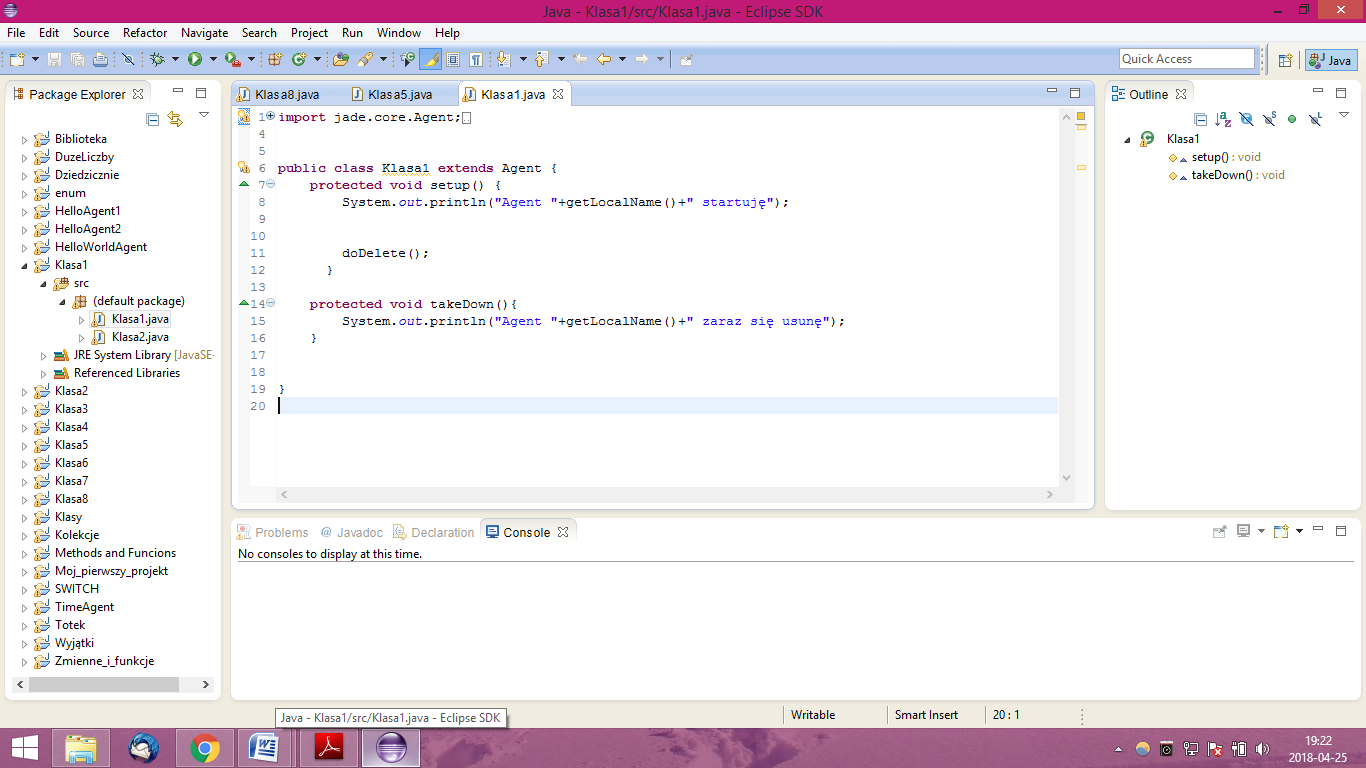
Marcelina Bałamut,

Inżynieria Obliczeniowa, gr1

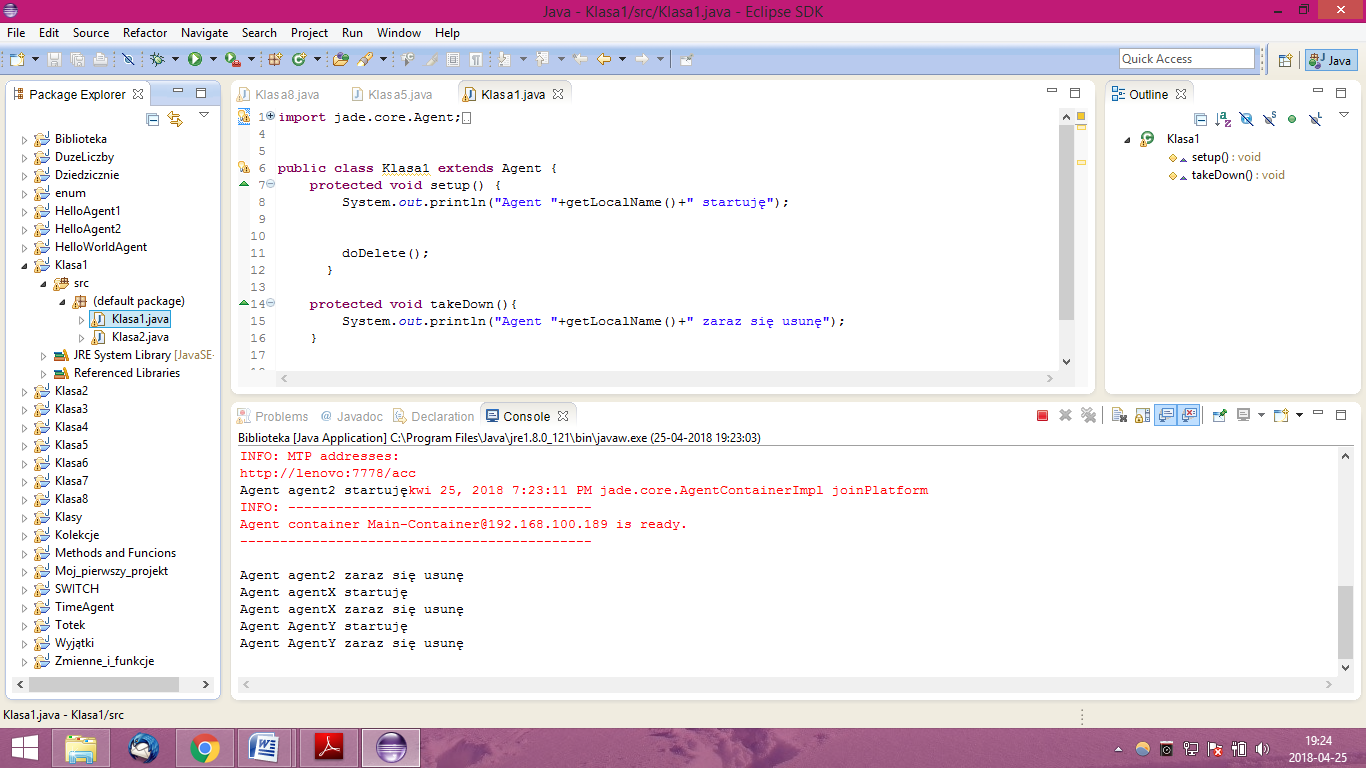
***Sprawozdanie z zagadnienia nr 