Създаване и управление на поведенията на агентите.

Аделина Алексиева-Петрова
aaleksieva@tu-sofia.bg

Поведение на агент

- Действителната работа, която агентите трябва да извършат се прави от така нареченото поведение (*Behavior*), което се разбира като начин на работа.
- Поведението на агента представлява задача, която той може да осъществи и класът, който описва това поведение е в пакета *jade.core.behaviours.Behaviour*.

JADE

- паралелен модел нишка-за-агент (thread-per-agent);
- всички поведения на даден агент са съхраняват в опашка;
- диспечер с кръгово движение на задачите (round robin scheduler).

Предимства – 1/2

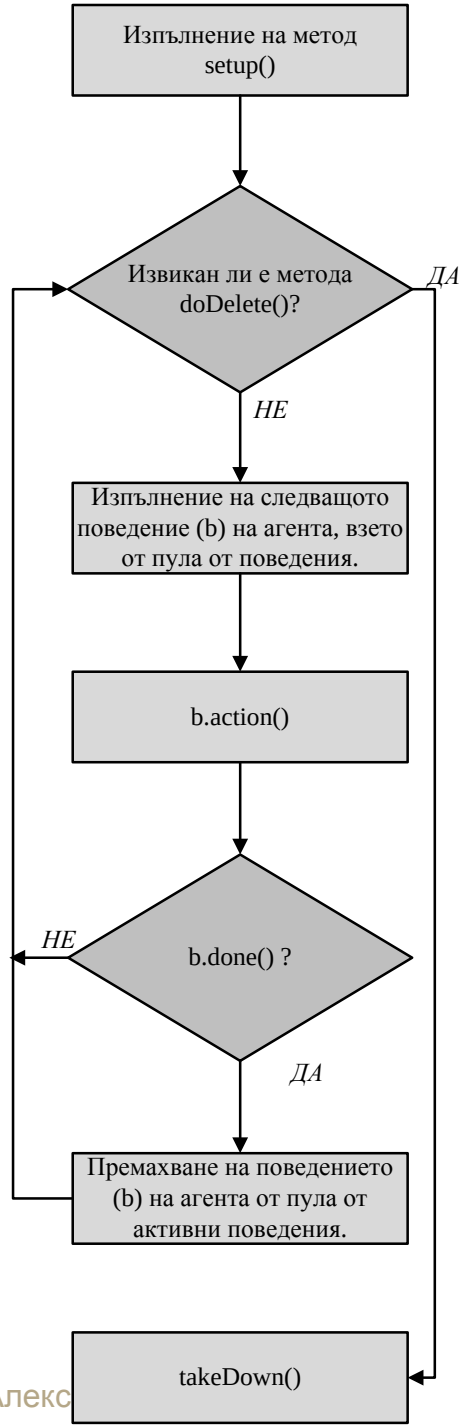
- Работата на агентите може да се осъществява от единична JAVA нишка.
- Подобрява производителността, тъй като превключването между отделните поведения е по-бързо, отколкото превключването между нишките на JAVA.

Предимства – 2/2

- Елиминират се всички проблеми, свързани със синхронизирането между конкурентни поведения, състезаващи се за едни и същи ресурси, тъй като всички те се изпълняват от една JAVA нишка.
- При преминаване от едно поведение към друго, агента не включва никаква информация, като запомня моментното състояние на агента.

Блок-схема на изпълнение на агента

- метода *action*() - определя какво трябва да бъде направено, когато се стартира самото поведение.
- метода *done*() - връща логически резултат дали поведението е приключило или не.



