```python
#Classes Challenge 36: Pythonagachi Simulator App
import random

#Define the Creature class
class Creature():
    """Create a simple Tomogachi clone."""

    def __init__(self, name):
        """Initialize attributes"""
        self.name = name.title()

        #Attributes to track playing the game (0-10)
        self.hunger = 0
        self.boredom = 0
        self.tiredness = 0
        self.dirtiness = 0

        self.food = 2 #Represents food inventory
        self.is_sleeping = False #Bool to track if creature is sleeping
        self.is_alive = True #Bool to track if creature is alive


    def eat(self):
        """Simulate eating.  Each time you eat, take one food away from the inventory
            and randomly take a value away from hunger."""
        #First, make sure there is food available
        if self.food > 0:
            self.food -= 1
            self.hunger -= random.randint(1,4)
            print("Yum! " + self.name + " ate a great meal!")
        else:
            print(self.name + " doesn't have any food!  Better forage for some.")

        #If the hunger is less than zero, set it to zero
        if self.hunger < 0:
            self.hunger = 0


    def play(self):
        """Play a guessing game to lower the creatures boredom.
            If you win the game, lower the boredom even move."""
        #Simple guessing game
        value = random.randint(0,2)
        print("\n" + self.name + " wants to play a game.")
        print(self.name + " is thinking of a number 0, 1, or 2.")
        guess = int(input("What is your guess: "))

        #Lower the boredom attribute based on the users guess
        if guess == value:
            print("That is correct!!!")
            self.boredom -= 3
        else:
            print("WRONG! " + self.name + " was thinking of " + str(value) + ".")
            self.boredom -= 1

        #If the boredom is less than zero, set it to zero
        if self.boredom < 0:
            self.boredom = 0


    def sleep(self):
        """Simulate sleeping.  The only thing a player can do when the creature is sleeping
```

```
63              is try to wake up.  However, tiredness and boredom should decrease
    each round when sleeping"""
64          #Put the creature to sleep
65          self.is_sleeping = True
66          self.tiredness -= 3
67          self.boredom -= 2
68          print("Zzzzzzz.....Zzzzzzz.....Zzzzzzz.....")
69
70          #If tiredness or boredom is less than zero, set it to zero
71          if self.tiredness < 0:
72              self.tiredness = 0
73          if self.boredom < 0:
74              self.boredom = 0
75
76
77      def awake(self):
78          """Simulate randomly waking a creature up."""
79          #Creature has a 1/3 chance to randomly wake up
80          value = random.randint(0,2)
81          #If creature wakes up, set tiredness to zero!
82          if value == 0:
83              print(self.name + " just woke up!")
84              self.is_sleeping = False
85              self.tiredness = 0
86          else:
87              print(self.name + " won't wake up...")
88              self.sleep()
89
90
91      def clean(self):
92          """Simulate taking a bath to completely clean the creature"""
93          self.dirtiness = 0
94          print(self.name + " has taken a bath.  All clean!")
95
96
97      def forage(self):
98          """Simulate foraging for food.  This will increase the creatures food
    attribute
99              however, it will also increase their dirtiness"""
100         #Randomly find food from 0 to 4 pieces
101         food_found = random.randint(0,4)
102         self.food += food_found
103
104         #Creature gets dirty from foraging
105         self.dirtiness += 2
106
107         print(self.name + " found " + str(food_found) + " pieces of food!")
108
109
110     def show_values(self):
111         """Show the current information about the creature"""
112         #Show creature attributes
113         print("\nCreature Name: " + self.name)
114         print("Hunger (0-10): " + str(self.hunger))
115         print("Boredom (0-10): " + str(self.boredom))
116         print("Tiredness (0-10): " + str(self.tiredness))
117         print("Dirtiness (0-10): " + str(self.dirtiness))
118
119         print("\nFood Inventory: " + str(self.food) + " pieces")
120
121         #Show current sleeping status
122         if self.is_sleeping:
123             print("Current Status:  Sleeping")
124         else:
```

```
125             print("Current Status:  Awake")
126
127
128     def incriment_values(self, diff):
129         """User must set an arbitrary difficulty.  This will control how much
    "damage" you take
130             each round.  Update the current values of the creature based on this
    difficulty."""
