Travis Moret
Final Project Design + Reflection

Design

Main

- Includes its own menu with a do while loop until the user chooses to exit

Pokemon

- Variables
    - name
    - armor
    - strengthPoints
    - attack
    - defense
- Functions
    - Virtual constructor
    - Virtual destructor
    - Virtual attack move - abstract so it doesn't need to be made in class
    - Virtual defense move – abstract so it doesn't need to be made in class
    - Getters and setters

Note: set up die values and child inherited variables in child constructors.

Pikachu

Child class

- Has inherited variables
- Overwritten functions for constructor, destructor, attack and defense move
    - Will have to implement a high chance of stunning the attacker so they can't attack

Lapras

Child class

- Has inherited variables
- Overwritten functions for constructor, destructor, attack and defense move
    - High hit points
    - Chance to freeze opponent for a turn so they can't attack

Machamp

Child class

- Has inherited variables
- Overwritten functions for constructor, destructor, attack and defense move
    - High attack but low defense and no special abilities

Travis Moret
Final Project Design + Reflection

Gengar

Child class

- Has inherited variables
- Overwritten functions for constructor, destructor, attack and defense move
    - High attack and low defense
    - Random chance for user to miss because of ghost type

Dragonite

Child class

- Has inherited variables
- Overwritten functions for constructor, destructor, attack and defense move
    - Very high defense and attack
    - Random chance to dodge attack because of speed

Note: There won't be much in the child classes, as they inherit things. The classes just have properties which we will use extensively in the Menu Class.

Die

Standalone class

- sets the characteristics of a die
- variables for sides, rollvalue and number of dice
- function to roll a random value up to the number of sides of the die

Game

Primary driver functions for the program

Variables

- A pointer for each space in the board

Functions

- Function to free pointers on board to free memory
- Houses input validation
- Function for instructions to user
- Functions for attack and defense sequences
- Function to print inventory

Space

Variables

Travis Moret
Final Project Design + Reflection

- Vector container with a limit of 1 for item inveontory containment
    - To hold a gold key string
    - User will receive gold key if they defeat all 4 of the elite 4
    - Needed to enter Champion Chamber

Functions

- Getters and setters
- Virtual functions for menu
- Virtual functions for dialogue
    - Dialogue and menus will be different for each gym leader

Indigo Plateau

Child class of Space

- Show location of current and surrounding spaces
- Print menu for this space
- Print dialogue for this space

Lorelei Gym

Child class of Space

- Show location of current and surrounding spaces
- Print menu for this space
- Print dialogue for this space
- Has a Lapras object

Bruno Gym

Child class of Space

- Show location of current and surrounding spaces
- Print menu for this space
- Print dialogue for this space
- Has a Machamp object

Agatha Gym

Child class of Space

- Show location of current and surrounding spaces
- Print menu for this space
- Print dialogue for this space
- Has a Gengar Object

Travis Moret
Final Project Design + Reflection

Lance Gym

Child class of Space

- Show location of current and surrounding spaces
- Print menu for this space
- Print dialogue for this space
- Has a Dragonite object

Champion Chamber

Child class of Space

- Print dialogue for this space

Test Scope	Test	Expected Results	Actual Results
Input Validation	1. User enters zeroes before numbers ex: 040 2. User enters negative numbers ex: -40 3. Characters or letters passed in ex: cool 4. Float or double input entered ex: 40.0 5. User enters numbers followed by letters ex: 4cd 6. User presses ENTER with no input 7. User enters a number too high ex: 200 for character 8. User enters a number too low ex: 0 for character	• Inputs do not crash the program • User is given an error report that asks for the proper input type and range of numbers again • Input validation does not create an endless loop	• Inputs do not crash the program • User is given an error report that asks for the proper input type and range of numbers again • Input validation does not create an endless loop

Travis Moret
Final Project Design + Reflection

Battle Simulation	1. Defender strengthPoints reaches 0 2. Damage against a player has a negative value 3. Players leave scope 4. Dice and arrays leave scope	1. Pokemon dies and Pikachu wins 2. Damage is set to 0 3. Players are destroyed to free memory 4. Dice and arrays are destroyed to free memory ○ No segmentation faults ○ Program does not crash ○ No memory leaks	1. Pokemon faints and Pikachu wins 2. Damage is set to 0 3. Players are destroyed to free memory 4. Dice and arrays are destroyed to free memory • No segmentation faults • Program does not crash • Some memory leaks
Play again	1. User wants to quit 2. User plays again 3. User enters invalid value	1. Game quits 2. Instructions shown again and game restarts 3. User is re-prompted for input and input is validated again. ○ No memory leaks ○ Program does not crash ○ No segfaults	1. Game quits 2. Instructions shown again and game restarts 3. User is re-prompted for input and input is validated again. ○ No memory leaks ○ Program does not crash ○ No segfaults
Pokemon	1. Pikachu wins round 2. Pikachu loses 17 hitpoints 3. Pikachu loses 30 HP	1. Pikachu can enter next space 2. Status function will display 13/30 HP 3. Game restarts	1. Pikachu can enter next space 2. Status function will display 13/30 HP 3. Game restarts
Makefile Compilation	1. Creator enters make to	1. Makefile created with no errors	1. Makefile created with no errors

