Pokemopoly

Nat Silprasert 6730149721

Jomphol Trongpanich 6730063321

1.Overview

Pokemopoly is a turn-based digital board game built with Java and JavaFX. It draws inspiration from classic Monopoly but replaces property mechanics with Pokémon-themed systems such as catching, battling, evolving, and using items.

- Player Count: Supports 2–4 players.
- Main Objective: Finish the game with the highest total assets(Coins, Pokémon cards, badge bonuses)

1. Game Start and Player System

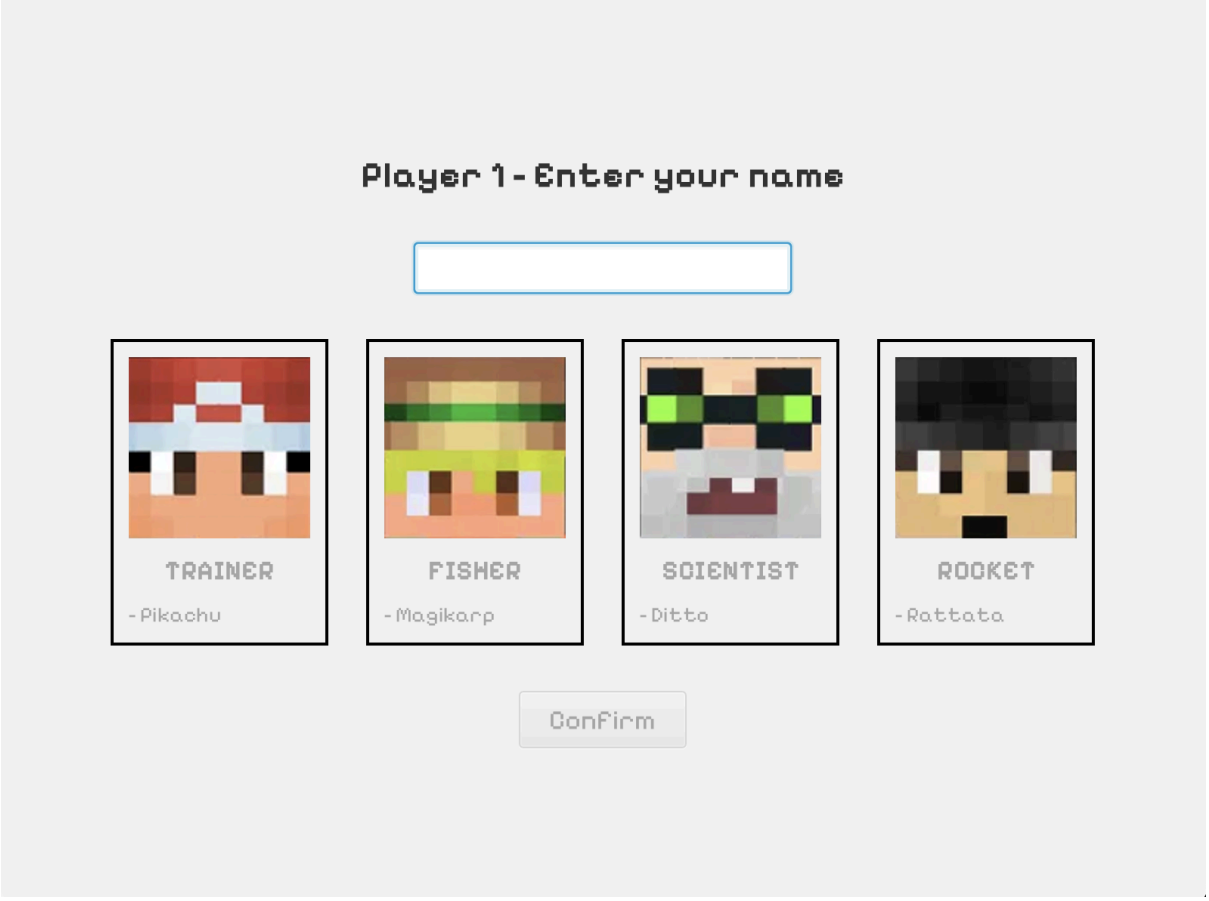
1.1 Game Start

At the beginning of the game, each player selects a Profession, which determines their starting Pokémon. After choosing a profession, each player receives:

- 2 random item cards
- 4 Pokéballs
- 10 coins
- Maximum Team Capacity: Up to 6 Pokémon
- Maximum Item Capacity: Up to 4 items

1.2 Player System

The profession chosen provides a different early-game advantage based on the Pokémon's abilities:



Profession	Starting Pokémon
Trainer	Pikachu
Fisher	Magikarp

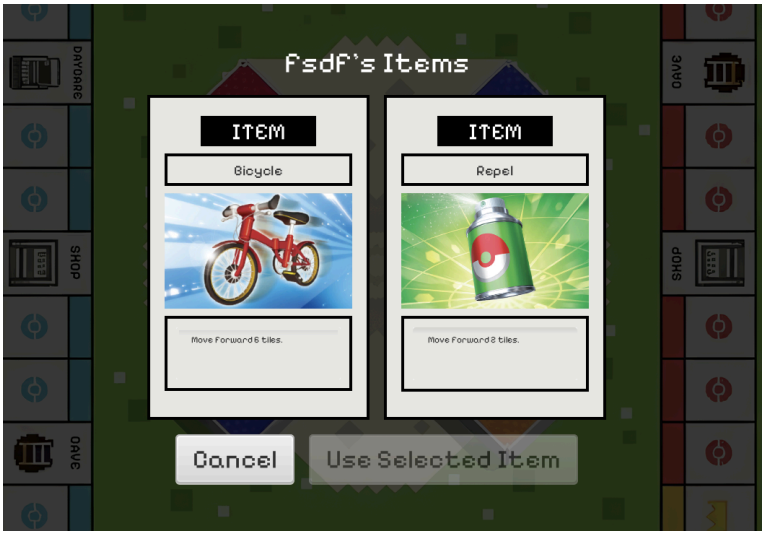
Scientist	Ditto
Rocket	Rattata

2. Core Gameplay

In a player's turn, they can choose one of two actions:

