Scenariusz 7

Badanie zachowań

1. Utwórz klasę agenta o nazwie klasa_1. Agent ten powinien:
 a. zawsze na samym początku wypisywać „startuję”,
b. zawsze przed swoim usunięciem wypisywać „zaraz się usunę”.

2. Utwórz klasę agenta o nazwie klasa_2 na podstawie kodu klasa_1. Do agenta dodaj zachowanie polegające na jednokrotnym wykonaniu operacji wypisania na ekranie słowa „wykonuję”. Uruchom agenta introspektora – jakie zachowania są widoczne?

W introspektorze pokazywane są aktualnie wykonywane zadania

```
protected void setup() {
    System.out.println("Agent "+getLocalName()+" started.");


    addBehaviour(new OneShotBehaviour( a this) {
        public void action() { System.out.println("wykonuje"); }

        @Override
        public int onEnd() {
            System.out.println("Agent "+getLocalName()+" terminated.");
            return super.onEnd();
        }
    });
```

3. Utwórz klasę agenta o nazwie klasa_3 na podstawie kodu klasa_1. Do agenta dodaj zachowanie polegające na wielokrotnym (cyklicznym) wykonaniu operacji wypisania na ekranie słowa „wykonuję”. Uruchom agenta introspektora – jakie zachowania są widoczne?

Widoczne jest zachowanie cykliczne, które wciąż się wykonuje.

```
addBehaviour(new CyclicBehaviour( a this) {
    public void action() {
        System.out.println("wykonuje");
    }
```

4. Utwórz klasę agenta o nazwie klasa_4 na podstawie kodu klasa_1. Do agenta dodaj zachowanie „generyczne”, polegające na wykonaniu trzech kroków:
a. W pierwszym kroku wypisuje „pierwszy krok”,
b. W drugim kroku wypisuje „drugi krok”,
c. W trzecim kroku wypisuje „trzeci krok” i zachowanie zostaje usunięte z puli zachowań agenta.

```
private class ThreeStepBehaviour extends Behaviour {
    private int step = 1;
    public void action() {
        switch (step) {
            case 1:
                addBehaviour(new CyclicBehaviour() {
                    @Override
                    public void action() {
                        System.out.println("1"); }
                });
                break;
            case 2:
                addBehaviour(new OneShotBehaviour() {
                    @Override
                    public void action() {
                        System.out.println("2");
                    }
                });
                break;
            case 3:
                addBehaviour(new CyclicBehaviour() {
                    @Override
                    public void action() {
                        System.out.println("3");
```

```
Agent xx started.
1
2
1
3
1
3
1
3
1
3
1
3
```

5. Utwórz agenta o nazwie klasa_5 na podstawie kodu klasa_1. Do agenta dodaj zachowanie, które będzie polegało na pobieraniu z klawiatury liczby całkowitej. Jeśli użytkownik poda liczbę ujemną, to zachowanie zostaje usunięte.

6. Utwórz klasę agenta o nazwie klasa_6 na podstawie kodu klasa_5. Zmodyfikuj kod tak, aby zawsze zachowanie na początku wypisywało „zachowanie startuje”, a na samym końcu wypisywało „zachowanie zakończone”.

```
public int step=1;
public boolean end=false;
Scanner scan=new Scanner(System.in);
public void action() {

    step=scan.nextInt();

    System.out.println(step);
    if(step>0)
        step=1;
    else step=2;
    switch (step) {
        case 1:
            //addBehaviour(new ThreeStepBehaviour());
            System.out.println("Liczba poprawna");
            break;
        case 2:
            end=true;
            System.out.println("koncze działanie");

            break;
    }
    }
        public boolean done () {
            return end==true;
        }

        public int onEnd () {
            myAgent.doDelete();
            takeDown();
            return super.onEnd();
        }
```

7. Utwórz klasę agenta o nazwie klasa_7 na podstawie kodu klasa_4. Do istniejącego zachowania „generycznego” dodaj dwa kolejne:
a. Pierwsze ma być dodawane na poziomie metody setup() agenta i ma polegać na jednokrotnym wypisaniu „pierwsze”
b. Drugie ma być dodane z poziomu zachowania „generycznego” – dokładnie ma być dodane w pierwszym kroku i ma polegać na jednokrotnym wypisaniu „drugie”. Zaobserwuj działanie agenta – jak wytłumaczyć kolejność wypisywania komunikatów – rozrysuj kolejkę zachowań.

```
switch (step) {
    case 1:
        // Perform operation 1: print out a message
        addBehaviour(new CyclicBehaviour() {
            @Override
            public void action() {
                System.out.println("Cyclic");
            }
        });
        break;
    case 2:
        // Perform operation 2: Add a OneShotBehaviour
        System.out.println("Operation 2. Adding one-shot behaviour");
        myAgent.addBehaviour(new OneShotBehaviour(myAgent) {
            public void action() { System.out.println("One-shot"); }
        });
        break;
```

```
Agent xx started.
pierwsze
Operation 2. Adding one-shot behaviour
Cyclic
One-shot
Cyclic
Cyclic
Cyclic
```

