Furkan YAMAK- Mine ALTUĞ

(200316027-200316021)

## STRATEGY TİMEOUT

We chose this strategy because it is very easy to handle when deadlock is occurred and, more easier than another strategies.

## CHANGES

- if statement is added to __exit__ method:

```
def __exit__(self, exc_type, exc_value, traceback):
    if self.picked_up == True:
        self.lock.release()
        self.picked_up = False
        self.owner = -1
```

- added 2 new function

```
def check_forks(philosophers: list[Philosopher], forks: list[Fork]):
    system_time = 0
    while True:
        eating_philosophers: int = sum(
            philosopher.eating for philosopher in philosophers
        )

        if eating_philosophers == 1 or eating_philosophers == 2:
            system_time = 0
        else:
            time.sleep(1)
            system_time += 1
            if system_time > 15:
                fork_index = random.randint(0,len(forks)-1)
                fork = forks[fork_index]
                fork.__exit__(*sys.exc_info())
                system_time = 0
```

```
def philosopher_information(philosophers: list[Philosopher]):

    while True:
        a = ""
        for p in philosophers:
            a += (
                f"Philosopher {p.index} LEFT: {p.left_fork.owner} - RIGHT: 
{p.right_fork.owner}"
                f" ({p.eating})\n"
            )
        print(a)
        time.sleep(0.5)
        print("\033[H\033[J")
```

## EXPLANATION

**if statement in the __exit__ method:**

Since we manually called the __exit__ method in the fork class, we added a condition to prevent it from being released again.

**check_forks Function:**

Controls the number of philosophers eating. He allows only one or two philosophers to eat during a certain period. Randomly selects the fork that releases the lock.

The condition if eating_philosophers == 1 or eating_philosophers == 2 resets system_time if the number of philosophers eating is 1 or 2. This initiates a type of check by resetting system_time when the number of philosophers eating is checked and when that number reaches a certain limit (either 1 or 2).If the number of philosophers eating is not 1 or 2, one second of sleep time is added and system_time is increased by one.The if system_time > 15: condition checks that the number of philosophers eating during a certain period of time does not remain constant. If the time is more than 15 seconds, a fork is selected and that fork is unlocked. This approach aims to reduce the possibility of deadlock by controlling a specific situation and intervening when necessary.

**philosopher_information Function**

It refers to a function that prints philosophers' fork usage information on the screen at regular intervals. The purpose of the function is to visually display philosophers' fork usage statuses and information about these fork usages at regular intervals. The function follows these steps:

It works in an endless loop. For each philosopher, it accumulates the relevant fork usage information on a string. By creating this array in a loop, it shows the owner of each philosopher's left and right fork and their eating status (eating or not). It prints the string to the screen and then clears the screen (this provides a continuously updated output). By adding a wait for a certain period of time (0.5 seconds), it ensures that the information is updated at regular intervals.