1. Roll the dice to move across the board.

2. Use an item before rolling to gain a strategic advantage.



2.1 Board and Tile System

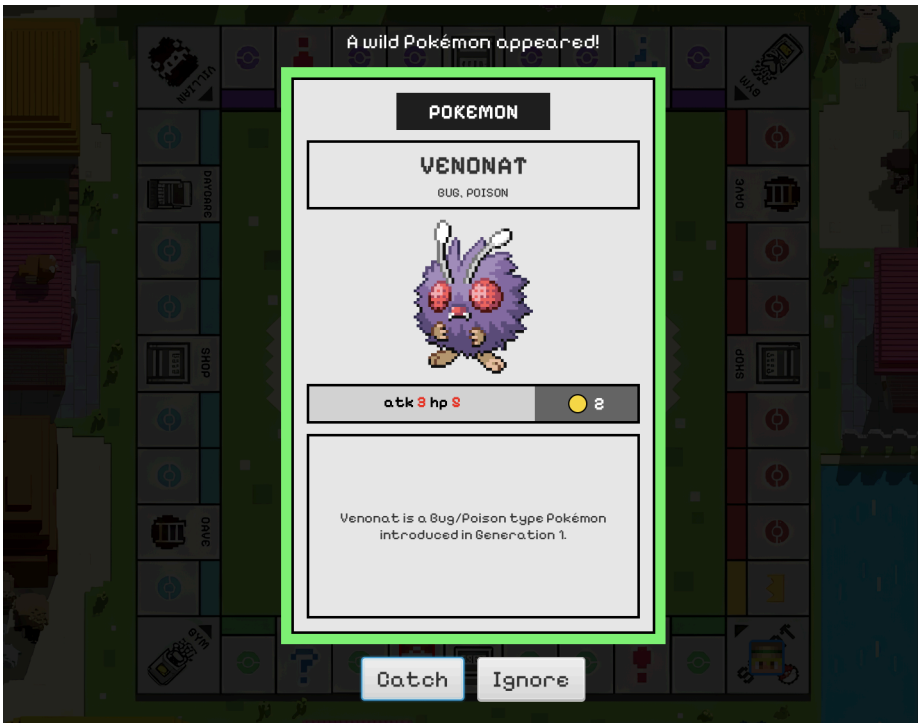
The board consists of 40 tiles, each with a special effect:

Tile Type	Effect
Start Tile	Passing this tile gives the player the option to sell Pokémon from their team

	for coins.
Grass Tile	Allows players to catch wild Pokémon, with the Pokémon depending on the tile's color (5 colors/difficulty tiers).
Event Tile	Player draws a random Event Card which may give buffs, debuffs, or special effects.
Item Tile	Player draws a random Item Card.
City Tile	Players can buy items and extra Pokéballs.
Daycare Tile	Allows players to evolve a Pokémon on their team.
Cave Tile	Acts as a trap. The player is stuck inside and skips their next turn.
Battle Tile (Boss Tiles)	Initiates a battle with either Gym Leader 1, Gym Leader 2, or a Villain.

2.2 Catching Pokémon Mechanics

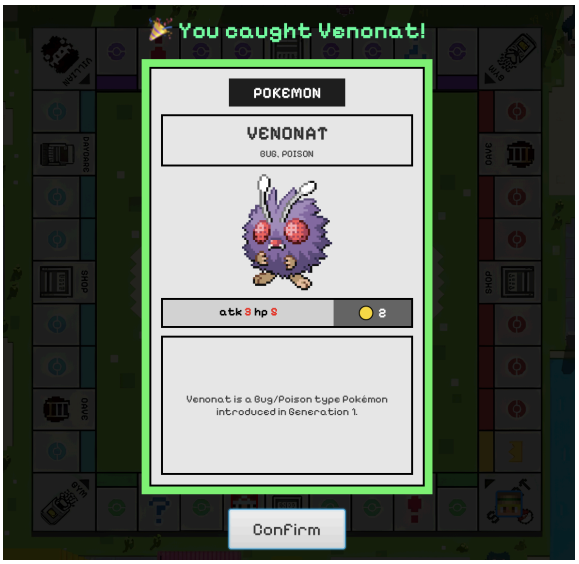
When on a Grass Tile, player can choose to catch or ignore pokemon.



If the player choose to catch a pokemon, there are 2 steps:

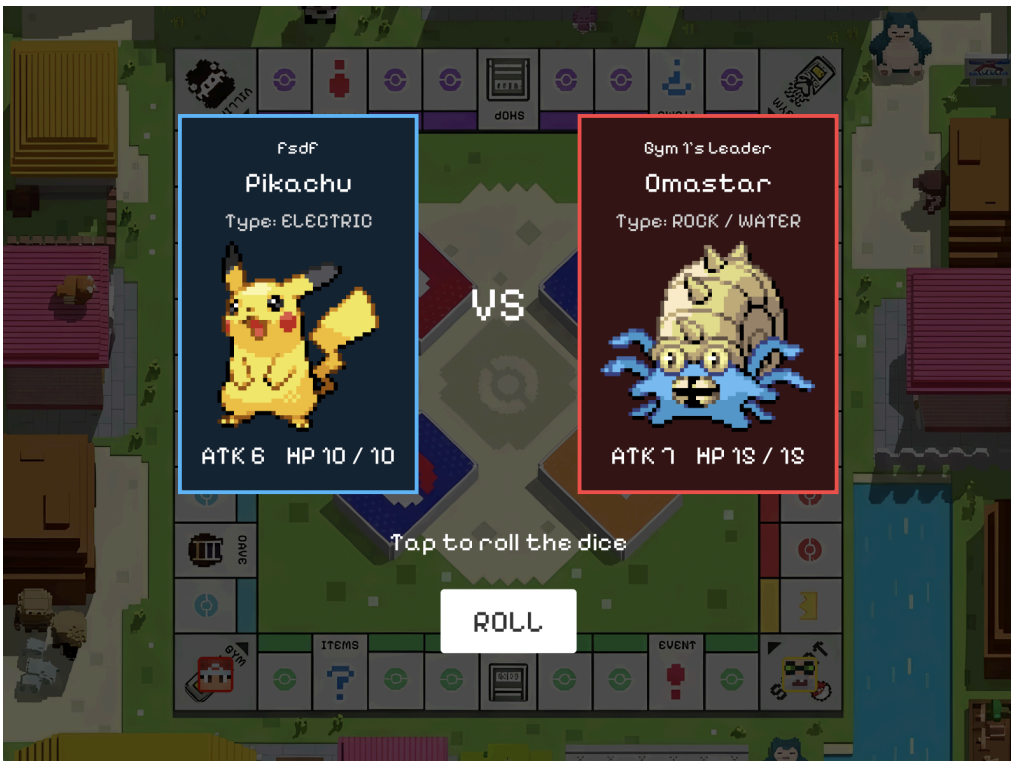
1. Select a Pokéball:
    - Red Ball: Normal catch condition.
    - Great Ball: Catch difficulty decreases by 1.
    - Hyper Ball: Catch difficulty decreases by 2.