7***

***Rozproszona sztuczna inteligencja***

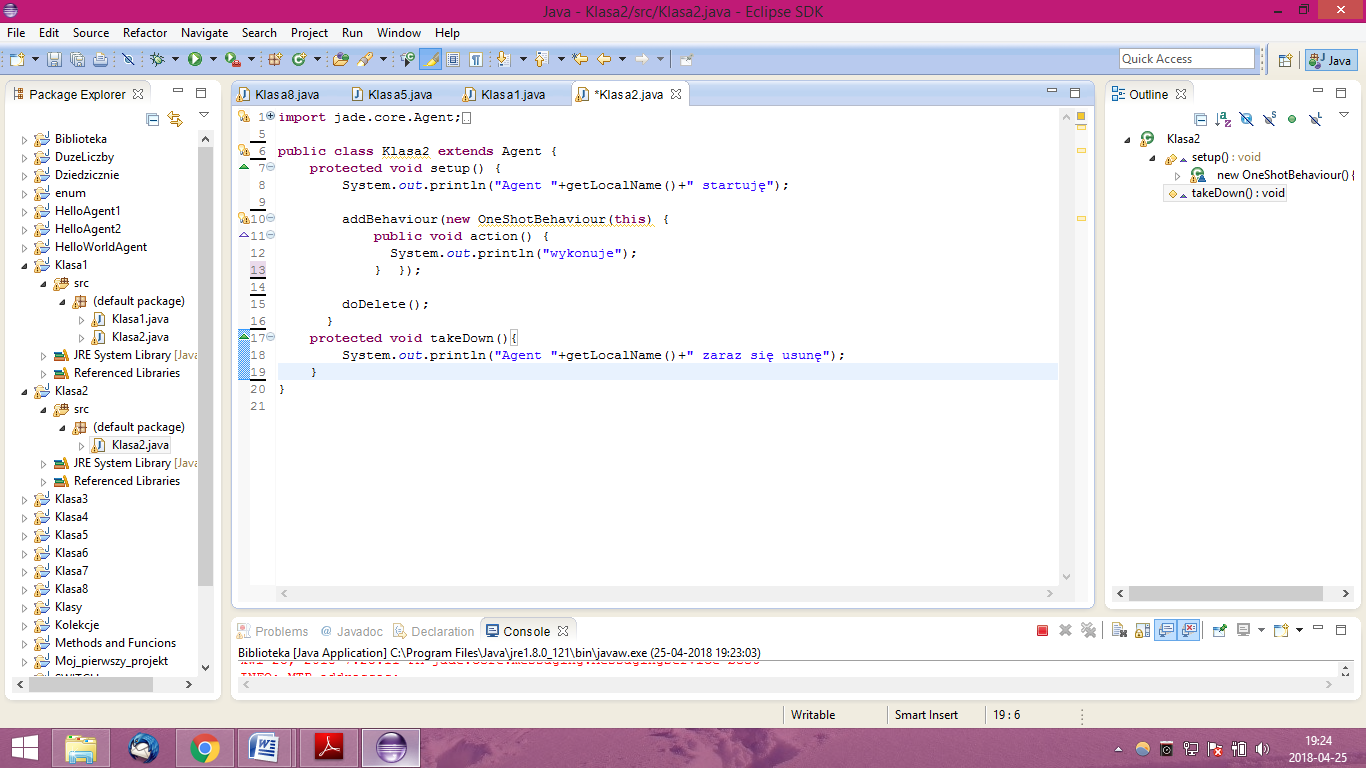
Pierwszym krokiem do wykonania zadania było utworzenie Klasa1, gdzie Agent