1. Funkcja statyczna - po co i jak
2. Metoda finalize() oraz final
3. funkcja onEnd() - kiedy używamy i w ktorej klasie
4. interakca agenta z agentem AMS - migracje, klonowanie, rejestracja/wyrejestroweanie do/z white pages tworzenie,kilowanie, spr poprawnosci nazw
5. Kompilacja i uruchamianie agentow
6. Jak pobrac argumenty w agencie
7. Zarejestruj cos w yellow pages
8. Wyrejestrowanie sie z yellow pages, gdzie sie to robi i jak
9. Zaimplementuj agenta, ktory bedzie co 2s wypisywal „dziala”
10. Serializacja - Example:
11. Napisz kod agenta, ktory wysle wiadomosc do innego agenta
12. AMS
13. Napisz kod konwertujacy string do int
14. Napisz kod agenta ktory 20 s od uruchomienia wypisze „to ja”
15. Pakiety
16. Funkcje klasy object
17. Inspector agent.
18. Jak klonować i migrować agenta?
19. CompositeBehaviour
20. Czy moze agent zamknac platfrome, tak czy nie i dlaczego
21. Podstawowe rodzaje zachowań
22. Jak uruchomić platforme JADE z GUI
23. Wymień możliwe tzw. performative wiadomości.
24. Dummy agent.
25. Sniffer agent.
26. Metody onStart(), onEnd().
27. Cykl życia agenta
28. Przyjmij i wypisz wiadomość
29. Wypisz wszystkich agentów którzy zarejestrowali w YP podana usługę

1. Funkcja statyczna - po co i jak

tworzymy ją po to aby móc wywoływać daną metode bez tworzenia obiektu klasy z której pochodzi

2. Metoda finalize() oraz final

metoda finalize() - uruchamia się zaraz przed tym jak GC collector chce usunąć obiekt (ginie ostatnia referencja do danego obiektu). Domyślnie nic nie robi, możemy ją sami zaimplementować.
final class - nie można z niej dziedziczyć
final metoda - nie można przeciążać w klasie potomnej
final obiekt (zmienna,pole) - po inicjalizacji nie może zmieniać wartości

3. funkcja onEnd() - kiedy używamy i w ktorej klasie

onEnd() znajduje sie w klasie Behavoiur
Nadpisujemy ją gdy chcemy coś zrobić gdy zachowanie zostanie zakończonie.

4. interakca agenta z agentem AMS - migracje, klonowanie, rejestracja/wyrejestroweanie do/z white pages tworzenie,kilowanie, spr poprawnosci nazw

Zabijanie danego agenta

```
KillAgent kill = new KillAgent();
kill.setAgent(new AID("john", AID.ISLOCALNAME));
Action actExpr = new Action(getAMS(), kill);
```

Migracja agenta do losowego kontenera:

```
Agent agent = this.getAgent();
int index = randomizer.nextInt( containers.size() );
ContainerID container = (ContainerID) containers.get( index );
agent.doSuspend();
agent.doMove( container );
agent.doActivate();
```

5. Kompilacja i uruchamianie agentow

java jade.Boot -gui (uruchamiamy gui musi byc uruchomione przed uruchomieniem agenta, da sie uruchomic gui i agenta z jedneo polecenia)
java jade.Boot -container -container-name nazwakontenera nazwa:program (tworzy kontener o danej nazwie a w nim agenta o danej nazwie)
java jade.Boot -container nazwa:program (tworzy kontener o domyslnej nazwie a w nim agenta o danej nazwie)
java jade.Boot -agents nazwa:program (tworzy agenta w glownym kontenerze) (z terminala nie zadzialalo)

6. Jak pobrac argumenty w agencie

java jade.Boot foo:FooAgent(1,arg2,argument3) (nie działa w konsoli)

```
public void setup() {
    Object[] args = getArguments();
    String arg1 = args[0].toString(); // this returns the String "1"
    String arg2 = args[1].toString(); // this returns the String "arg2"
    String arg3 = args[2].toString(); // this returns the String "argument3"
}
```

7. Zarejestruj cos w yellow pages

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd = new ServiceDescription();
sd.setName("Nazwa_serwisu");
sd.setType("Typ_serwisu");
dfd.addServices(sd);
try{
        DFService.register(myAgent,dfd);
} catch (FIPAException e) {
        e.printStackTrace();
}
```

8. Wyrejestrowanie sie z yellow pages, gdzie sie to robi i jak

Dobrą praktyką jest wyrejestrowywanie się z Yellow Pages gdy agent ginie (patrz cykl życia agenta).