2. Roll the dice: If the roll meets or exceeds the required value, the Pokémon is caught.



2.3 Battle System

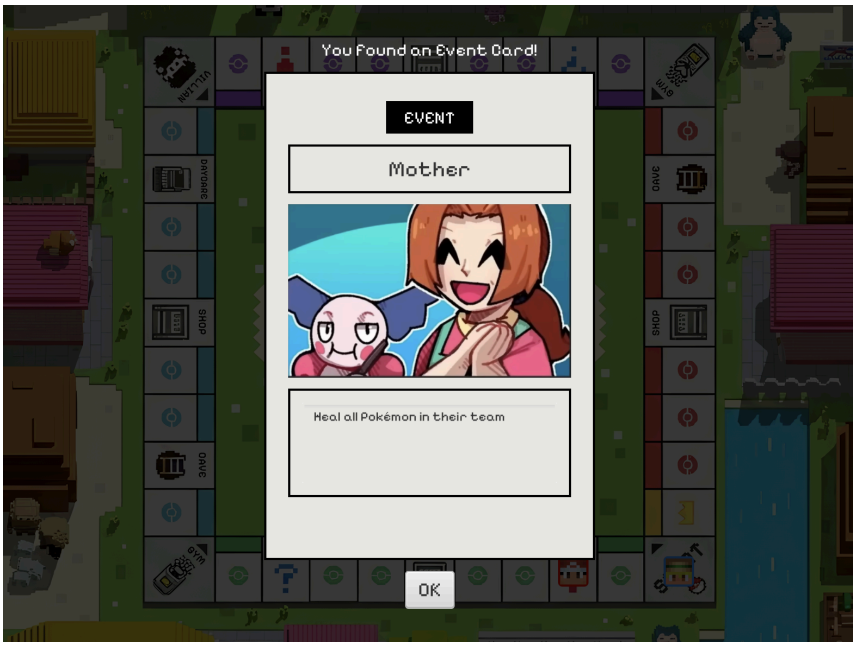
When landing on a Battle Tile, the player may choose to fight the boss:



- Attack: The player rolls the dice; if their roll is higher than the boss's, they attack and win.
- Rewards:
    - Defeating Gym Leader 1 or 2 grants a badge (worth bonus coins). If the player already owns the badge, they receive +5 coins instead.
    - Defeating the Villain grants +10 coins.
- Penalty for losing: The selected Pokémon’s HP becomes 0, rendering it unusable until healed.

3. Cards and Items

3.1 Event Cards



Event cards apply immediate effects such as:

- Movement boosts or penalties
- Gaining coins
- Forced item usage
- Temporary buffs for battles or catching Pokémon

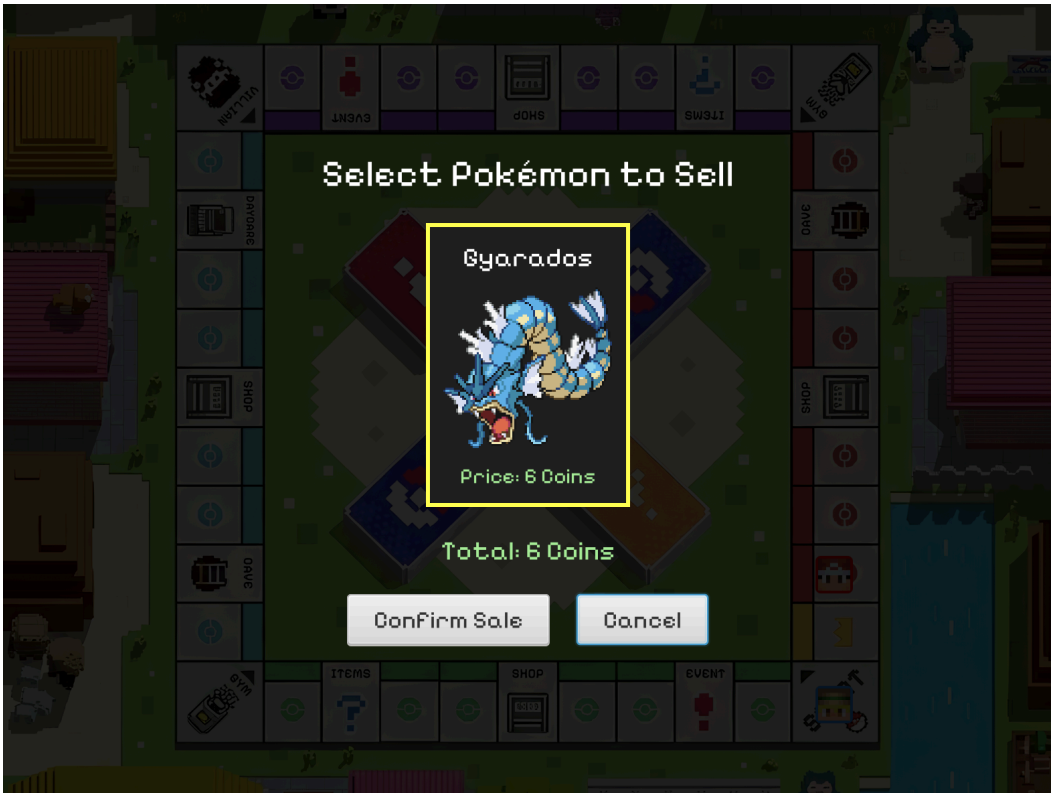
3.2 Item Cards

Items can be used before starting a turn to give players advantages:

- Increased movement
- Improved chances of catching Pokémon
- Healing or reviving Pokémon
- Additional coins

4. Progression and Winning the Game

4.1 Selling Pokémon



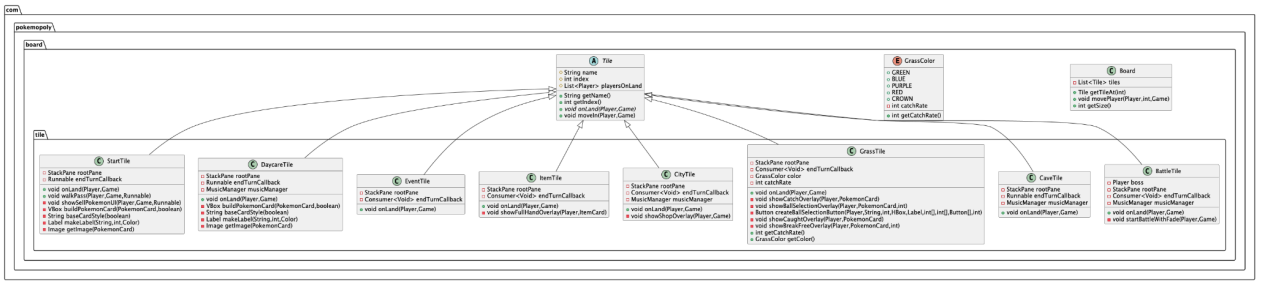
After completing a full lap around the board, players can pass the Start Tile again and sell Pokémon from their team for coins. Selling is a strategic choice to increase the final score.

