Klaudia Dziewit
Inżynieria Obliczeniowa
III rok gr. 1

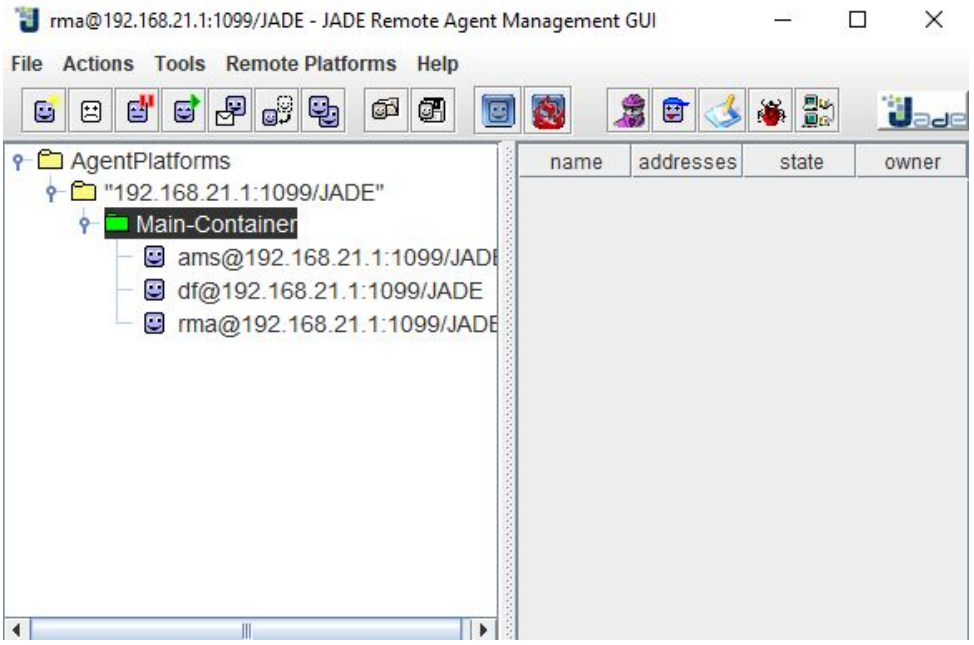
Rozproszona sztuczna inteligencja - ćwiczenia 7

Wykonanie

Wykonanie zadania rozpoczęłam od utworzenia klasy agenta o nazwie *class_1*. Agent ten powinien przy każdorazowym uruchomieniu wypisywać komunikat *"Agent is starting"*, natomiast przed swoim usunięciem *"Agent is deleting"*. Fragment kodu:

```
public class class_1 extends Agent{
    protected void setup() {
        System.out.println("Agent is starting");
        System.out.println("Agent is deleting");
        doDelete();
    }
}
```

Agent nie jest już dostępny w kontenerze.

