```python
#Classes Challenge 38:  Pykemon Simulation App
import random

#Parent class
class Pykemon():
    """A model of a generic Pykemon character."""

    def __init__(self, name, element, health, speed):
        """Initialize attributes."""
        self.name = name.title()
        self.element = element
        #Current health is current, max health will be referenced for healing
        self.current_health = health
        self.max_health = health
        self.speed = speed
        self.is_alive = True


    def light_attack(self, enemy):
        """A light attack guaranteed to do minimal damage."""
        #Generate damage
        damage = random.randint(15, 25)

        #All pykemon will have a list moves = [light, heavy, restore, special]
        #All light attacks will appear at index 0 in the list moves
        #This attribute will be initialized in the child class
        print("Pykemon " + self.name + " used " + self.moves[0] + ".")
        print("It dealt " + str(damage) + " damage.")

        #Deal damage to the enemy
        enemy.current_health -= damage


    def heavy_attack(self, enemy):
        """A heavy attack that could deal MASSIVE damage, or no damage at all."""
        #Generate damage
        damage = random.randint(0, 50)

        #All pykemon will have a list moves = [light, heavy, restore, special]
        #All heavy attacks will appear at index 1 in the list moves
        #This attribute will be initialized in the child class
        print("Pykemon " + self.name + " used " + self.moves[1] + ".")

        #Dealt no damage
        if damage < 10:
            print("The attack missed!!!")
        else:
            print("It dealt " + str(damage) + " damage.")
            #Deal the damage to the enemy
            enemy.current_health -= damage


    def restore(self):
        """A healing move that will restore our current health."""
        #Generate restore value
        heal = random.randint(15, 25)

        #All pykemon will have a list moves = [light, heavy, restore, special]
        #All restore moves will appear at index 2 in the list moves
        #This attribute will be initialized in the child class
        print("Pykemon " + self.name + " used " + self.moves[2] + ".")
        print("It healed " + str(heal) + " health points.")

        #Heal the pykemon
        self.current_health += heal

        #Check to see if we have exceeded the max health of the pykemon
        if self.current_health > self.max_health:
            self.current_health = self.max_health


    def faint(self):
        """If you run out of health, you faint..."""
        if self.current_health <= 0:
            self.is_alive = False
            print("Pykemon " + self.name + " has fainted!")
            input("Press Enter to continue.")


    def show_stats(self):
        """Display the current pykemon stats."""
        print("\nName: " + self.name)
        print("Element Type: " + self.element)
        print("Health: " + str(self.current_health) + " / " + str(self.max_health))
        print("Speed: " + str(self.speed))

#Child Classes
class Fire(Pykemon):
    """A Fire based pykemon that is a child of the Pykemon parent class."""

    def __init__(self, name, element, health, speed):
        """Initialize attributes from the parent Pykemon class."""
        super().__init__(name, element, health, speed)
        #Move list unique to all Fire type Pykemon
        self.moves = ['Scratch', 'Ember', 'Light', 'Fire Blast']


    def special_attack(self, enemy):
        """FIRE BLAST: an elemental fire move.  Massive damage to grass type,
            normal damage to fire type, minimal damage to water type."""
        print("Pykemon " + self.name + " used " + self.moves[3].upper() + "!")

        #Generate damage based on enemy type
        if enemy.element == 'GRASS':
            print("It's SUPER effective!")
            damage = random.randint(35, 50)
        elif enemy.element == 'WATER':
            print("It's not very effective...")
            damage = random.randint(5, 10)
        else:
            random.randint(10, 30)

        #Deal damage
        print("It dealt " + str(damage) + " damage.")
        enemy.current_health -= damage


    def move_info(self):
        """Display pykemon move info"""
        print("\n" + self.name + " Moves: ")

        #Light attack
        print("-- " + self.moves[0] + " --")
        print("\tAn efficient attack...")
        print("\tGuaranteed to do damage within a range of 15 to 25 damage points.")
        #Heavy attack
        print("-- " + self.moves[1] + " --")
        print("\tAn risky attack...")
        print("\tCould deal damage up to 50 damage points or as little as 0 damage points.")
        #Restore move
        print("-- " + self.moves[2] + " --")
        print("\tA restorative move...")
        print("\tGuaranteed to heal your Pykemon 15 to 25 damage points.")
        #Special attack
        print("-- " + self.moves[3] + " --")
        print("\tA powerful FIRE based attack...")
        print("\tGuaranteed to deal MASSIVE damage to GRASS type Pykemon.")

class Water(Pykemon):
    """A Water based pykemon that is a child of the Pykemon parent class."""