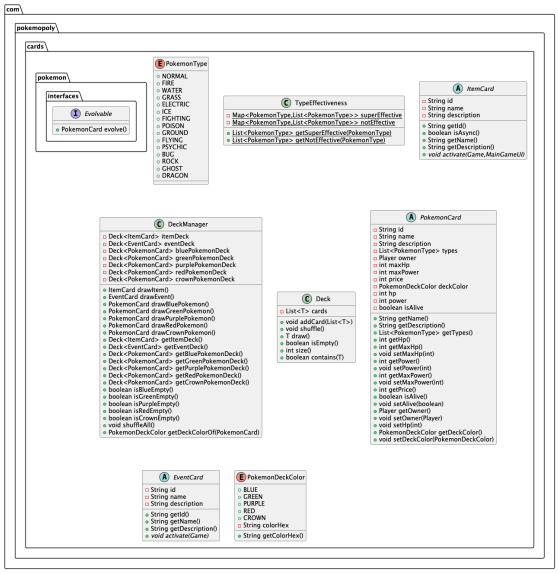
4.2 Winning the Game

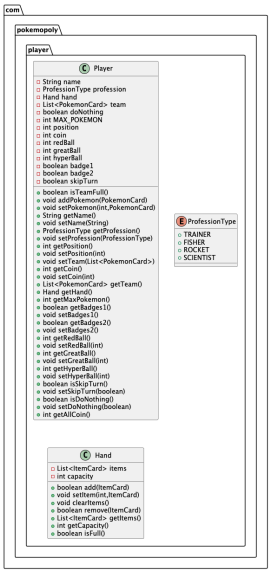
The game ends after a fixed number of turns (50 turns per player). The winner is the player with the highest total wealth, calculated from:

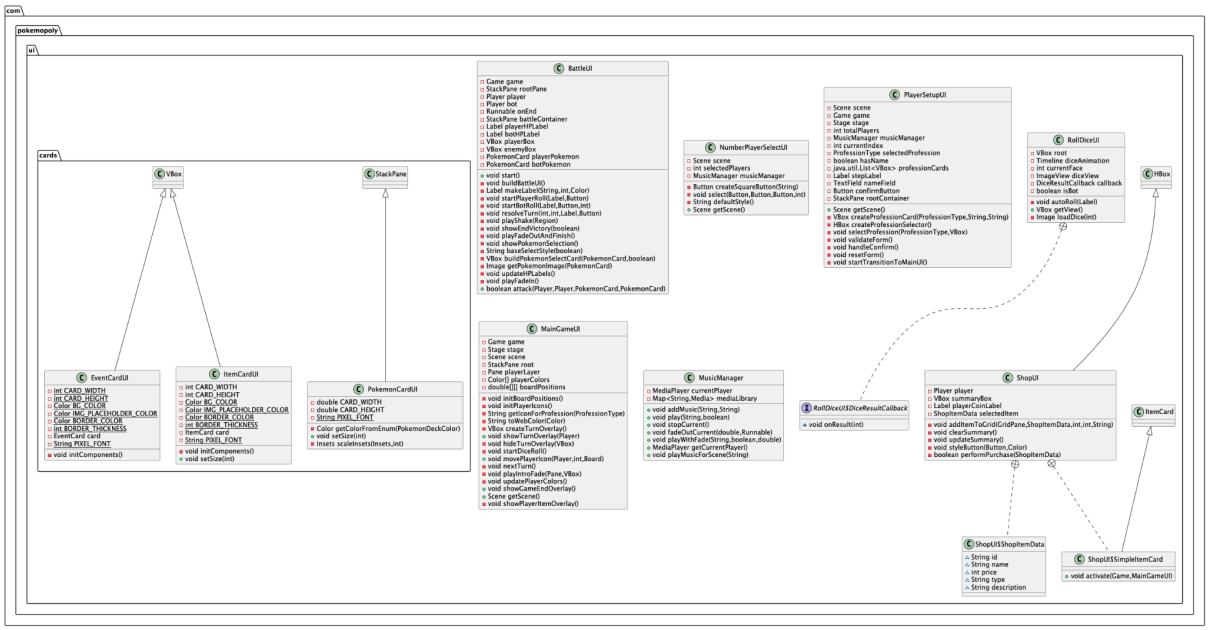
- Coins
- Value of all Pokémon owned
- Badges earned (each badge adds bonus coins)

5. Class diagram









**not include pokemon, item, event class

1. package com.pokemopoly

1.1 class Main extends Application

This class is the main entry point of the program. It initializes the main window, and starts the application.

Methods

Method	Description
+ void start(Stage primaryStage)	Initializes the game ,musicManager and shows the numberPlayerSelectUI window.
+ static void main(String[] args)	Launches the JavaFX application.

1.2 class Game

This class is the main logic of the program.

Methods

Method	Description
+ void start(Stage primaryStage)	Initializes the game ,musicManager and shows the numberPlayerSelectUI window.
+ static void main(String[] args)	Launches the JavaFX application.

2. package com.pokemopoly.board

2.1 class Board

The board is circular, meaning indices wrap around when exceeding the size.

It provides movement logic for players and handles tile events such as passing or landing on the Start tile.

Methods

Method	Description
+ Tile getTileAt(int index)	Returns the tile at the given index. The index wraps around the board size, allowing traversal of the board in a circular manner.
+ void movePlayer(Player player, int steps, Game game)	Moves the specified player forward by a number of steps. This method handles the following: - Wrapping around the board when exceeding its size - Detecting and processing when the player passes the Start tile - Triggering the tile's landing or movement effect

+ int getSize()	Returns the number of tiles on the board

2.2 Enum GrassColor

Represents the different colors of grass tiles on the board.
Each grass color corresponds to a specific catch rate value, which determines
the difficulty of catching Pokemon that appear on tiles of that color.

