



Sprawozdanie

LABORATORIUM 7: ZACHOWANIA⁽¹⁾

Kamil Udziela | RSI | Nr.indeksu: 286133

Przebieg ćwiczenia:

Ćwiczenie przedstawione w scenariuszu nr. 7 ma na celu zapoznanie się z zachowaniami agentów w platformie JADE w języku JAVA.

Etapy:

1). Pierwszym etapem ćwiczenia było stworzenie klasy o nazwie *klasa_1*. Do zadań tego agenta należeć miało wypisywanie odpowiedniej komendy zarówno na początku działania jak i przed swoim usunięciem.

- zawsze po uruchomieniu, na początku nasz agent ma wypisać napis: „startuję”.
- zawsze przed swoim usunięciem, na końcu nasz agent ma wypisać napis: „zaraz się usunę”.

Tworzymy klasę i dołączamy do niej klasę wbudowaną *Agent*, aby to było możliwe, należy zaimportować odpowiednie biblioteki (konkretnie *jade.core.Agent*).

Tworzenie klasy:

```
import jade.core.Agent;  
public class klasa_1 extends Agent
```

Ciało naszej klasy wygląda następująco:

```
protected void setup() {  
    System.out.println("startuję!");  
}  
protected void takeDown() {  
    System.out.println("zaraz się usunę!");  
}
```

W metodzie **setup()** znajduję się formuła wypisania pierwszego napisu, wypisanie to wykona się po uruchomieniu naszego agenta. Natomiast **takeDown()** to metoda, która wywoływana jest tuż przed zakończeniem działania agenta. Można w niej umieścić np. operacje czyszczenia.

2). Kolejnym zadaniem było utworzenie *klasy_2*, analogicznie do *klasy_1*. Do agenta dodałem zachowanie jednokrotnego wypisania na ekranie napisu „wykonuje”. Następnie uruchomiłem agenta inspektora. Opisane wyżej zachowania dodałem w metodzie **setup()**.

```
protected void setup() {
    System.out.println("startuje!");

    addBehaviour(new OneShotBehaviour() {
        @Override
        public void action() {
            System.out.println("wykonuje!");
        }
    });
}
```

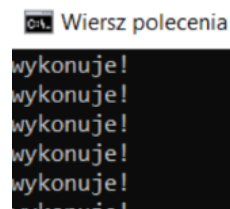


```
protected void setup() {
    System.out.println("startuje!");

    addBehaviour(new CyclicBehaviour() {
        @Override
        public void action() {
            System.out.println("wykonuje!");
        }
    });
}
```

3). Następnym krokiem zadania do wykonania w scenariuszu do lab. Nr 7 było utworzenie klasy o nazwie *klasa_3*. Jej zadaniem było cykliczne (wielokrotne) wypisywanie słowa „wykonuję”. Podobnie jak wyżej, zachowanie to dodałem w metodzie **setup()** w *klasie_3*.

W przeciwieństwie do poprzedniego etapu, agent wypisywał podany przez nas napis przez cały swój czas działania.



4). Analogicznie do *klasy_1* utworzyłem klasę o nazwie *klasa_4*. Tym razem do agenta dodałem zachowanie polegające na wykonaniu trzech kroków:

I krok: wypisanie napisu „pierwszy krok”.

II krok: wypisanie napisu „drugi krok”

III krok: wypisanie napisu „trzeci krok” i usunięcie zachowania z puli zachowań agenta.

W klasie **setup()** należało więc stworzyć trzy kroki za pomocą *switch...case*, aby w każdym z kroków wypisywany napis się różnił. Niezbędne było także stworzenie kroku (*step*), który musimy zwiększać w każdym z „case’ów”.