    def __init__(self, name, element, health, speed):
        """Initialize attributes from the parent Pykemon class."""
        super().__init__(name, element, health, speed)
        #Move list unique to all Water type Pykemon
        self.moves = ['Bite', 'Splash', 'Dive', 'Water Cannon']


    def special_attack(self, enemy):
        """WATER CANNON: an elemental water move.  Massive damage to fire type,
            normal damage to water type, minimal damage to grass type."""
        print("Pykemon " + self.name + " used " + self.moves[3].upper() + "!")

        #Generate damage based on enemy type
        if enemy.element == 'FIRE':
            print("It's SUPER effective!")
            damage = random.randint(35, 50)
        elif enemy.element == 'GRASS':
            print("It's not very effective...")
            damage = random.randint(5, 10)
        else:
            random.randint(10, 30)

        #Deal damage
        print("It dealt " + str(damage) + " damage.")
        enemy.current_health -= damage


    def move_info(self):
        """Display pykemon move info"""
        print("\n" + self.name + " Moves: ")

        #Light attack
        print("-- " + self.moves[0] + " --")
        print("\tAn efficient attack...")
        print("\tGuaranteed to do damage within a range of 15 to 25 damage points.")
        #Heavy attack
        print("-- " + self.moves[1] + " --")
        print("\tAn risky attack...")
        print("\tCould deal damage up to 50 damage points or as little as 0 damage points.")
        #Restore move
        print("-- " + self.moves[2] + " --")
        print("\tA restorative move...")
        print("\tGuaranteed to heal your Pykemon 15 to 25 damage points.")
        #Special attack
        print("-- " + self.moves[3] + " --")
        print("\tA powerful WATER based attack...")
        print("\tGuaranteed to deal MASSIVE damage to FIRE type Pykemon.")

class Grass(Pykemon):
    """A Grass based pykemon that is a child of the Pykemon parent class."""

    def __init__(self, name, element, health, speed):
        """Initialize attributes from the parent Pykemon class."""
        super().__init__(name, element, health, speed)
        #Move list unique to all Grass type Pykemon
        self.moves = ['Vine Whip', 'Wrap', 'Grow', 'Leaf Blade']


    def special_attack(self, enemy):
        """LEAF BLADE: an elemental grass move.  Massive damage to water type,
            normal damage to grass type, minimal damage to fire type."""
        print("Pykemon " + self.name + " used " + self.moves[3].upper() + "!")

        #Generate damage based on enemy type
        if enemy.element == 'WATER':
            print("It's SUPER effective!")
            damage = random.randint(35, 50)
        elif enemy.element == 'FIRE':
            print("It's not very effective...")
            damage = random.randint(5, 10)
        else:
            random.randint(10, 30)

        #Deal damage
        print("It dealt " + str(damage) + " damage.")
        enemy.current_health -= damage


    def move_info(self):
        """Display pykemon move info"""
        print("\n" + self.name + " Moves: ")

        #Light attack
        print("-- " + self.moves[0] + " --")
        print("\tAn efficient attack...")
        print("\tGuaranteed to do damage within a range of 15 to 25 damage points.")
        #Heavy attack
        print("-- " + self.moves[1] + " --")
        print("\tAn risky attack...")
        print("\tCould deal damage up to 50 damage points or as little as 0 damage points.")
        #Restore move
        print("-- " + self.moves[2] + " --")
        print("\tA restorative move...")
        print("\tGuaranteed to heal your Pykemon 15 to 25 damage points.")
        #Special attack
        print("-- " + self.moves[3] + " --")
        print("\tA powerful GRASS based attack...")
        print("\tGuaranteed to deal MASSIVE damage to WATER type Pykemon.")


#Game class
class Game():
    """A game object to control the creation and flow of pykemon and simulate battle!"""

    def __init__(self):
        """Initialize attributes"""
        #Upon creating a pykemon, element and name will be chosen randomly
        self.pykemon_elements = ['FIRE', 'WATER', 'GRASS']
        self.pykemon_names = ['Chewdie', 'Spatol', 'Burnmander', 'Pykachu', 'Pyonx', 'Abbacab',
                              'Sweetil', 'Jampot', 'Hownstooth', 'Swagilybo', 'Muttle', 'Zantbat',
                              'Wiggly Poof', 'Rubblesaur']
        #Upon creating a pykemon, no battles are won
        self.battles_won = 0


    def create_pykemon(self):
        """Randomly generate a Pykemon!"""