Methods

Method	Description
+ int getCatchRate()	return the catch rate value

2.3 public abstract Class Tile

Represents a tile on the game board.
Each tile has a name, an index position, and maintains a list of players
currently standing on it. Tiles define specific behaviors that occur when a
player lands on them through the onLand(Player, Game) method.

Methods

Method	Description
+ String getName()	Returns the full display name of the tile in the format: "{index} - {name}"
+ int getIndex()	Returns the index of the tile on the board.
+ *void onLand(Player player, Game game)*	Called when a player lands on this tile. Each tile type must define its own behavior when landed on, such as granting rewards, triggering events, or initiating battles.

2.4 package com.pokemopoly.board.tile

2.4.1 public class BattleTile extends Tile

Represents a battle-related tile on the board, such as Gyms or Villain encounters.
When a player lands on this tile, they are prompted to start a Pokemon battle
against a pre-defined boss trainer. The tile also manages UI overlays, music transitions, and initiates the {@link BattleUI}
battle screen.

Methods

Method	Description
- void startBattleWithFade(Player player, Game game)	Handles the transition into the battle scene. Performs a fade-in animation, ensures the boss has at least one Pokemon, draws Pokemon from the deck if needed, and initializes BattleUI.
+ void onLand(Player player, Game game)	Called when a player lands on this tile. Displays a UI overlay asking the player if they want to start a battle. Validates that the player has at least one alive Pokemon. If the player chooses to fight, a fade transition is triggered and the battle begins. If the player skips, the callback is executed and normal gameplay continues.

2.4.2 public class CaveTile extends Tile

Represents a Cave Trap tile on the board.

When a player lands on this tile, they immediately become trapped and are forced to skip their next turn. A UI overlay is displayed informing the player, and the background music transitions to a cave theme.

Methods

Method	Description
+ void onLand(Player player, Game game)	Triggered when a player lands on this tile. The player becomes trapped and will skip their next turn. The method also displays a modal UI overlay informing the player, and changes the music to the cave theme. When the user acknowledges the message, it triggers the end turn callback and returns the music to normal background theme.

2.4.3 public class CityTile extends Tile

Represents a City tile on the game board.

When a player lands on a CityTile:

All of the player's Pokemon are fully healed.

Background music transitions to the Pokemon Center theme.

A shop UI overlay is displayed, allowing the player to buy various items.

After leaving the shop, the turn ends and the background music returns to the normal overworld theme.

Methods

Method	Description
+ void onLand(Player player, Game game)	Called when a player lands on this tile. This method fully restores all Pokemon in the player's team, logs the event, switches the background music to the Pokemon Center theme, and opens a shop UI.
- void showShopOverlay(Player player, Game game)	Displays a shop overlay allowing the player to purchase items. The UI consists of: A title label The ShopUI component showing shop items An exit button that closes the shop and ends the turn

2.4.4 public class DaycareTile extends Tile

Represents a Daycare tile on the board:

When a player lands on this tile: All Pokemon that implement {@link Evolvable} are displayed. The player may select one Pokemon to evolve.

A confirmation interface appears showing both the old and evolved form. Background music changes to the Daycare theme. If the player has no evolvable Pokemon, the turn ends immediately.

Methods

Method	Description
+ void onLand(Player player, Game game)	Handles the event when a player lands on the Daycare tile. The method: Finds all Pokemon in the player's team that can evolve. If none exist, immediately ends the player's turn. Otherwise, transitions music and displays the evolution UI.
- VBox buildPokemonCard(PokemonCard card, boolean shrink)	Builds a UI card displaying Pokemon details including name, type, sprite, and stats.
- String baseCardStyle(boolean shrink)	Returns the CSS style string for a Pokemon card.
- Label makeLabel(String text, int size, Color color)	Creates a styled label using the Pixelify Sans font.
- Image getImage(PokemonCard card)	Retrieves the sprite image for the given Pokemon.

	If the sprite is missing, a placeholder is used instead

2.4.5 public class EventTile extends Tile

Represents a board tile that triggers an Event Card draw when a player lands on it. When a player lands on the EventTile, a random {@link EventCard} is drawn from the game's com.pokemopoly.cards.DeckManager The tile displays an overlay showing the card, and lets the player confirm to activate it. After activation, the turn ends via the callback.

Methods

Method	Description
+ void onLand(Player player, Game game)	Triggered when a player steps on this tile. The method performs the following steps: Draws an EventCard from the game's deck Displays an overlay showing the card and an OK button Once confirmed, activates the event's effect using EventCard activate(Game) Removes the overlay and ends the turn

2.4.6 public class GrassTile extends Tile

Represents a grass tile on the board where players can encounter and catch Pokemon. The catch rate and Pokemon rarity depend on the GrassColor of the tile.

Methods

Method	Description
+ void onLand(Player player, Game game)	Called when a player lands on this tile. Determines the Pokemon deck that will be drawn from based on the grass color's catch rate. If the player's team is full or the deck is empty, the catch attempt is skipped.
- void showCatchOverlay(Player player, PokemonCard pokemonCard)	Displays UI overlay showing the encountered Pokemon and provides options to catch or ignore it.
- void showBallSelectionOverlay(Player player, PokemonCard pokemonCard, int baseCatchRate)	Shows the selection screen for Pokeball types. Selection affects the required dice roll for successful capture.
- Button createBallSelectionButton(Player player,	Creates a UI button for selecting a specific Pokeball type.

String imagePath, int typeModifier, HBox ballBox, Label catchRateLabel, int[] effectiveCatchRate, int[] selectedBallType, Button[] selectedBtn, int baseCatchRate)	
- void showCaughtOverlay(Player player, PokemonCard pokemonCard)	Displays success UI after the player successfully catches a Pokemon. Adds the Pokemon to the player's team if space is available.
- void showBreakFreeOverlay(Player player, PokemonCard pokemonCard, int baseCatchRate)	Displays UI when a Pokemon breaks free, offering retry or release options.
+ int getCatchRate()	return the base catch rate of this grass tile
+ GrassColor getColor()	return the grass color determining encounter rarity

2.4.7 public class ItemTile extends Tile

Represents a tile on the board that gives the player an {@link ItemCard} when landed on.

If the player's hand is not full, the drawn item is added automatically.

If the hand is full, a UI overlay appears allowing the player to either:

Discard the new item

Swap the new item with an existing item in the hand

This tile also displays card UI and buttons using JavaFX overlays.

