ring.py

```
class Ring:
    def __init__(self, num_process=5):
        self.num_process = num_process
        self.coordinator = 5
        self.active_processes = set(range(1, num_process + 1))


    def election(self, process_id):
        if self.coordinator is None:
            # Only one process in the system
            self.coordinator = process_id
            print(f"Process {process_id} is the coordinator.")
            return


        if process_id not in self.active_processes:
            print(f"Process {process_id} is not active.")
            return


        highest_id = process_id
        next_process = (process_id % self.num_process) + 1


        while next_process != process_id:
            if next_process in self.active_processes:
                print(
                    f"Process {process_id} is passing election message to process {next_process}."
                )
                if next_process > highest_id:
                    highest_id = next_process
            else:
                print(
                    f"Process {next_process} is down and cannot receive the election message."
                )
            next_process = (next_process % self.num_process) + 1


        self.coordinator = highest_id
        print(f"Process {self.coordinator} is the coordinator.")


    def start_election(self, process_id):
        if process_id not in self.active_processes:
            print(f"Process {process_id} is not active.")
            return

```

```
48
49          print(f"Process {process_id} starts the election process.")
50          self.election(process_id)
51
52
53      def bring_up_process(self, process_id):
54          if process_id in self.active_processes:
55              print(f"Process {process_id} is already up.")
56              return
57
58
59          self.active_processes.add(process_id)
60          print(f"Process {process_id} is up.")
61
62
63      def bring_down_process(self, process_id):
64          if process_id not in self.active_processes:
65              print(f"Process {process_id} is already down.")
66              return
67
68
69          self.active_processes.remove(process_id)
70          print(f"Process {process_id} is now down.")
71
72
73          if self.coordinator == process_id:
74              self.start_election(process_id)
75
76
77      def print_active_processes(self):
78          print("Active processes:")
79          for process_id in self.active_processes:
80              print(f"Process {process_id}")
81
82
83      def print_coordinator(self):
84          if self.coordinator is None:
85              print("Coordinator: None")
86          else:
87              print(f"Coordinator: Process {self.coordinator}")
88
89
90
91
92  if __name__ == "__main__":
93      ring = Ring()
94
95
96      while True:
97          print("-------------------------------------")
```

```
        print("1) Start Election")
        print("2) Bring Up Process")
        print("3) Bring Down Process")
        print("4) Print Active Processes")
        print("5) Print Coordinator")
        print("6) Exit")


        choice = int(input("Enter choice: "))


        if choice == 1:
            process_id = int(input("Enter process id to start the election: "))
            ring.start_election(process_id)


        elif choice == 2:
            process_id = int(input("Enter process id to bring up: "))
            ring.bring_up_process(process_id)


        elif choice == 3:
            process_id = int(input("Enter process id to bring down: "))
            ring.bring_down_process(process_id)


        elif choice == 4:
            ring.print_active_processes()


        elif choice == 5:
            ring.print_coordinator()


        else:
            break

```

bully.py

```python
class Bully:
    def __init__(self, num_process=5):
        # Initialize the Bully object with the number of processes and their states
        self.num_process = num_process
        self.state = [True for _ in range(num_process)]
        self.leader = num_process


    def election(self, process_id):
        # Perform the election algorithm to elect a coordinator
        print(f"Process {process_id} is sending election messages to higher processes")
        cod = process_id
        for i in range(process_id + 1, self.num_process + 1):
            if self.state[i - 1]:
                print(
                    f"Process {process_id} is sending election message to process {i}"
                )
                cod = i


        print(f"Process {cod} is sending coordinator message to all")


        # Update the leader to the elected coordinator
        self.leader = cod
        print(f"Process {self.leader} is now coordinator.")


    def up(self, process_id):
        # Bring up a process and trigger an election if necessary
        if self.state[process_id - 1]:
            print(f"Process {process_id} is already up")
            return
        else:
            self.state[process_id - 1] = True
            print(f"Process {process_id} is up")
            self.election(process_id)


    def down(self, process_id):
        # Bring down a process and initiate a new election if the leader is down
        if not self.state[process_id - 1]:
            print(f"Process {process_id} is already down.")
        else:
            self.state[process_id - 1] = False
            print(f"Process {process_id} is now down")


```

```
            if self.leader == process_id:
                # If the leader is down, randomly select a new active process and trigger an
election
                active = [i for i, _ in enumerate(self.state) if i]
                import random


                index = random.randint(0, len(active) - 1)
                self.election(active[index])


    def message(self, process_id):
        # Send a message and check if the coordinator is active
        if self.state[process_id - 1]:
            if self.state[self.leader - 1]:
                print("OK")
            else:
                # If the coordinator is down, initiate a new election
                self.election(process_id)
        else:
            print(f"Process {process_id} is down.")




if __name__ == "__main__":
    # Create a Bully object
    bully = Bully()


    print("5 Active processes are:")
    print("Processes up = p1 p2 p3 p4 p5")
    print(f"Process {bully.leader} is the coordinator")


    choice = 5


    while choice != 4:
        print("-------------------------------------")
        print("1) Up a process")
        print("2) Down a Process")
        print("3) Send a Message")
        print("4) Exit")


        choice = int(input("Enter choice: "))


        if choice == 1:
```

```
            process_id = int(input("Enter process id: "))
            bully.up(process_id)


        elif choice == 2:
            process_id = int(input("Enter process id: "))
            bully.down(process_id)


        elif choice == 3:
            process_id = int(input("Enter process id: "))
            bully.message(process_id)


        else:
            break

```