Aby nasz agent po wypisaniu tych trzech napisów zakończył działanie należy w trzecim kroku również zwiększyć naszą zmienną *step*.

```
private int step=0;
protected void setup() {
    System.out.println("startuje!");
    addBehaviour(new Behaviour() {
        @Override
        public void action() {
            switch(step){
                case 0:
                    System.out.println("pierwszy krok!");
                    step++;
                    break;
                case 1:
                    System.out.println("drugi krok!");
                    step++;
                    break;
                case 2:
                    System.out.println("trzeci krok!");
                    step++;
                    break;
            }
        }
    })
}
```

```
protected void setup() {
    System.out.println("startuje!");

    Scanner scanner = new Scanner(System.in);
    addBehaviour( new Behaviour()
    {
        public void action() {
            System.out.print("Podaj liczbę całkowitą: ");
            int number = scanner.nextInt();
            if(number>0)
            {
                System.out.println("liczba dodatnia!");
            }
            if(number<0)
            {
                System.out.println("liczba ujemna!-usuwa Agent");
                removeBehaviour( this);
            }
        }
    })
}
```

5). Następnie, stworzyłem klasę o nazwie *klasa_5*. Jej zadaniem było pobieranie z klawiatury liczby całkowitej. Kolejny raz wykonałem tą klasę analogicznie do *klasy_1*. W klasie **setup()** znalazł się warunek, który sprawdza czy podana przez nas liczba jest ujemna czy dodatnia. Jeżeli jest ujemna to usuwa to zachowanie.

6). W nowoutworzonej klasie (*klasa_6*), tak zmodyfikowałem kod z *klasa_5*, aby na początku zachowania nasz agent wypisywał napis „zachowanie startuje”.

```
addBehaviour( new Behaviour()
{
    public void action() {
        System.out.println( "zachowanie startuje" );
    }
    @Override
    public boolean done() {
        return true; }
});
```

Podczas gdy zachowanie się kończy,

```
addBehaviour( new Behaviour()
{
    public void action() {
        System.out.print("Podaj liczbę całkowitą: ");
        int number = scanner.nextInt();
        if(number>0) {
            System.out.println("liczba dodatnia!");
        }
    }
})
```

miał pojawić się napis „Zachowanie zakończone”. Zrobiłem to analogicznie do poprzedniego zadania. Znowu za pomocą metody **action()** dodałem napis.

7). Następnym krokiem było utworzenie klasy_7. Do zadania z pkt. 4 należało dodać dwa zachowania:

Zachowanie nr.1 (jednokrotne): w metodzie **setup()**, wypisanie napisu „pierwsze”.

```
protected void setup() {
    System.out.println("startuje!");
    OneShotBehaviour oneShotBehaviour1=new OneShotBehaviour() {
        @Override
        public void action() {
            System.out.println("pierwsze");
        }
    };
    addBehaviour(oneShotBehaviour1);
}
```

Zachowanie nr.2 (generyczne): w pierwszym bloku, wypisanie napisu „drugiego”.

```
addBehaviour(new Behaviour() {
    @Override
    public void action() {
        switch(step){
            case 0:
                OneShotBehaviour oneShotBehaviour2=new OneShotBehaviour() {
                    @Override
                    public void action() {
                        System.out.println("drugie");
                    }
                };
                addBehaviour(oneShotBehaviour2);
            default:
                break;
        }
    }
});
```

8). W ostatniej z utworzonych w tym scenariuszy klas, o nazwie *klasa_8* do naszego agenta dodałem (w metodzie **setup()**):

- Wypisywanie „mały tick” co 2 sekundy.
- Wypisywanie „duży tick” co 5 sekund.

```
Behaviour smallTick=new TickerBehaviour( @ this, period: 2000) {
    @Override
    protected void onTick() {
        System.out.println("mały tick");
    }
};
```

```
Behaviour bigTick=new TickerBehaviour( @ this, period: 5000) {
    @Override
    protected void onTick() {
        System.out.println("duży tick");
    }
};
```

Aby dodać te zachowania, należy po ich zinterpretowaniu dopisać w metodzie **setup()** następujące wywołanie metod:

```
addBehaviour(smallTick);
addBehaviour(bigTick);
```

- Usunięcie zachowania z drugiego punktu po 50-ciú sekundach.

```
addBehaviour(new WakerBehaviour( a: this, timeout: 50000) {  
    protected void handleElapsedTimeout() {  
        removeBehaviour(bigTick);  
    }  
});
```

- Usunięcie całego agenta po 100 sekundach.

```
addBehaviour(new WakerBehaviour( a: this, timeout: 100000) {  
    protected void handleElapsedTimeout() {  
        myAgent.doDelete();  
    }  
});
```