Methods

Method	Description
+ void onLand(Player player, Game game)	Called when a player lands on the tile. Draws an ItemCard and displays it with UI. If the player's hand has space, the item is added automatically. If the hand is full, the player must choose whether to discard the new item or swap it with an existing item in the hand.
- void showFullHandOverlay(Player player, ItemCard newItem)	Displays an overlay UI when the player's hand is full. The player can: Discard the new item

	Select an item in their hand to swap it with the new item

2.4.8 public class ItemTile extends Tile

Represents the Start tile on the Pokemopoly board.

The Start tile allows players to sell their Pokemon when they land on it

or when they walk past it (depending on game rules) This tile opens an overlay UI that displays all Pokemon in the player's team and allows them to select Pokemon to sell in exchange for coins. If the player has no Pokemon, the tile immediately ends the turn or triggers the provided callback

Methods

Method	Description
+ void onLand(Player player, Game game)	Triggered when a player lands on the Start tile. Opens the Pokemon selling UI.
+ void walkPass(Player player, Game game, Runnable callback)	Called when the player walks past the Start tile without landing on it. Behaves the same as landing, but uses a custom callback.
- void showSellPokemonUI(Player player, Game game, Runnable callback)	Displays an overlay allowing the player to select Pokemon from their team to sell for coins. Supports selection, calculating total value, confirmation,and cancellation. If the player has no Pokemon, the method immediately calls the appropriate callback.
- VBox buildPokemonCard(PokemonCard card, boolean shrink)	Builds a UI card representing a Pokemon that can be clicked for selection. Includes name, sprite image, and price.
- String baseCardStyle(boolean shrink)	Generates the base CSS style string for a Pokemon card.
- Label makeLabel(String text, int size, Color color)	Creates a styled label using the Pixelify Sans font.
- Image getImage(PokemonCard card)	Attempts to load the image sprite for the given Pokemon. * If the specific sprite does not exist, falls back to a

	placeholder image.

3. package com.pokemopoly.player

3.1 class Player

Represents a player participating in the game

Represents a player participating in the game.

Method	Description
+ String getName()	Returns the player’s display name.
+ void setName(String name)	Sets the player’s name.
+ ProfessionType getProfession()	Returns the player’s selected profession.
+ void setProfession(ProfessionType profession)	Changes the player’s profession.
+ int getPosition()	Returns the player’s current tile index on the board.
+ void setPosition(int position)	Updates and wraps the player’s position on the board (0–39).
+ List<PokemonCard> getTeam()	Returns the player’s active Pokémon team.
+ void addPokemon(PokemonCard pokemon)	Adds a Pokémon to the team if there is space available.

+ boolean isTeamFull()	Returns whether the team has reached the maximum of 6 Pokémon.
+ Hand getHand()	Returns the player’s item hand with capacity 4.
+ int getCoin()	Returns the number of coins the player currently has.
+ void setCoin(int coin)	Sets the player’s coin amount.
+ boolean isSkipTurn()	Returns whether the player must skip their next turn.
+ void setSkipTurn(boolean skipTurn)	Flags the player to skip or play their next turn.
+ boolean isDoNothing()	Returns whether the player has chosen to take no action this turn.
+ void setDoNothing(boolean doNothing)	Sets the “do nothing” state for the turn.
+ int getAllCoin()	Returns the player’s total wealth, calculated as: coins + value of all Pokémon + badges.

3.2 class Hand

Represents a player's hand of item cards, with a fixed capacity.

Method	Description
+ Hand(int capacity)	Initializes the hand with the specified capacity.

+ boolean add(ItemCard card)	Attempts to add an ItemCard to the hand. Returns true if successful, false if hand is full.
+ void setItem(int idx, ItemCard card)	Replaces the ItemCard at the given index.
+ void clearItems()	Removes all ItemCards from the hand.
+ boolean remove(ItemCard card)	Removes a specific ItemCard from the hand and returns true if it was present.
+ List<ItemCard> getItems()	Returns the current list of ItemCards in the hand.
+ int getCapacity()	Returns the maximum number of items the hand can hold.
+ boolean isFull()	Returns whether the hand has reached its maximum capacity.

3.3 enum ProfessionType

Represents the available professions for a player to choose at the start of the game.

Enum Constant	Description
TRAINER	The Trainer profession
FISHER	The Fisher profession
ROCKET	The Rocket profession
SCIENTIST	The Scientist profession

4. package com.pokemopoly.ui

4.1 class BattleUI

Represents the user interface and logic for a Pokémon battle.

Method	Description

+ BattleUI(Game game, StackPane rootPane, Player player, Player bot, Runnable onEnd)	Constructor: Initializes the battle UI, linking it to the main Game, the root JavaFX pane, the competing Player and bot, and a callback to run when the battle ends.
+ void start()	Initiates the battle sequence. Checks if both teams have Pokémon, sets the bot's Pokémon, and displays the Pokémon selection screen for the player.
+ boolean attack(Player attacker, Player defender, PokemonCard atkPoke, PokemonCard defPoke)	Executes an attack. Calculates damage based on the attacker's power and type effectiveness against the defender's Pokémon. Reduces the defender's HP and returns true if the defender's Pokémon faints.

4.2 class MainGameUI

The main user interface for the running Pokemopoly game, handling the game board display, player movements, and turn-based interactions.

Method	Description
+ MainGameUI(Game game, Stage stage)	Constructor: Initializes the main game UI, including loading the board, setting up player icons, and initiating the game's introduction fade sequence.
+ void showTurnOverlay(Player p)	Displays the turn-specific overlay for the given player, handling logic for skipping a turn and enabling/disabling the 'Use item' button based on the player's hand.
+ void movePlayerIcon(Player currentPlayer, int n, Board board)	Animates the movement of the current player's icon across the board by 'n' steps and triggers the underlying board's movement logic once the animation completes.
+ void showGameEndOverlay()	Displays a final overlay screen after the game ends, showing the player rankings based on total wealth, and providing options to exit the game or

	start a new one.
+ Scene getScene()	Returns the JavaFX Scene object that contains the entire main game interface.

4.3 class MusicManager

Method	Description
+ MusicManager()	Constructor: Initializes the music manager and loads all predefined sound files (title, palletTown, pokemoncenter, battle, opening, cave, daycare) into the media library.
+ void addMusic(String key, String path)	Loads a media file from the given path and stores it in the library with the specified key.
+ void play(String key, boolean loop)	Stops the current player (if any) and starts playing the music associated with the given key. The music is set to loop indefinitely if true.
+ void stopCurrent()	Stops the currently playing music and releases the media player.
+ void fadeOutCurrent(double seconds, Runnable afterFade)	Fades out the volume of the current track over the specified number of seconds. The Runnable is executed after the fade-out is complete and the music is stopped.
+ void playWithFade(String key, boolean loop, double fadeInSeconds)	Starts playing the music associated with the given key from a volume of 0 and fades it in to full volume over the specified number of seconds. The music is set to loop indefinitely if true .
+ MediaPlayer getCurrentPlayer()	Returns the currently active MediaPlayer instance.
`` + void playMusicForScene(String sceneKey) ```	Plays the music corresponding to a specific scene key (e.g., "title", "battle", "cave"). If the key is not recognized, it

	stops the current music.