zawsze na samym początku wypisuje tekst „startuję”, a przed swoim usunięciem wypisuje: „zaraz się usunę”.



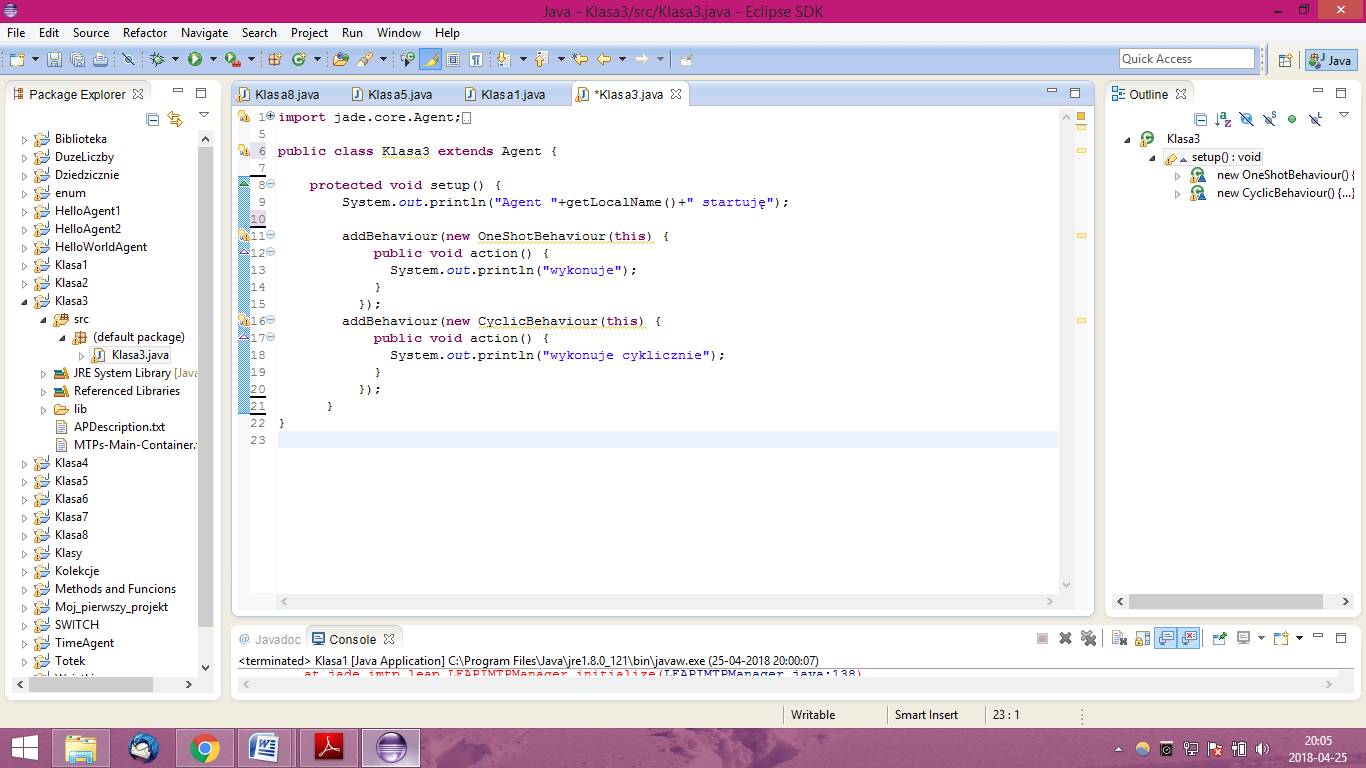
Wynik programu:



W kolejnym kroku utworzyłam klasę Klasa2 na podstawie kodu Klasa1. Dodałam do agenta zachowanie które jednokrotnie wykonuje operację wypisywania na ekran słowa” wykonuję”



Następnie utworzyłam klasę o nazwie Klasa3 na podstawie Klasa1. Dodałam do agenta zachowanie które wielokrotnie wypisywało na ekran słowa „wykonuję”.



Działanie to możliwe było przy użyciu zachowania CyclicBehaviour, i metody action(), która wypisywała na konsole pożądane wyjścia.

Nastepnie utworzyłam klasę agenta o nazwie Klasa4 na podstawie Klasa1 i dodałam do agenta zachowanie „generyczne” polegające na tym, że pierwszym kroku program wypisuje” pierwszy krok”, w drugim: „drugi krok” a w trzecim „ trzeci krok” i zachowanie zostaje usunięte z puli zachowań:

**public** **class** Klasa4 **extends** Agent {

**protected** **void** setup() {

System.*out*.println("Agent "+getLocalName()+" startuję");

addBehaviour(**new** TreeStepBehaviour());

}

**class** TreeStepBehaviour **extends** Behaviour{

**private** **int** step=0;

**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");

removeBehaviour(**this**);

step++;

**break**;

}

}

**public** **boolean** done (){

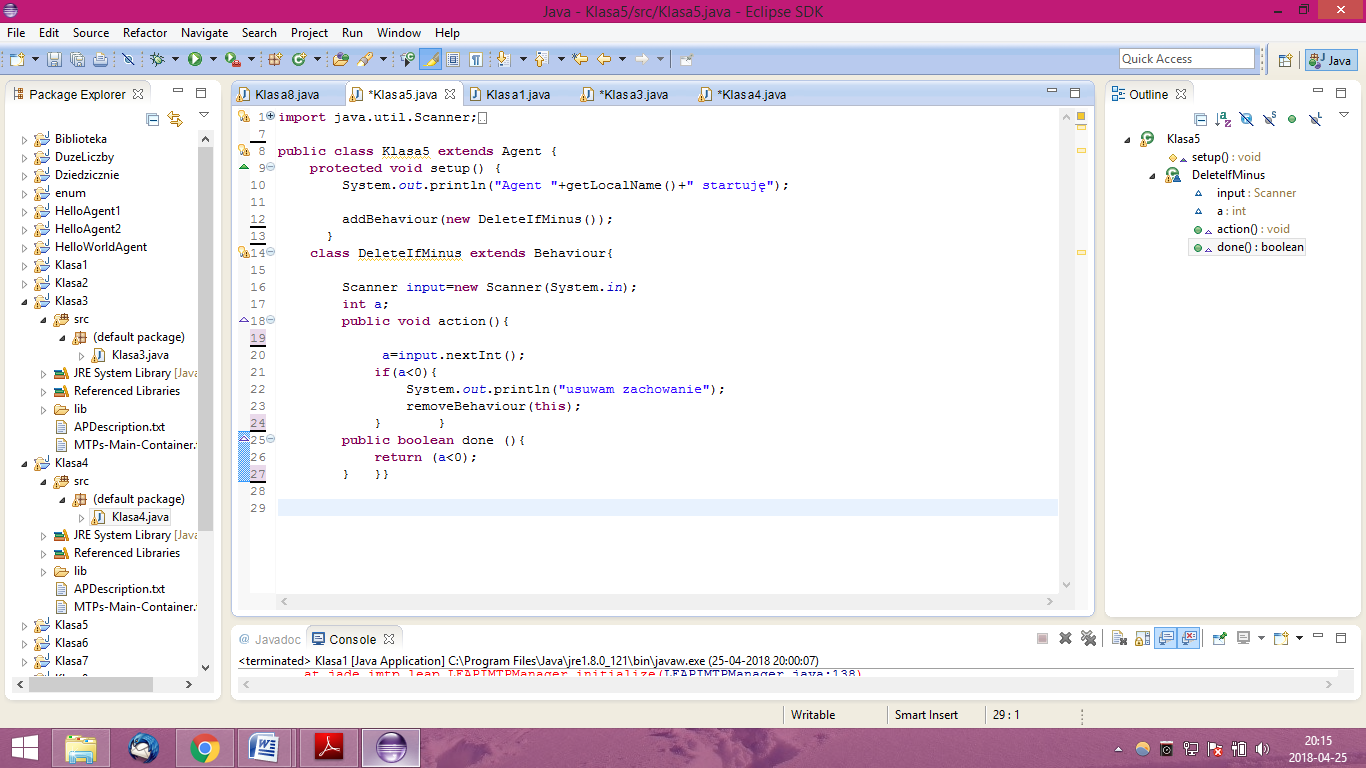
**return** step ==3;