        #Randomly generate health and speed attributes
        health = random.randint(70, 100)
        speed = random.randint(1, 10)

        #Randomly choose an element and name
        element = self.pykemon_elements[random.randint(0, len(self.pykemon_elements)-1)]
        name = self.pykemon_names[random.randint(0, len(self.pykemon_names)-1)]

        #Create the right elemental pykemon
        if element == 'FIRE':
            pykemon = Fire(name, element, health, speed)
        elif element == 'WATER':
            pykemon = Water(name, element, health, speed)
        else:
            pykemon = Grass(name, element, health, speed)

        return pykemon



    def choose_pykemon(self):
        """A method to simulate choosing a starting Pykemon similar to Pokemon"""
        #A list to hold 3 unique starter pykemon
        starters = []

        #Pick 3 different elemental type pykemon for the starter list
        while len(starters) < 3:
            #Make a starter pykemon
            pykemon = self.create_pykemon()

            #Bool to determine if it is unique and should be added to the starters list
            valid_pykemon = True
            for starter in starters:
                #Check if the name or element is already used by another starter
                if starter.name == pykemon.name or starter.element == pykemon.element:
                    valid_pykemon = False
            #The created pykemon is unique, add it to the list starter
            if valid_pykemon:
                starters.append(pykemon)

        #Starters list is complete, show off the starter pykemon
        for starter in starters:
            starter.show_stats()
            starter.move_info()

        #Present information to user
        print("\nProfessor Eramo presents you with three Pykemon: ")
        print("(1) - " + starters[0].name)
        print("(2) - " + starters[1].name)
        print("(3) - " + starters[2].name)
        choice = int(input("Which Pykemon would you like to choose: "))
        pykemon = starters[choice-1]

        return pykemon

    def get_attack(self, pykemon):
        """Get a users attack choice"""
        #Show the moves list using pykemon specific move names
        print("\nWhat would you like to do...")
        print("(1) - " + pykemon.moves[0])
        print("(2) - " + pykemon.moves[1])
        print("(3) - " + pykemon.moves[2])
        print("(4) - " + pykemon.moves[3])
        choice = int(input("Please enter your move choice: "))

        #Formatting
        print()
        print("------------------------------------------------------------------------------")

        return choice

    def player_attack(self, move, player, computer):
        """Attack the computer AI"""
        #Call the appropriate attack method based on the given move
        if move == 1:
            player.light_attack(computer)
        elif move == 2:
            player.heavy_attack(computer)
        elif move == 3:
            player.restore()
        elif move == 4:
            player.special_attack(computer)

        #Check to see if the computer has fainted
        computer.faint()


    def computer_attack(self, player, computer):
        """Let the computer AI attack the player"""
        #Randomly pick a move for the computer to execute
        move = random.randint(1, 4)

        #Call the appropriate attack method based on the given move
        if move == 1:
            computer.light_attack(player)
        elif move == 2:
            computer.heavy_attack(player)
        elif move == 3:
            computer.restore()
        elif move == 4:
            computer.special_attack(player)

        #Check to see if the player has fainted
        player.faint()


    def battle(self, player, computer):
        """Simulate a battle round. Faster Pykemon go first."""
        #Get the players move for the round
        move = self.get_attack(player)

        #If the player's pykemon is faster than the computer, they go first
        if player.speed >= computer.speed:
            #Player attacks
            self.player_attack(move, player, computer)
            if computer.is_alive:
                #Computer is still alive, let them attack
                self.computer_attack(player, computer)
        #The player's pykemon is slower than the computer, the computer goes first
        else:
            self.computer_attack(player, computer)
            if player.is_alive:
                #Player is still alive, let them attack
                self.player_attack(move, player, computer)


#The main code
#Narrative introduction
print("Welcome to Pykemon!")
print("Can you become the worlds greatest Pykemon Trainer???")
print("\nDon't worry! Prof Eramo is here to help you on your quest.")
print("He would like to gift you your first Pykemon!")
print("Here are three potential Pykemon partners.")
input("Press Enter to choose your Pykemon!")

#The main game loop
playing_main = True
while playing_main:
    #Create a game instance
    game = Game()

    #Choose your starter pykemon
    player = game.choose_pykemon()
    print("\nCongratulations Trainer, you have chosen " + player.name + "!")
    input("\nYour journey with " + player.name + " begins now...Press Enter!")

    #While your pykemon is alive, continue to do battle
    while player.is_alive:
        #Create a computer pykemon to battle
        computer = game.create_pykemon()
        print("\nOH NO! A wild " + computer.name + " has approached!")
        computer.show_stats()

        #While this enemy pykemon is alive and the player pykemon is alive, engage in battle
        while computer.is_alive and player.is_alive:
            game.battle(player, computer)

            #Both parties survived a round, show their current stats
            if computer.is_alive and player.is_alive:
                player.show_stats()
                computer.show_stats()
                #Formatting
                print("-----------------------------------------------------------------------------")

        #If the player survived the battle, increment battles_won
        if player.is_alive:
            game.battles_won += 1

    #The player has finally fainted
    print("\nPoor " + player.name + " has fainted...")
    print("But not before defeating " + str(game.battles_won) + " Pykemon!")

    #Ask the user if they want to play again
    choice = input("Would you like to play again (y/n): ").lower()
    if choice != 'y':
        playing_main = False
        print("Thank you for playing Pykemon!")
```