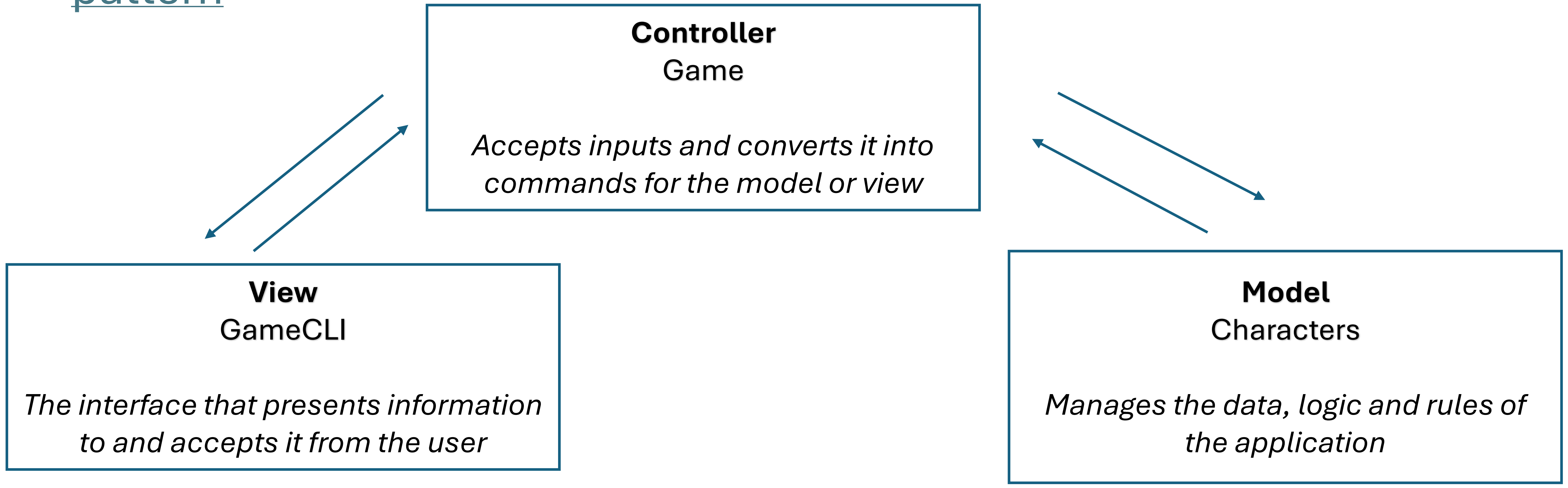
Using Object-Oriented Programming

Fighting Fantasy Battles – Game and Interface

Model View Controller

- We will program our Fighting Fantasy game using the MVC design pattern



Controller
Game

Accepts inputs and converts it into commands for the model or view

View
GameCLI

The interface that presents information to and accepts it from the user

Model
Characters

Manages the data, logic and rules of the application

Keeping track of the state of the game

- We can keep track of the state of the game, by creating a Game class
- *set_player* sets the player to be a player_character
- *choose_opponent* chooses a random opponent from the list of creatures

```
class Game:
    creatures = [Character("Dragon", 10, 22),
                 Character("Orc", 7, 10),
                 Character("Skeleton", 5, 8),
                 Character("Rat", 6, 6),
                 ]

    def __init__(self):
        self.opponent = None
        self.player = None
        self.round_result = None

    def choose_opponent(self):
        self.opponent = random.choice(self.creatures)
        self.creatures.remove(self.opponent)

    def set_player(self, player_character):
        self.player = player_character
```

Running a fight round and return status reports

- Write a method for the Game class called `.resolve_fight_round`, which will fight one round and record the result in the `.round_result` attribute.
- Write a method called `.return_character_status` which will return a string with information about the player and their opponent (which it can get from each Character instance):

```
Sir Andrew has skill 8 and stamina 15
Giant Rat has skill 6 and stamina 6
```

Running a fight round and return status reports

```
def resolve_fight_round(self):
    self.round_result = self.player.fight_round(self.opponent)

def return_characters_status(self):
    msg = (self.player.return_character_status() + "\n" +
           self.opponent.return_character_status())
    return msg
```

