar bottomBar	bottom bar

6.4.2. Constructor

+ RootPane(Player player)	- initialize super class - init all fields and add to children

6.4.3. Methods

+ void redraw(Cell newCell, Cell cell,	set image in cell blank, and Add

String string)	image in newCell by checking string
+ static void redrawTreeStump(Cell cell)	set image to tree stump if object in this cell is palm tree

7. Package application

7.1. Class Main extends Application

7.1.1. Fields

- RootPane rootPane	root pane
- Player player	player
- ThreadMain threadMain	thread main
- AudioClip themeSound	theme song sound
- AudioClip walkingSound	walking sound
- AudioClip actionSound	action sound
- AudioClip warningSound	warning 0 hp sound
- AudioClip[] sound	list of all sound

7.1.2. Methods

+ void start(Stage primaryStage)	The main entry point of the JavaFX application.
+ void main(String[] args)	An entry point of the application.
- addEventListener(Scene scene)	method that add event listener to *scene*

7.2. Class ThreadMain

This is a class that handles threads in the game.

7.2.1. Methods

+ void refreshObject(Object object)	check the type of object, and then refresh this type of object including check whether it is ready.
+ boolean activateEagle(Player player)	return true, if player is in the metal zone, then eagle move to player and hit player if it can. return false when player is out of metal zone
+ void setGrow(Object object)	set this tree object from seed to tree by grow().