}}

}

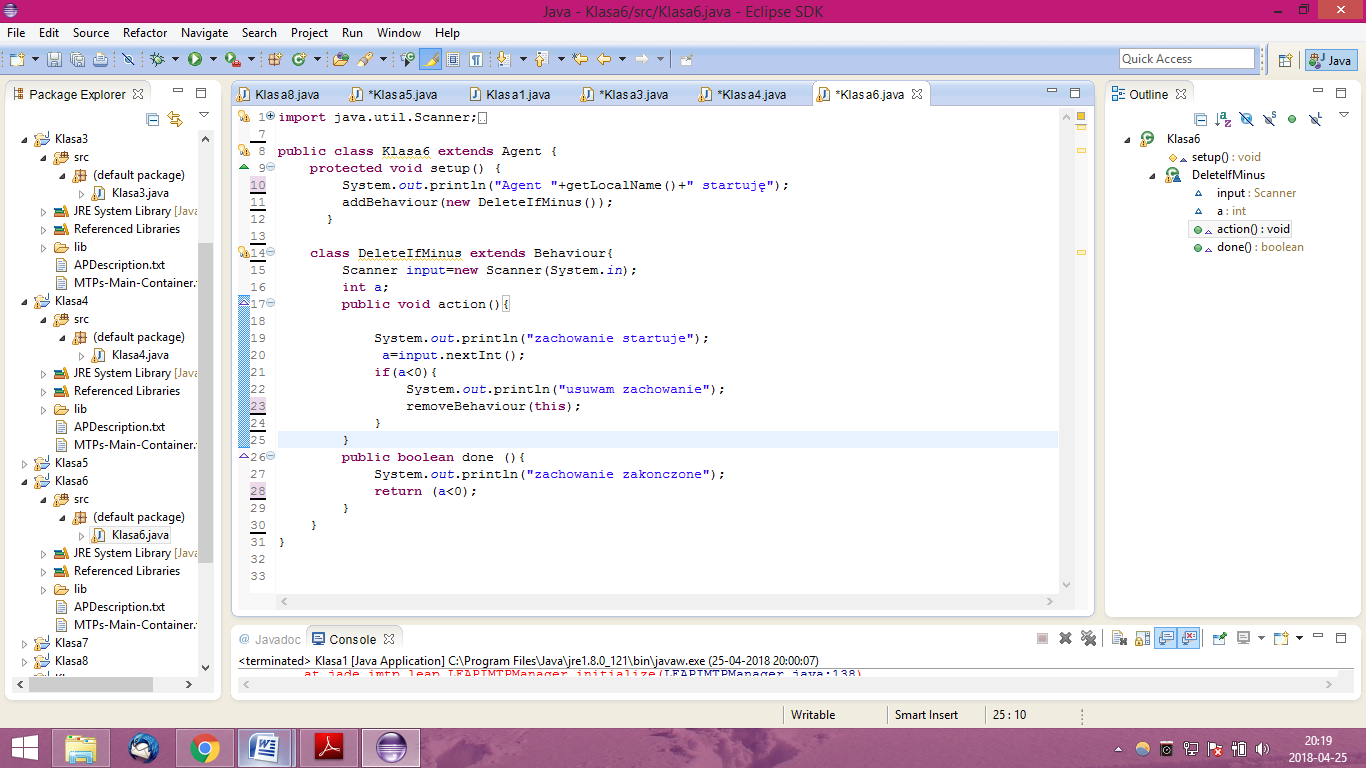
Wykorzystano tutaj mechanizm switch który za pomocą zmiennej wypisuje na ekran odpowiedni komunikat. Mechanizm ten został zaimplementowany w nowo stworzonym przeze mnie zachowaniu dziedziczącym po Behaviour, a następnie zachowanie zostało dodane do agenta. Została zaimplementowana również funkcja done() w której zawiera się warunek kończący switch.

