

# Teoria Współbieżności

---

## Laboratorium 7 - Wzorzec projektowy Active Object.

Magdalena Pastuła

Data laboratorium: 17.11.2020

### 1. Zadanie do wykonania.

Celem tego laboratorium było zaimplementowanie problemu producentów i konsumentów przy wykorzystaniu wzorca projektowego Active Object.

### 2. Koncepcja rozwiązania.

Rozwiązanie wykorzystuje wzór projektowy Active Object. Dodatkowo, do zaimplementowania części **Future** wzorca wykorzystano klasę Javy o nazwie **CompletableFuture**.

### 3. Implementacja i wyniki.

Poniżej znajduje się implementacja wątku schedulera:

```
public void run()
{
    while(true)
    {
        var request = queue.poll();
        if(request != null && request.guard()){
            request.call();
        } else if(request != null){
            queue.add(request);
        }
    }
}
```

Dodatkowo, poniżej znajduje się implementacja klasy AddRequest, która obsługuje dodawanie elementów do bufora.

```
public class AddRequest implements MethodRequest {
    private Buffer buffer;
    private Integer value;
    private CompletableFuture<Integer> future;

    public AddRequest(Buffer buffer, Integer value, CompletableFuture<Integer>
future) {
        this.buffer = buffer;
        this.value = value;
    }
}
```

```

        this.future = future;
    }

    @Override
    public boolean guard() {
        return !buffer.isFull();
    }

    @Override
    public void call() {
        buffer.add(value);
        future.complete(value);
    }
}

```

Również poniżej znajduje się implementacja klasy Proxy:

```

public class Proxy {
    private Scheduler scheduler;
    private Buffer buffer;

    public Proxy(Scheduler scheduler, Buffer buffer) {
        this.scheduler = scheduler;
        this.buffer = buffer;
    }

    CompletableFuture<Integer> put(Integer toPut) {
        var future = new CompletableFuture<Integer>();
        scheduler.enqueue(new AddRequest(buffer, toPut, future));
        return future;
    }

    CompletableFuture<Integer> get() {
        var future = new CompletableFuture<Integer>();
        scheduler.enqueue(new RemoveRequest(buffer, future));
        return future;
    }
}

```

Poniżej znajduje się klasa ActiveObject:

```

public class ActiveObject {
    private Buffer buffer;
    private Scheduler scheduler;
    public Proxy proxy;

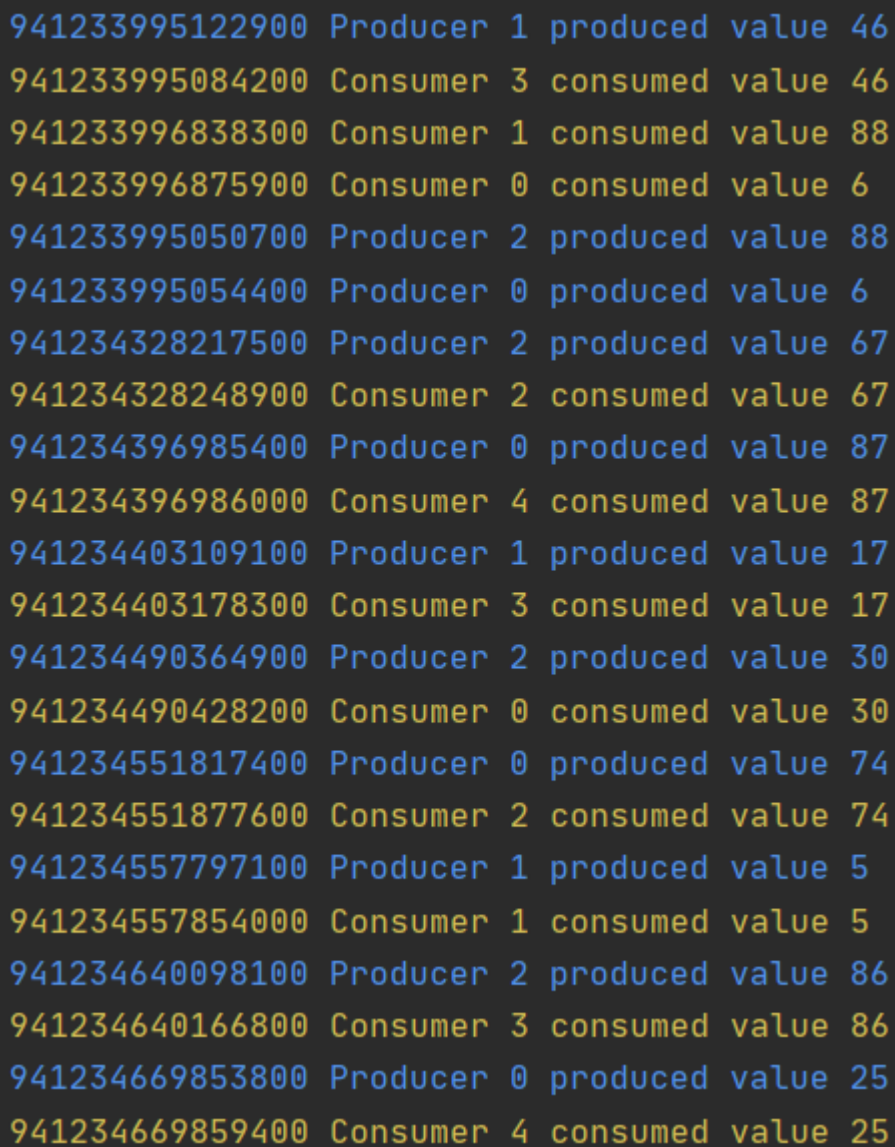
    public ActiveObject(int queueSize){
        buffer = new Buffer(queueSize);
        scheduler = new Scheduler(queueSize);
    }
}