```
protected void takeDown() {
```

```
        super.takeDown();
        try {
                DFService.deregister(this);
        } catch (FIPAException e) {
                e.printStackTrace();
        }
}
```

9. Zaimplementuj agenta, ktory bedzie co 2s wypisywal „dziala”

```
1.public class NewAgent extends Agent {
2.protected void setup() {
3.addBehaviour(new TickerBehaviour(this,2000){
4.protected void onTick() {
5.System.out.println("Działa");
6.}
7.});
8.}
9.}
```

10. Serializacja - Example:

Klasa zawierająca przykład użycia - Student implementuje klase Serializable

```
1.import java.io.Serializable;
2.public class Student implements Serializable{
3.private String imie;
4.void setImie(String imie) {
5.     this.imie = imie;
6.}
7.String getImie() {
8.       return imie;
9.}
10.Student(){}
11.Student(String name)
12.{
13.   imie = name;
14.}
15.}
```

Klasa main - zapis obiektu serializowanego i odczyt z pliku - pobranie imienia

```
1.public class SerializationExample {
2.public static void main(String [] args) throws FileNotFoundException, IOException, ClassN
otFoundException
3.{
4.//tworzymy nowego studenta
5.Student john = new Student("Johnie");
6.// tworzymy obiekt klasy ObjectOutputStream do zapisywania do pliku
7.ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("file.dat"));
8.out.writeObject(john);
9.// tworzymy obiekt klasy ObjectInputStream do odczytywania z pliku
10.ObjectInputStream in = new ObjectInputStream(new FileInputStream("file.dat"));
11. //rzutujemy wejscie na dany obiekt
12.Student kto = (Student)in.readObject();
13.// i ładnie na konsolę wyrzucamy wynik
14.System.out.println(kto.getImie());
15.}
16.}
```

11. Napisz kod agenta, ktory wysle wiadomosc do innego agenta

```
public class NewAgent extends Agent {
        protected void setup() {
addBehaviour(new OneShotBehaviour() {
         public void action() {
ACLMessage msg
= new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("agent008", AID.ISLOCALNAME));
msg.setContent("Wiadomosc testowa");
send(msg);
}
});
}
```

```
}
```

12. AMS

agent specjalny, który ma moliwości dodawania i usuwanie agentów oraz kontenerów, zamykanie platformy (agent również może rządać zamknięcia bierzącego agenta).

13. Napisz kod konwertujacy string do int

```
String to Int : Integer.parseInt(String s)
Int to String :  int a = 2;
String aa = Integer.toString(a);
```

14. Napisz kod agenta ktory 20 s od uruchomienia wypisze „to ja”