	create makefile		
Memory Leaks	1. Ran program and quit on first menu. 2. Ran program and completed program with large number of rounds.	• For each case, all memory should have been freed and no memory leaks should be possible.	• No memory leaks or errors
Space class	1. Child class made 2. Virtual menu and dialogue	1. Inherits values from Space class 2. Functions overwritten in each child class	1. Inherits values from Space class 2. Functions overwritten in each child class
Game Class	1. User chooses to move to next space 2. User is on Indigo Plateau 3. Pikachu faints 4. Pikachu defeats Lance 5. User reaches Champion Chamber	1. Next space is found and moved to 2. Current space is Indigo Plateau and next space is Lorelei Gym 3. Game restarts 4. User receives Golden Key 5. User Wins	1. Next space is found and moved to 2. Current space is Indigo Plateau and next space is Lorelei Gym 3. Game restarts 4. User receives Golden Key 5. User Wins
Inventory	1. User wants to see inventory 2. User has no items in inventory 3. User has an item in inventory 4. User has golden key in inventory	1. Inventory is printed 2. “No items in inventory” printed 3. Inventory item string is printed 4. Golden Key is shown in inventory and user asked if	1. Inventory is printed 2. “No items in inventory” printed 3. Inventory item string is printed 4. Golden Key is shown in inventory and user asked if

Travis Moret
Final Project Design + Reflection

		they want to use it	they want to use it

Class Hierarchy

Standalone classes

- Die
- Game

Other files

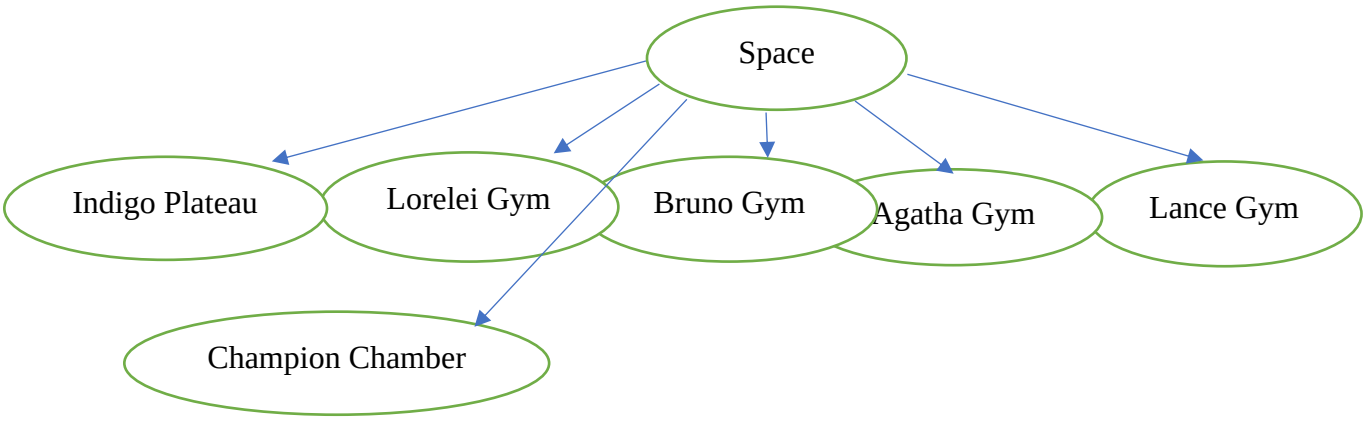
- Main

Base (Parent) Class

- Space

Child classes (subclasses of Character class)

- Indigo Plateau
- Lorelei Gym
- Bruno Gym
- Agatha Gym
- Lance Gym
- Champion Chamber



Reflection

Day 1 – I had been thinking abut what I had wanted to do for my final for some time, so before I started making this design, I had a large part of the idea in mind. My idea, however, did not include a gameboard, so I had to change it from picking a starter Pokémon and battling your rival with it to moving through spaces in a train structure to defeat the Elite Four. I spent some time

stressing about how to make a gameboard, and then decided to start with the parts I knew. I gathered all of my files in a directory, and renamed them appropriately. I repurposed copies of the Vampire from project 4 and made slight differences in stats to have Pokémon for the user and each gym leader. I also made the Character class from project 4 the Pokémon supergroup for each Pokémon type to inherit from. I did not run into many errors with this as this code was already error free and just being repurposed.

Day 2 – I focused on the development of the Space class and its derived classes. I realized I needed variables for the name of the space, the name of the gym leader and also a Pokémon to defend each space. I started adding driver functions to space class, and began to feel overwhelmed. I decided to make a Game Class that would house a lot of setup, teardown and driver functions. This simplified my code. I created the dialogue and menu for each space, and created a vector if strings with a limit of 1 space reserved for the item gained at the end of the game.

Day 3 – I added Pokémon to each space, and began testing. I realized I had to add a way to exit the program if the user's Pikachu ran out of HP. I implemented a function that would show Pikachu's health after the battle in comparison to it's starting health, to show its status after each Gym battle. My step limit is implemented just so: Pikachu must win each battle with the HP that it has, and then will heal before the next space, in which it must do the same. If it defeats each Gym Leader with it's limited HP, it can move on. I then implemented adding the item to the user's inventory at the end of the last gym battle (a key) to open the last space. The vector will show no items until the key is given to the user, and the user must use the key to enter the last space, or they can't move on.

Day 4 – I implemented a status report of what space the user s currently on, and the previous and next space. User can interact with the space by battling and entering next space once the user has won the battle. If Pikachu faints, the game starts over. I tracked down memory leaks and solved them. I finished commenting code, and tested that the user can get to each space, can lose to each Pokémon, and can't open the final space without the key. I am very proud of my final project!