Kolejno utworzyłam klasę Klasa5 na podstawie Klasa1 i dodałam do agenta zachowanie o nazwie polegające na pobieraniu z klawiatury liczby całkowitej, a jeśli użytkownik poda liczbę ujemną to usuwa zachowanie.

.

Zachowanie zawiera prosty warunek, który jest zaimplementowany w metodzie done(), jeśli użytkownik poda liczbę ujemną zostanie wywołana metoda removeBehaviour() usuwająca zachowanie.

Kolejno utworzyłam klasę o nazwie Klasa6 na podstawie Klasa5 i zmodyfikowałam kod tak, aby zawsze zachowanie na początku wypisywało „zachowanie startuje”, a na samym końcu wypisywało „zachowanie zakończone”.



Następnie utworzyłam klasę agenta Klasa7 na podstawie kody Klasa4, gdzie do istniejącego zachowania dodałam dwa kolejne:

- dodawane na poziomie metody setup() i ma polega na jednokrotnym wypisaniu „pierwsze”

- dodane z poziomu zachowania „generycznego” – dokładnie dodane w pierwszym kroku i jednokrotnie wypisuje „drugie”.

**public** **class** Klasa7 **extends** Agent {

**protected** **void** setup() {

System.*out*.println("Agent "+getLocalName()+" startuję");

addBehaviour(**new** First());

addBehaviour(**new** TreeStepBehaviour());

}

**class** TreeStepBehaviour **extends** Behaviour{

**private** **int** step=0;

**public** **void** action(){

**switch**(step){

**case** 0:

System.*out*.println("Krok pierwszy ");

addBehaviour(**new** Second());

step++;

**break**;

**case** 1:

System.*out*.println("Krok drugi ");

step++;

**break**;

**case** 2:

System.*out*.println("Krok trzeci");

removeBehaviour(**this**);

step++;

**break**;

}

}

**public** **boolean** done (){

**return** step ==3;

}

}

**class** First **extends** OneShotBehaviour{

**public** **void** action(){

System.*out*.println("pierwsze");