131         #Increase the hunger and dirtiness regardless if the creature is awake
    or sleeping.
132         self.hunger += random.randint(0, diff)
133         self.dirtiness += random.randint(0, diff)
134
135         #If the creature is awake, he should be growing tired and growing bored.
136         if self.is_sleeping == False:
137             self.boredom += random.randint(0, diff)
138             self.tiredness += random.randint(0, diff)
139
140
141     def kill(self):
142         """Check for all conditions to kill or sleep the creature."""
143         #First two checks, will kill the creature
144         if self.hunger >= 10:
145             print(self.name + " has starved to death...")
146             self.is_alive = False
147         elif self.dirtiness >= 10:
148             print(self.name + " has suffered an infection and died...")
149             self.is_alive = False
150         #Next two checks, will put the creature to sleep
151         elif self.boredom >= 10:
152             self.boredom = 10
153             print(self.name + " is bored.  Falling asleep...")
154             self.is_sleeping = True
155         elif self.tiredness >= 10:
156             self.tirednress = 10
157             print(self.name + " is sleepy.  Falling asleep...")
158             self.is_sleeping = True
159
160
161 #Helper functions outside of the creature class
162 def show_menu(creature):
163     """Show the menu options for the player.  If the creature is sleeping, the
    player
164         can ONLY try to wake the creature up by default."""
165     #If the creature is sleeping, only allow the user to wake the creature.
166     #Hard code the value for sneaky users.
167     if creature.is_sleeping:
168         choice = input("\nEnter (6) to try and wake up: ")
169         choice = '6'
170     #Creature is awake, give full functionality to user
171     else:
172         print("\nEnter (1) to eat.")
173         print("Enter (2) to play.")
174         print("Enter (3) to sleep.")
175         print("Enter (4) to take a bath.")
176         print("Enter (5) to forage for food.")
177         choice = input("What is your choice: ")
178
179     return choice
180
181
182 def call_action(creature, choice):
183     """Given the players choice, call the appropriate class method."""
184     #Call the appropriate creature method
```

```
    if choice == '1':
        creature.eat()
    elif choice == '2':
        creature.play()
    elif choice == '3':
        creature.sleep()
    elif choice == '4':
        creature.clean()
    elif choice == '5':
        creature.forage()
    elif choice == '6':
        creature.awake()
    #User entered in invalid input.  Do not call any methods.
    else:
        print("Sorry, that is not a valid move.")


#The main code
print("Welcome to the Pythonagachi Simulator App")

#Set the difficulty level
difficulty = int(input("Please choose a difficulty level (1-5): "))
if difficulty > 5:
    difficulty = 5
elif difficulty < 1:
    difficulty = 1

#The overall main game loop
running = True
while running:
    #Get user input for creature name and make a creature
    name = input("What name would you like to give your pet Pythonagachi: ")
    player = Creature(name)

    rounds = 1
    #The game loop that simulates an individual round
    #This loop should run as long as the creature is alive
    while player.is_alive:

print("\n--------------------------------------------------------------------------------")
        print("Round #" + str(rounds))

        #An individual round should show values, get a players move, and call
the appropriate method
        player.show_values()
        round_move = show_menu(player)
        call_action(player, round_move)

        print("\nRound #" + str(rounds) + " Summary: ")

        #Summarize the effects of the current round
        player.show_values()
        input("\nPress (enter) to continue...")

        #Increment values and check for death
        player.incriment_values(difficulty)
        player.kill()

        #Round is over
        rounds += 1

    #The creatures has died.  Game over
    print("\nR.I.P.")
    print(player.name + " survived a total of " + str(rounds-1) + " rounds.")
```

```
247
248         #Ask the user to play again.
249         choice = input("Would you like to play again (y/n): ").lower()
250         if choice != 'y':
251             running = False
252             print("Thank you for playing Pythonagachi!")
```