```

```
        proxy = new Proxy(scheduler, buffer);
        scheduler.start();
    }
}
```

Natomiast cały kod źródłowy znajduje się w załączonym archiwum.

Poniżej znajduje się zdjęcie kilku pierwszych logów do pokazania działania programu. Dla ułatwienia logi producentów wypisują się w kolorze niebieskim, a konsumentów w kolorze żółtym. Wartości czasowe na początku komunikatów to wartości zwrócone przez funkcję `System.nanoTime()`.



```
941233995122900 Producer 1 produced value 46
941233995084200 Consumer 3 consumed value 46
941233996838300 Consumer 1 consumed value 88
941233996875900 Consumer 0 consumed value 6
941233995050700 Producer 2 produced value 88
941233995054400 Producer 0 produced value 6
941234328217500 Producer 2 produced value 67
941234328248900 Consumer 2 consumed value 67
941234396985400 Producer 0 produced value 87
941234396986000 Consumer 4 consumed value 87
941234403109100 Producer 1 produced value 17
941234403178300 Consumer 3 consumed value 17
941234490364900 Producer 2 produced value 30
941234490428200 Consumer 0 consumed value 30
941234551817400 Producer 0 produced value 74
941234551877600 Consumer 2 consumed value 74
941234557797100 Producer 1 produced value 5
941234557854000 Consumer 1 consumed value 5
941234640098100 Producer 2 produced value 86
941234640166800 Consumer 3 consumed value 86
941234669853800 Producer 0 produced value 25
941234669859400 Consumer 4 consumed value 25
```

#### 4. Wnioski z ćwiczenia.

Wydawać by się mogło, że aplikacja nie działa, jak powinna: na przykład już w trzeciej linii logu konsument pobiera wartość zanim zostanie dodana do bufora. Jednakże, po porównaniu czasu wypisania komunikatu, który znajduje się na samym jego początku, można zauważyć, że tak naprawdę element został najpierw dodany, a potem zdjęty z bufora. Natomiast różnica czasowa między tymi dwoma wydarzeniami jest na tyle mała, że prawdopodobnie konsument jako pierwszy dostał dostęp do wypisania komunikatu.

Dodatkowo, rozbieżności czasowe między wypisaniem komunikatu o dodaniu danego elementu do bufora a jego zdjęciem mogą wynikać z faktu, że producent wypisuje swój komunikat już po wykonaniu akcji.

## 5. Bibliografia.

1. [Opis wzorca](#)
2. [Przykład implementacji w Javie](#)
3. [Dokumentacja klasy CompletableFuture](#)