Running a fight round and return status reports

- Write a method called `.return_round_result` which will return a string with information about what the characters' rolls were and who won the round.
- Again the information about the character rolls can be drawn from the Character instances.

```
Sir Andrew rolled 8 for a total score of 17
Giant Rat rolled 5 for a total score of 11
Player won this round
```

Running a fight round and return status reports

```python
def return_round_result(self):
    msg = (self.player.return_roll_status() + "\n" +
           self.opponent.return_roll_status() + "\n")
    if self.round_result == "won":
        msg += 'Player won this round\n'
    elif self.round_result == "lost":
        msg += 'Player lost this round\n'
    else:
        msg += 'This round was a draw\n'
    return msg
```

Creating a Command Line Interface

- Finally, we need to create a way of interacting with the game.
- Create a CLI which:
  - Initiates a game
  - Allows the player to choose a name
  - Fights successive rounds until the player chooses to flee or either the player or their opponent is dead

```
Welcome to Fighting Fantasy Battle
Enter the name for your character: Sir Andrew
Welcome Sir Andrew
You have skill = 8 and stamina = 20

You will be fighting Skeleton
Skeleton has skill 5 and stamina 8

Sir Andrew has skill 8 and stamina 20
Skeleton has skill 5 and stamina 8

Would you like to fight a round (y/n)? y
Sir Andrew rolled 10 for a total score of 18
Skeleton rolled 7 for a total score of 12
Player won this round
…
```

Creating a Command Line Interface

```python
class GameCLI:
    """“Initialises a game class and launches the script to run the game.”""”
    def __init__(self):
        self.game = Game()
        self.run_game()

    def run_game(self):
        """Welcomes the player to Fighting Fantasy - asks for a player_name
          calls self.game methods to set the player, then runs self.fight_opponent"""

    def fight_opponent(self):
        """Chooses an opponent and displays their stats, then runs self.fight_battle"""

    def fight_battle(self):
        """Continues to fight rounds until the player chooses to quit or either player or opponent are dead."""

if __name__ == "__main__":
    GameCLI()
```

```
class GameCLI:
   def __init__(self):
      self.game = Game()
      self.run_game()

   def run_game(self):
      """Welcomes the player to Fighting Fantasy - asks for a player_name
         calls self.game methods to set the player"""
      print('Welcome to Fighting Fantasy Battle')
      player_name = input("Enter the name for your character: ")

self.game.set_player(PlayerCharacter.generate_player_character(player_name))
      print(f'Welcome {player_name}')
      print(self.game.player.return_character_status())
      self.fight_opponent()

   def fight_opponent(self):
      """Chooses an opponent and displays their stats"""
      self.game.choose_opponent()
      print(f'You will be fighting {self.game.opponent}')
      print(self.game.opponent.return_character_status() + '\n')
      self.fight_battle()
```

```
 def fight_battle(self):
      """Continues to fight rounds until the player chooses to quit or either player or
opponent are dead."""
      continue_battle = True
      while continue_battle:
         print(self.game.return_characters_status())
         print()
         action = input("Would you like to fight a round (y/n)? ").strip().lower()
         if action == 'n':
            print("You flee in terror!")
            continue_battle = False
         else:
            self.game.resolve_fight_round()
            print(self.game.return_round_result())
            if self.game.player.is_dead:
               print('You died')
               continue_battle = False
            if self.game.opponent.is_dead:
               print(f"You defeated the {self.game.opponent.name}")
               continue_battle = False

if __name__ == "__main__":
   GameCLI()
```

Extras

- What could we add?
  - Allow the player to keep facing random monsters
  - Fleeing could mean a random penalty
  - Create a score – how many random monsters can you kill before you die?

Analysis of the Fighting Fantasy problem

Simulators for Fighting Fantasy already exist on the www.

- Look at https://fsyth.github.io/fighting-fantasy/dom/www/ and https://fanbooks.fightingfantasy.net/combat_simulator.php
- Take screen shots of each simulation
  - Annotate with features that you like
  - Provide a critique of features that you don’t like
  - What could you build into your own Fighting Fantasy program?