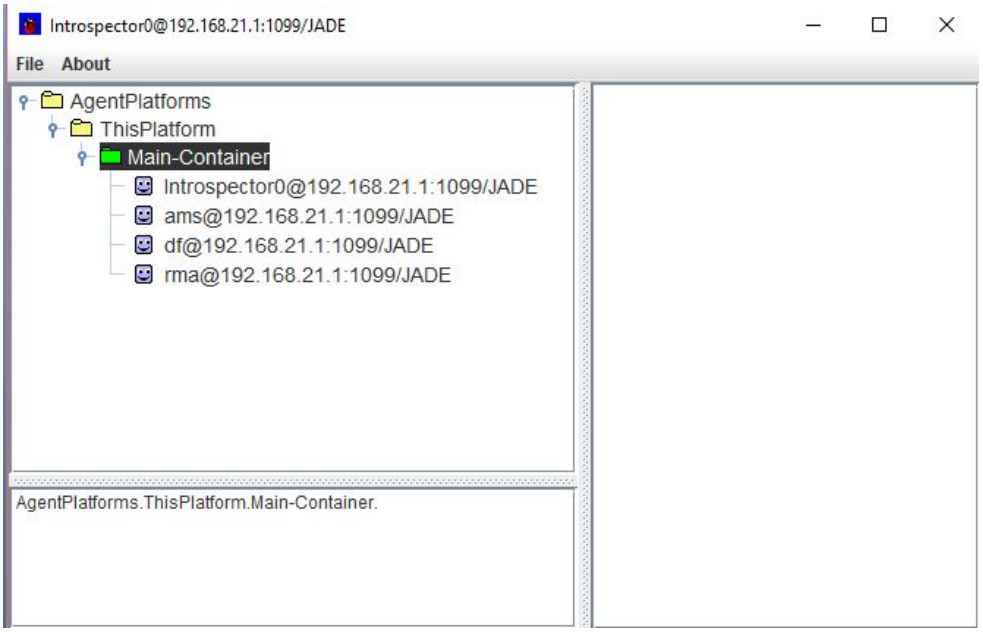


Następnie utworzyłam klasę agenta o nazwie *class_2*, na podstawie kodu *class_1*. Zmodyfikowałam ten fragment jednak, poprzez dodanie jednokrotnego zachowania wypisania na ekranie słowa wyrażenia *"In progress..."*. W tym celu skorzystałam z klasy *OneShootBehaviour*, gdzie metoda action wykonuje się tylko raz. Fragment kodu:

```
public class class_2 extends Agent{
    protected void setup() {
        System.out.println("Agent is starting");

        addBehaviour(new OneShotBehaviour(this) {
            public void action() {
                System.out.println("In progress...");
                System.out.println("Agent is deleting");
                doDelete();
            }
        });
    }
}
```

Operacje są zbyt krótkie aby móc je dostrzec w Introspectorze.

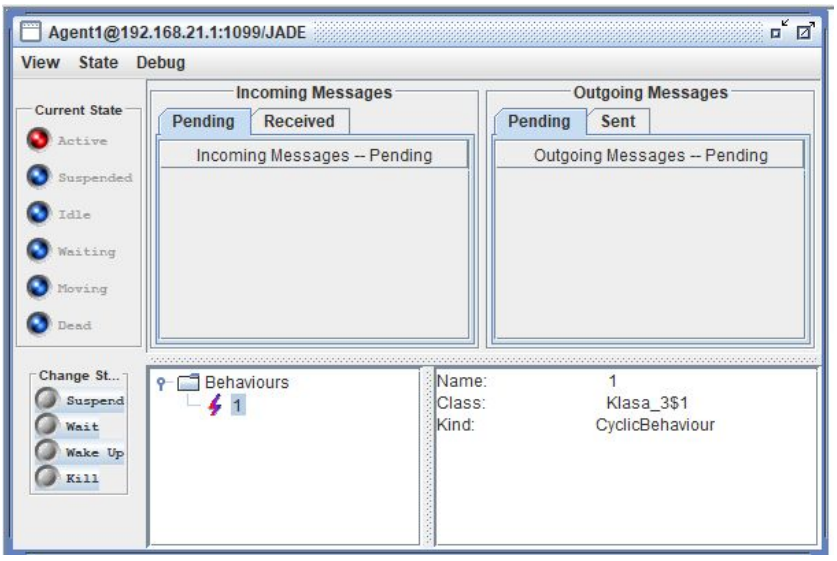


Następnie utworzyłam klasę agenta o nazwie *class_3*, na podstawie kodu *class_1*. Zmodyfikowałam ten fragment jednak, poprzez dodanie wielokrotnego zachowania wypisania na ekranie słowa wyrażenia *"In progress..."*. W tym celu skorzystałam z klasy *CyclicBehaviour*, gdzie przy każdym uruchomieniu wykonywane są te same operacje. Fragment kodu:

```
public class class_3 extends Agent{
    protected void setup() {
        System.out.println("Agent is starting");

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

Tym razem po uruchomieniu Introspectora można dostrzec, że agent wykonuje operacje w nieskończonej pętli.



Następnie utworzyłam klasę agenta o nazwie *class_4*, na podstawie kodu *class_1*. Zmodyfikowałam ten fragment jednak, poprzez dodanie zachowania generycznego, w którym w zależności od warunku możliwe jest wykonanie różnych operacji. W tym przypadku zadanie polegało na wykonaniu trzech kroków: wypisaniu konkretnego komunikatu oraz usunięciu zachowania z puli zachować agenta w trzecim kroku. Fragment kodu:

```java
public class class_4 extends Agent{
    protected void setup() {
        System.out.println("Agent is starting");
        addBehaviour(new ThreeStepsBehaviour());
    }
}

class ThreeStepsBehaviour extends Behaviour{
    private int step=0;
    public void action(){

        switch(step){
        case 0:
            System.out.println("First step");
            step++;
            break;

        case 1:
            System.out.println("Second step");
            step++;
            break;

        case 2:
            System.out.println("Third step");
            step++;
            break;
        }
    }
    public boolean done (){
        return step == 3;
    }
}
```

W wyniku czego otrzymujemy:

Następnie utworzyłam klasę agenta o nazwie *class_5*, na podstawie kodu *class_1*. Zmodyfikowałam ten fragment jednak, poprzez dodanie zachowania, które polega na pobraniu z klawiatury liczby całkowitej. W przypadku podania liczby ujemnej to zachowanie zostanie usunięte. Fragment kodu:

```
public class class_5 extends Agent{
    protected void setup() {
        System.out.println("Agent " +getLocalName()+ " starting");
        addBehaviour(new DeleteIfMinus());
    }

    class DeleteIfMinus extends Behaviour{
        Scanner input=new Scanner(System.in);
        int a;

        public void action(){
            a=input.nextInt();
            if(a<0){
                System.out.println("deleting");
            }
            if(a==0){
                System.out.println("Input equals to zero");
            }
            if(a>0){
                System.out.println("Input is positive");
            }
        }

        public boolean done (){
            return (a<0);
        }
    }
}
```

Następnie utworzyłam klasę agenta o nazwie *class_6,* na podstawie kodu *class_5*. Zmodyfikowałam ten fragment jednak, w taki sposób, aby zachowanie zawsze na początku wypisywało *"Behaviour is starting"*, a na końcu "Behaviour is finished". Fragment kodu:

```
public class class_6 extends Agent{
    protected void setup() {
        System.out.println("Agent " +getLocalName()+ " starting");
        addBehaviour(new DeleteIfMinus());
    }

    class DeleteIfMinus extends Behaviour{
        Scanner input=new Scanner(System.in);
        int a;

        public void action(){
            System.out.println("Behaviour is starting");
            a=input.nextInt();
            if(a<0){
                System.out.println("deleting");
            }
            if(a==0){
                System.out.println("Input equals to zero");
            }
            if(a>0){
                System.out.println("Input is positive");
            }
        }

        public boolean done (){
            if(a<0) {
                System.out.println("Behaviour is finished");
            }
            return (a<0);
        }
    }
}
```

Następnie utworzyłam klasę agenta o nazwie *class_7*, na podstawie kodu *class_4*. Zmodyfikowałam ten fragment jednak, dodając do istniejącego zachowania generycznego dwa kolejne. Pierwsze jest wykonywane na poziomie metody setup() i wypisuje *"First behaviour"*. Drugie jest dodane z poziomu zachowania generycznego. Jest dodane w pierwszym kroku i wypisuje *"Second behaviour"*. Fragment kodu:

```java
public class class_7 extends Agent{
    protected void setup() {
    System.out.println("Agent " +getLocalName()+ " starting");
    addBehaviour(new Behaviour() {
        boolean isFinished = false;
        public void action() {
            System.out.println("First behaviour");
            isFinished = true;
        }

        public boolean done() {
            return isFinished;
        }
    });

    addBehaviour(new ThreeStepsBehaviour(this));
    }
}

class ThreeStepsBehaviour extends Behaviour{
    private int step=0;
    Agent agent;

    public ThreeStepsBehaviour(Agent agent) {
        super(agent);
        agent = agent;
    }

    public void action(){
        switch(step){
        case 0:
            agent.addBehaviour(new Behaviour() {
                boolean isFinished = false;
                public void action() {
                    System.out.println("Second behaviour");
                    isFinished = true;
                }

                public boolean done() {
                    return isFinished;
                }
            });
            System.out.println("First step");
            step++;
            break;

        case 1:
            System.out.println("Second step");
            step++;
            break;

        case 2:
            System.out.println("Third step");
            step++;
            break;
        }
    }
    public boolean done (){
        return step == 3;
    }
```

Następnie utworzyłam klasę agenta o nazwie *class_8,* na podstawie kodu *class_1*. Zmodyfikowałam ten fragment jednak, dodając zachowania, które spowodują kolejno wypisanie *"A small tick"* co dwie sekundy oraz *"A big tick"* co pięć sekund. Oprócz tego po 50 sekundach zostanie usunięte zachowanie odpowiedzialne za Big tick, natomiast po 100 sekundach zostanie usunięty cały agent. W tym celu skorzystałam z zachowania TickerBehaviour oraz utworzyłam własne zachowanie. Oprócz tego skorzystałam z metody removeBehaviour. Fragment kodu:

```
public class class_8 extends Agent{
    //protected static TickerBehaviour tickerBehaviour;

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

        addBehaviour(new TickerBehaviour(this, 2000) {
            protected void onTick() {
                System.out.println("a small tick");
            }
        });

        BigTick bigTick = new BigTick(this,5000);
        addBehaviour(bigTick);

        addBehaviour(new TickerBehaviour(this, 50000) {
            protected void onTick() {
                System.out.println("Removing big tick");
                removeBehaviour(bigTick);
            }
        });

        addBehaviour(new WakerBehaviour(this, 100000) {
            protected void handleElapsedTimeout() {
                System.out.println("deleting Agent");
                myAgent.doDelete();
            }
        });
    }
}

class BigTick extends TickerBehaviour{
    public BigTick(Agent a, long p) {
        super(a, p);
    }

    protected void onTick() {
        System.out.println("A big tick");
    }
}
```

Podczas analizy Introspectorem można dostrzec, że początkowo dostępne były cztery zachowania.

Po upływie 50 sekund zostało usunięte zachowanie Big tick, w wyniku czego pozostały trzy zachowania.

Następnie, po 100 sekundach usunął się cały agent, w wyniku czego, w Intospectorze nie ma już nic.