Реализиране на агент с поведение

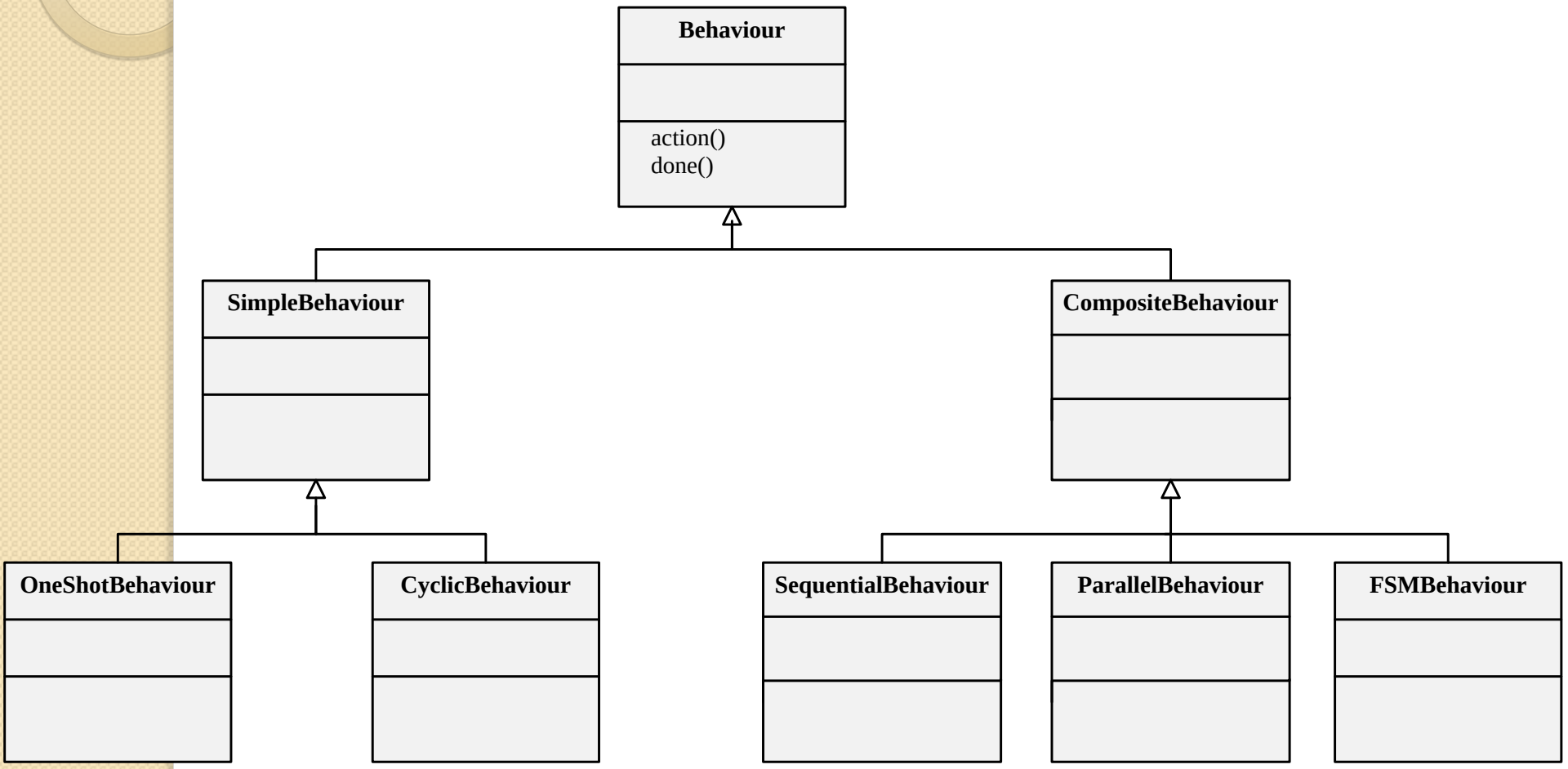
```
import jade.core.Agent;
import jade.core.behaviours.OneShotBehaviour;
public class HelloWorldAgent extends Agent {
    protected void setup() {
            addBehaviour(new HelloWorldBehaviour());
  }
}

class HelloWorldBehaviour extends OneShotBehaviour{
    public void action(){
            System.out.println("Hello World! My name is
                        "+myAgent.getLocalName());
  }
}
```

Реализиране на агент с поведение

- Вместо да се изпълнява агента чрез метода *setup()*, е възможно да се опише поведението на агента.
- Клас в JADE jade.core.behaviours.Behaviour:

```
import jade.core.Agent;
import jade.core.behaviours.OneShotBehaviour;
public class HelloWorldAgent extends Agent {
    protected void setup() {
      addBehaviour(new OneShotBehaviour() {
            public void action(){
              System.out.println("Hello World! My name
                                  is "+getLocalName());
            }
    });
  }
}
```

UML клас диаграма на основните поведения в JADE



Класове за управление на поведението на агент – 1/2

- OneShotBehaviour – е проектиран така, че неговия метод action() се изпълнява един път и приключва. Класът на този метод имплементира в себе си метода done(), който връща стойност true.
- CyclicBehaviour – CyclicBehaviour е направен така, че поведението на агента никога да не приключва или това е версията, при която променливата finished е със стойност false. Техният метод action() изпълнява една и съща операция всеки път, когато е извикан. Този клас също е имплементирал метода done(), тук неговата стойност е false.

Класове за управление на поведението на агент – 1/2

- *CompositeBehaviour* описва множество от под-поведения за даден агент, като дъщерни поведения.
  - Класът *SequentialBehaviour* описва комплексно поведение, чиито под-поведения се изпълняват последователно.
  - Класът *ParallelBehaviour* описва комплексно поведение, чиито под-поведения се изпълняват паралелно и приключва, когато всички дъщерни поведения приключат.
  - Класът *FSMBehaviour* използва механизъма на крайните автомати за политика за изпълнение на под-поведенията, като под-поведенията са състоянията и текущото състояние е активното поведение.

Методи на клас Behaviour

- onStart() – инициализиращи действия, извиква се преди да се извика метода *action() за първи път;*
- onEnd() – извиква се преди да се премахне поведението;
- block() – блокира поведението докато не се появи ново съобщение;
- block (long millis) – блокира поведението докато не се появи ново съобщение или докато не изтече зададеното време;
- isRunnable() – връща стойност false, ако поведението е блокирано.

Пример

```
SequentialBehaviour seq2 = new SequentialBehaviour(this);
     seq2.setBehaviourName("SecondSequentialBehaviour");

     seq2.addSubBehaviour(new MyBehaviour(this));
     seq2.addSubBehaviour(new MyBehaviour(this));
     seq2.addSubBehaviour(new MyBehaviour(this));

ParallelBehaviour p = new ParallelBehaviour(this,
                           ParallelBehaviour.WHEN_ALL);
     p.addSubBehaviour(seq1);
     p.addSubBehaviour(seq2);

     addBehaviour(p);
```