}}

**class** Second **extends** OneShotBehaviour{

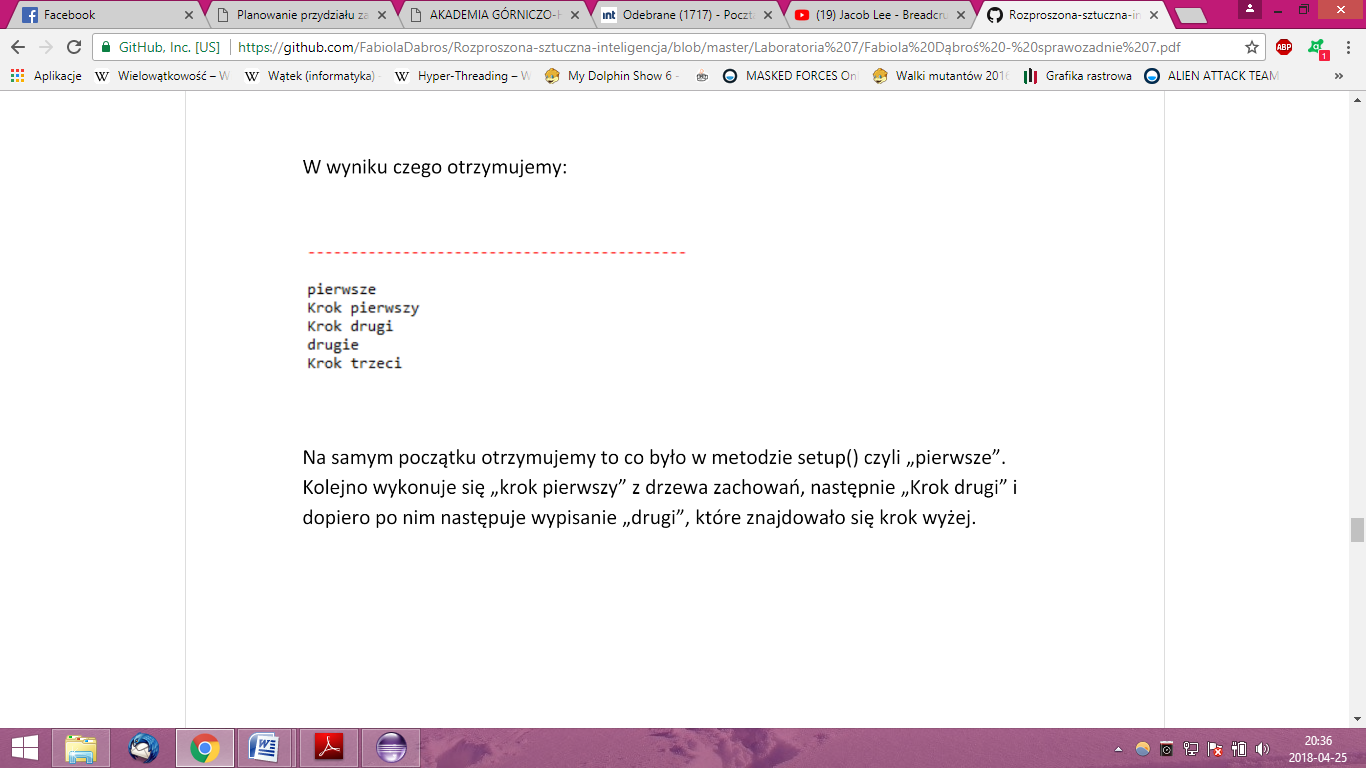
**public** **void** action(){

System.*out*.println("drugie");

}}

}

Kolejka działania programu:



Pierwsze wykonują się operacje zawarte w metodzie setup(). Następnie wykonują się operacje zawarte w mechanizmie switch, jednak zachowanie zagnieżdżone w nim wykona się później niż podstawowa operacja.

Ostatnim krokiem było utworzenie klasy Klasa8 na podstawie Klasa1. Do agenta dodałam następujące zachowania:

- Wypisanie „mały tick” co 2 sekundy,

- Wypisaniu „duży tick” co 5 sekund,

- Po 50 sekundach usunięcie zachowania z punktu b,

- Po 100 sekundach usunięcie całego agenta

**public** **class** Klasa8 **extends** Agent {

**protected** **static** TickerBehaviour *a*;

**protected** **void** setup() {

System.*out*.println("Agent "+getLocalName()+" startuję");

addBehaviour(**new** TickerBehaviour(**this**, 2000) {

**protected** **void** onTick() {

System.*out*.println("mały tick");

}

});

addBehaviour(*a*=**new** TickerBehaviour(**this**, 5000) {

**protected** **void** onTick() {

System.*out*.println("duży tick");

}

});

addBehaviour(**new** WakerBehaviour(**this**, 50000) {

**protected** **void** handleElapsedTimeout() {

System.*out*.println(" usuwanie dużego ticka");

removeBehaviour(*a*);

}

});

addBehaviour(**new** WakerBehaviour(**this**, 100000) {

**protected** **void** handleElapsedTimeout() {

System.*out*.println(" usuwanie agenta");

myAgent.doDelete();

}

});

}

}

Do realizacji tego dodałam dwa zachowania w WakerBehaviour w których wywołałam odpowiednie metody usuwające. Pierwsza z nich to removeBehaviour, która usuwa zachowanie. Aby móc przesłać jej zachowanie jako argument stworzyłam statyczną referencję. Druga metoda to doDelete() usuwająca agenta wywołana na rzecz tego właśnie agenta. Wcześniej dodałam zachowania TickerBehaviour które po zadanym czasie wykonywały wypisywania komunikatów w funkcjach onTick().