4.4 class NumberPlayerSelectUI

Handles the UI for selecting the number of players (2, 3, or 4) at the start of the game.

Method	Description
+ NumberPlayerSelectUI(Game game, Stage stage, MusicManager musicManager)	Constructor: Initializes the UI, sets up player count buttons, and defines the logic for the "Confirm" button to proceed to the next setup stage (after fading out the current music).
- Button createSquareButton(String text)	Creates a stylized square JavaFX Button with the given text.
- void select(Button b2, Button b3, Button b4, int num)	Handles the click event for the player count buttons, setting theselectedPlayers variable, updating the button styles to highlight the selection, and enabling the "Confirm" button.
- String defaultStyle()	Returns the default CSS style string for the player count buttons.
+ Scene getScene()	Returns the JavaFX Scene object for this UI.

4.5 class PlayerSetupUI

Handles the UI for player name entry and profession selection at the start of the game.

Method	Description
+ PlayerSetupUI(Game game, Stage stage, int totalPlayers, MusicManager musicManager)	Constructor: Initializes the UI for setting up each player's name and profession, managing the total number of players and music manager.
+ Scene getScene()	Returns the JavaFX Scene object for this UI.

- VBox createProfessionCard(ProfessionType type, String imgPath, String abilityText)	Creates a clickable UI card for selecting a profession, including the profession type, image, and starting Pokémon.
- HBox createProfessionSelector()	Creates the horizontal container (HBox) holding all four profession selection cards.
- void selectProfession(ProfessionType type, VBox selectedCard)	Updates the selected profession when a card is clicked, visually highlighting the chosen card and enabling the confirm button if a name has been entered.
- void validateForm()	Checks if both a name has been entered and a profession has been selected, enabling or disabling the confirm button accordingly.
- void handleConfirm()	Called when the confirm button is pressed. Adds the new Player to the Game. If more players need setup, it calls resetForm(); otherwise, it calls startTransitionToMainUI().
- void resetForm()	Clears the input fields and resets the UI for the next player's setup.
- void startTransitionToMainUI()	Initiates a sequence of animations (fade-in, zoom, fade-out) to transition from the setup screen to the main game UI (MainGameUI), accompanied by a music fade-out and new music playback.

4.6 class RollDiceUI

Represents the user interface component for rolling a six-sided die, including animation and result callback.

Method	Description
+ RollDiceUI(DiceResultCallback callback)	Constructor: Initializes the UI for a human player.

+ RollDiceUI(DiceResultCallback callback, boolean isBot)	Constructor: Initializes the UI, specifying whether it's for a human player (isBot = false) or an automated bot (isBot = true).
+ VBox getView()	Returns the main JavaFX container (VBox) for the dice UI.
- void autoRoll(Label resultLabel)	Handles the dice roll logic and animation sequence when the roller is a bot.
- Image loadDice(int face)	Loads the corresponding image file for the given dice face number.
public interface DiceResultCallback void onResult(int diceValue);	Interface defining a callback method that is executed once the dice roll animation is complete and the final result (diceValue) is available.

4.7 class ShopUI

Represents the shop user interface where players can buy items and Pokéballs.

Method	Description
+ ShopUI(Player player)	Constructor: Initializes the shop UI and displays available items for the given player.
- void addItemToGrid(GridPane grid, ShopItemData itemData, int col, int row, String imagePath)	Creates the interactive UI card for a shop item, adds it to the grid, and sets up selection/hover effects.
- void clearSummary()	Clears the item summary/preview box and resets the selected item.
- void updateSummary()	Displays the details, cost, and purchase controls for the currently selected item in the summary box.
- void styleButton(Button btn, Color color)	Applies custom CSS styling (color, font, cursor) to a button.

- boolean performPurchase(ShopItemData item)	Executes the logic for purchasing an item or Pokéball, deducting coins and adding the item/ball to the player's inventory. Returns true if the purchase succeeds.

4.8 package com.pokemopoly.ui.cards

4.8.1 class EventCardUI

Represents the visual display (a JavaFX VBox) for an Event Card.

Method	Description
+ EventCardUI(EventCard card)	Constructor: Initializes the UI component for an Event Card, setting its layout properties and building its visual elements from the card data.

4.8.2 class ItemCardUI

Represents the visual display (a JavaFX VBox) for an Item Card.

Method	Description
+ ItemCardUI(ItemCard card)	Constructor: Initializes the UI component for an Item Card, setting its layout properties and building its visual elements from the card data.
+ void setSize(int ratio)	Resizes the card UI by scaling its dimensions, padding, spacing, and font sizes by the given ratio. Note: This method is marked as private/package-private in the original document but appears public in the provided code snippet, so it is included.

4.8.3 class PokemonCardUI

Represents the visual display (a JavaFX StackPane) for a Pokémon Card, including its stats and description.

Method	Description
+ PokemonCardUI(PokemonCard card)	Constructor: Initializes the UI component for a Pokémon Card, setting its layout and displaying its stats, type, and description.

- Color getColorFromEnum(PokemonDeckColor colorEnum)	Returns the corresponding JavaFX Color object for a given PokemonDeckColor enum value.
+ void setSize(int ratio)	Resizes the card UI by scaling its dimensions, padding, spacing, and font sizes by the given ratio.
- Insets scaleInsets(Insets in, int ratio)	Scales JavaFX Insets values by the given ratio.

5. package com.pokemopoly.cards

5.1 class Deck<T>

Represents a generic deck of cards with basic deck operations like shuffling and drawing.

Method	Description
+ Deck()	Constructor: Initializes an empty deck of cards.
+ void addCard(List<T> cardsList)	Adds a list of cards to the deck.
+ void shuffle()	Randomly shuffles the cards currently in the deck.
+ T draw()	Removes and returns the first card from the deck. Prints a message and returns null if the deck is empty.
+ boolean isEmpty()	Returns true if the deck contains no cards.
+ int size()	Returns the number of cards currently in the deck.
+ boolean contains(T card)	Returns true if the specified card is present in the deck.