```
public class NewAgent extends Agent {
protected void setup() {
addBehaviour(new WakerBehaviour(this,2000) {
protected void handleElapsedTimedout() {
System.out.println("to ja");
}
});
}
}
```

15. Pakiety

-Pakiety to nic innego niż foldery w których zawarte są klasy, więc po utworzeniu folderu np: `com/mycompany/myproject` (?)
-Pakiety umożliwiają np. oddzielenie naszego kodu od kodu bibliotek napisanych przez innych programistów.
-Wraz z JDK otrzymujemy dość pokaźną liczbę pakietów, zawierających w sumie kilka tysięcy klas.
-Głównym powodem stosowania pakietów jest gwarancja unikalności nazw klas.
-Na samym początku pliku źródłowego Javy może znajdować się instrukcja **package**, informująca kompilator o nazwie pakietu, do którego mają należeć wszystkie zdefiniowane klasy.
-Pominięcie instrukcji **package** powoduje, że klasy z danego pliku są umieszczane w pakiecie domyślnym, nie posiadającym nazwy.
-Użyta instrukcja **package** musi stanowić w pliku pierwszy element nie będący komentarzem

16. Funkcje klasy object

String toString()
Object clone()
boolean equals(Object obj),
int hashCode()
void finalize()
getClass()
notify(), notifyAll()
wait(), wait(long)

17. Inspector agent.

IntrospectorAgent pozwala na wybranie agenta i monitorowanie jego zachowań oraz wiadomości jakie do niego przychodzą i są przez niego wysyłane, możliwy jest zarówno podgląd kolejki wiadomości agenta (oczekujących na odebranie) jak i tych odebranych.

18. Jak klonować i migrować agenta?

Metody w agencie: doClone(Location arg0, String arg1) i doMove(Location arg0).

19. CompositeBehaviour

Ta klasa skomponowana jest z innych zachowań (dzieci). Operacje wykonywane są więc zdefiniowane w „dzieciach” natomiast sama klasa zajmuje się harmonogramowaniem tych operacji według określonych reguł. Sama klasa nie definiuje tych zasad a daje tylko interfejs. Zasady muszą być zdefiniowane w podklasach (SequentialBehaviour, ParallelBehaviour, FSMBehaviour).

20. Czy moze agent zamknac platfrome, tak czy nie i dlaczego

nie moze, moze to zrobic AMS, agent co najwyzej moze zarzadac od AMS zeby zamknal platforme

21. Podstawowe rodzaje zachowań

OneShotBehaviour – wykonuje akcję raz
CyclickBehaviour – wykonuje akcję cały czas (“w kółko”)
WakerBehaviour – wykonuje akcję po upływie podanego czasu
TickerBehaviour – wykonuje akcję „w kółko” z odstępami czasowymi

22. Jak uruchomić platforme JADE z GUI

Java jade.Boot –gui

23. Wymień możliwe tzw. performative wiadomości.

INFORM, REQUEST, AGREE, CANCEL, CONFIRM, DISCONFIM, FAILURE, UNKNOW, NOT_UNDERSTOOD, SUBSCRIBE itd.

24. Dummy agent.

Jest narzędziem do monitorowania i debugowania. Tworzy graficzny interfejs. Używając GUI możemy tworzyć wiadomości ACL i wysyłać je do innych agentów. Można wyświetlić wszystkie wiadomości wysłane i otrzymane.

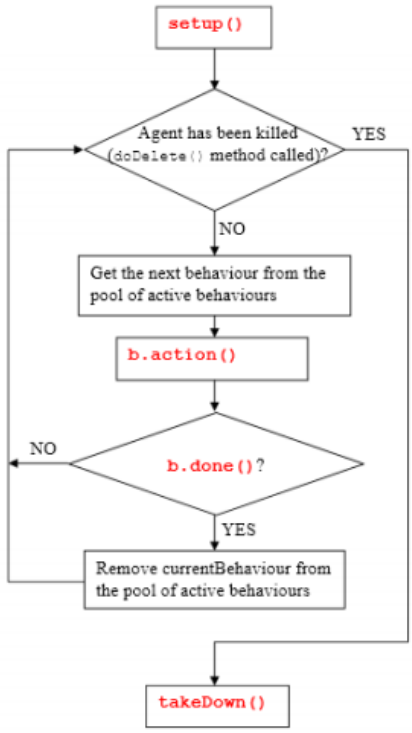
25. Sniffer agent.

Sniffer pokazuje przepływ wiadomości między agentami w perspektywie wielu agentów, które zostały wybrane jako przedmioty monitoringu. Można powiedzieć, że łapie on wiadomości w locie i pokazuje nam ich użycie. Tworzy diagramy zbliżone do UML.

26. Metody onStart(), onEnd().

Znajdują się w zachowaniu i są jego „prologiem” i „epilogiem”. Są puste i można je nadpisać w klasach potomnych. Przed wykonaniem akcji z zachowania przez agenta zostanie wykonana metoda onStart() (jeden raz nawet dla CyclicBehaviour). Analogicznie jest na onEnd() tylko, że wykona się na koniec wykonywanej akcji. onEnd zwrana int’a – kod zakończenia zachowania.

27. Cykl życia agenta



28. Przyjmij i wypisz wiadomość

```
addBehaviour(new CyclicBehaviour() {
public void action(){
ACLMessage msg = receive();
if(msg!=null){
System.out.println(msg.getContent());
}}
});
```

29. Wypisz wszystkich agentów którzy zarejestrowali w YP podana usługę

```
DFAgentDescription dfd = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType("Typ_serwisu");
dfd.addServices(sd);
try{
DFAgentDescription results[] = DFService.search(myAgent,dfd);
for(int i=0;i<results.lenght;i++){
System.out.println(results[i].getName().getLocalName());
}
} catch (FIPAException e) {
e.printStackTrace();
}
```