pierwsze

cyclic

dodanie 2

one shot

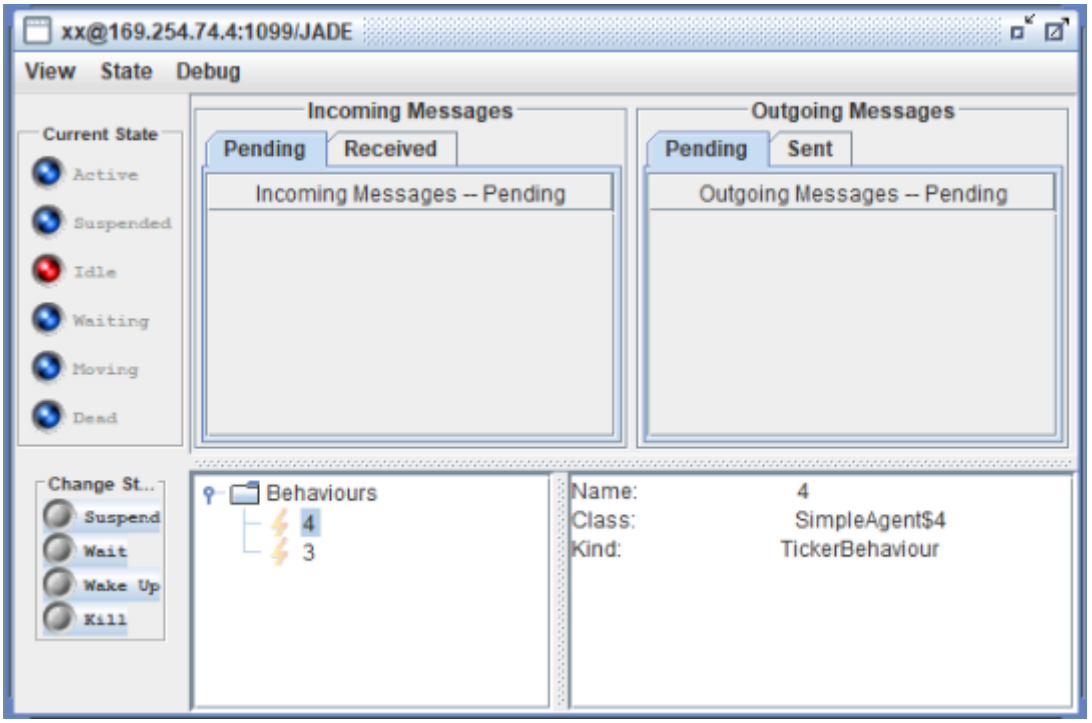
Dodawane w setup zachowanie, usuwa sie po wykonaniu.
Cykliczne zachowanie w generycznym, wykona sie i przechodzi na koniec kolejki.
Następnie wypisywane jest zachowanie drugie, które dodaje zachowanie na koniec kolejki.
Następne w kolejności jest zachowanie cykliczne, a potem dopiero zachowanie które wcześniej dodaliśmy

8. Utwórz klasę agenta o nazwie klasa_8 na podstawie kodu klasa_1. Do agenta dodaj zachowania, które spowodują:
a. Wypisanie „mały tick” co 2 sekundy,
b. Wypisaniu „duży tick” co 5 sekund,
c. Po 50 sekundach usunięcie zachowania z punktu b,
d. Po 100 sekundach usunięcie całego agenta. Przeanalizuj działanie agenta z użyciem Introspektora

```
final int[] step1 = {0};

addBehaviour(new TickerBehaviour( a: this, period: 2000) {
    @Override

    protected void onTick() {
        step1[0]++;
        System.out.println(step1[0]+"malytick");
        if(step1[0]==50)
        {
            this.stop();
            this.done();
        }

    }
});

int[] step2={0};

addBehaviour(new TickerBehaviour( a: this, period: 5000) {
    @Override

    protected void onTick() {
        step2[0]++;
        System.out.println(step2[0]+"duzyick");
        if(step2[0]==10)
        {
            this.stop();
            this.done();
```

```
Agent xx started.
zaczynam zadanie
1malytick
2malytick
1duzyick
3malytick
4malytick
5malytick
2duzyick
6malytick
7malytick
3duzyick
```

Przy użyciu Introspektora:



Po usunięciu zadania “duży tick”