5.2 class DeckManager

Manages all card decks (Item, Event, and colored Pokémon decks) for the game.

Method	Description

+ DeckManager(...)	Constructor: Initializes the deck manager with all necessary card decks.
+ ItemCard drawItem()	Draws and returns the top card from the item deck.
+ EventCard drawEvent()	Draws and returns the top card from the event deck.
+ PokemonCard drawBluePokemon()	Draws and returns the top card from the blue Pokémon deck.
+ PokemonCard drawGreenPokemon()	Draws and returns the top card from the green Pokémon deck.
+ PokemonCard drawPurplePokemon()	Draws and returns the top card from the purple Pokémon deck.
+ PokemonCard drawRedPokemon()	Draws and returns the top card from the red Pokémon deck.
+ PokemonCard drawCrownPokemon()	Draws and returns the top card from the crown Pokémon deck.
+ Deck<ItemCard> getItemDeck()	Returns the item deck object.
+ Deck<EventCard> getEventDeck()	Returns the event deck object.
+ Deck<PokemonCard> getBluePokemonDeck()	Returns the blue Pokémon deck object.
+ Deck<PokemonCard> getGreenPokemonDeck()	Returns the green Pokémon deck object.
+ Deck<PokemonCard> getPurplePokemonDeck()	Returns the purple Pokémon deck object.
+ Deck<PokemonCard> getRedPokemonDeck()	Returns the red Pokémon deck object.
+ Deck<PokemonCard> getCrownPokemonDeck()	Returns the crown Pokémon deck object.
+ boolean isBlueEmpty()	Returns whether the blue Pokémon deck is empty.
+ boolean isGreenEmpty()	Returns whether the green Pokémon

	deck is empty.
+ boolean isPurpleEmpty()	Returns whether the purple Pokémon deck is empty.
+ boolean isRedEmpty()	Returns whether the red Pokémon deck is empty.
+ boolean isCrownEmpty()	Returns whether the crown Pokémon deck is empty.
+ void shuffleAll()	Shuffles all card decks (item, event, and all colored Pokémon decks).
+ PokemonDeckColor getDeckColorOf(PokemonCard card)	Returns the color of the deck the given Pokémon card belongs to, or null if the card is not found in any of the decks.

5.3 abstract class EventCard

Represents an abstract Event Card with common properties and the method to activate its effect in the game.

Method	Description
+ EventCard(String id, String name, String description)	Constructor: Initializes the card with its unique ID, name, and description.
+ String getId()	Returns the unique ID of the event card.
+ String getName()	Returns the name of the event card.
+ String getDescription()	Returns the description of the event card.
+ abstract void activate(Game game)	Abstract method: Defines the card's effect to be executed when the event is activated during the game.

5.4 abstract class ItemCard

Represents an abstract Item Card with common properties and the method to activate its effect in the game.

Method	Description
+ ItemCard(String id, String name, String description)	Constructor: Initializes the item card with its unique ID, name, and

	description.
+ String getId()	Returns the unique ID of the item card.
+ boolean isAsync()	Returns true if the item's activation involves asynchronous operations, otherwise false. The default is false.
+ String getName()	Returns the name of the item card.
+ String getDescription()	Returns the description of the item card.
+ abstract void activate(Game game, MainGameUI gameUI)	Abstract method: Defines the card's effect to be executed when the item is used during the game.

5.5 abstract class PokemonCard

Represents an abstract Pokémon Card with combat and ownership properties.

Method	Description
+ PokemonCard(...)	Constructor: Initializes the Pokémon with its stats, price, types, and base data.
+ String getName()	Returns the name of the Pokémon.
+ String getDescription()	Returns the description of the Pokémon.
+ List<PokemonType> getTypes()	Returns the list of types the Pokémon has.
+ int getHp()	Returns the Pokémon's current HP.
+ int getMaxHp()	Returns the Pokémon's maximum HP.
+ void setMaxHp(int maxHp)	Sets the Pokémon's maximum HP.
+ int getPower()	Returns the Pokémon's current attack power.
+ void setPower(int power)	Sets the Pokémon's attack power.
+ int getMaxPower()	Returns the Pokémon's maximum attack power.
+ void setMaxPower(int maxPower)	Sets the Pokémon's maximum attack

	power.
+ int getPrice()	Returns the coin price/sell value of the Pokémon.
+ boolean isAlive()	Returns true if the Pokémon's HP is greater than 0.
+ void setAlive(boolean alive)	Manually sets the alive status.
+ Player getOwner()	Returns the Player who owns this Pokémon.
+ void setOwner(Player owner)	Sets the Player who owns this Pokémon.
+ void setHp(int hp)	Sets the Pokémon's current HP, ensuring it is not less than 0, and setting isAlive to false if HP becomes 0.
+ PokemonDeckColor getDeckColor()	Returns the color of the deck this Pokémon belongs to.
+ void setDeckColor(PokemonDeckColor deckColor)	Sets the color of the deck this Pokémon belongs to.

5.6 class TypeEffectiveness

Provides static utility methods for determining Pokémon type effectiveness (super effective and not effective match-ups).

Method	Description
+ static List<PokemonType> getSuperEffective(PokemonType type)	Returns a list of PokemonType that the given attack type is super effective against.
+ static List<PokemonType> getNotEffective(PokemonType type)	Returns a list of PokemonType that the given attack type is not very effective against (or which resist the attack type's damage).

5.7 enum PokemonDeckColor

Represents the different colors used to categorize Pokémon decks, each with a corresponding hexadecimal color value.

Element	Description (Color Hex)
BLUE	#3DA9FC
GREEN	#16C172
PURPLE	#A66DD4
RED	#FF595E
CROWN	#F4CA16

Method	Description
+ String getColorHex()	Returns the hexadecimal color string associated with the deck color.

5.8 enum PokemonType

Represents the various types a Pokémon can have.

Enum Constant	Description
NORMAL	Normal type
FIRE	Fire type
WATER	Water type
GRASS	Grass type
ELECTRIC	Electric type
ICE	Ice type
FIGHTING	Fighting type
POISON	Poison type
GROUND	Ground type
FLYING	Flying type
PSYCHIC	Psychic type
BUG	Bug type
ROCK	Rock type

GHOST	Ghost type
DRAGON	Dragon type

6. package com.pokemopoly.cards.pokemon.interfaces

6.1 interface Evolvable

Defines the contract for Pokémon cards that are capable of evolution.

Method	Description
+ PokemonCard evolve()	Returns a new PokemonCard instance representing the evolved form.