Correction du TP N°5 : Implémentation du 2ème algorithme distribué en utilisant la plateforme JADE (3ème variante)

Dans cette implémentation :

1. **Le consultation de la boite aux lettes va se faire dans un comportement qui va s'exécuter en parallèle avec les autres comportements.**
2. **Les comportements 'DevenirDemandeur' et 'AttendreJetonPresent' sont regroupé dans un seul comportement nommé 'Aquerir'.**

Remarque : Nous pouvons aussi regrouper les comportement 'Liberer' et 'Dehors' dans un seul comportement mais nous devons intégrer les instructions du comportement 'Liberer' dans le test : if (etat.equals("dedans")).

Classe site :

```
public class site extends Agent{
    String NomSuivant;
    String etat = "dehors" ;
    int jetonPrésent;

    public void setup(){
        System.out.println("Agent "+ this.getLocalName());
        Object [] args = this.getArguments();
        if (args!= null){
            NomSuivant = args[0].toString();
            jetonPrésent = Integer.parseInt(args[1].toString());
            if (jetonPrésent == 1){//le site qui possède le jeton doit l'envoyer à son successeur car il est dans l'état dehors
                ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                msg.addReceiver(new AID(NomSuivant,AID.ISLOCALNAME));
                msg.setContent("Jeton");
                send(msg);
            }
            System.out.println("Agent "+ this.getLocalName()+ " NomSuivant :" + NomSuivant+ " jetonPrésent "+jetonPrésent );
        }
        addBehaviour(new Dehors());//rajouter le 1er comportement
        addBehaviour(new ConsulterBoite());//rajouter ce comportement pour consulter la boite aux lettres //1ère façon de le faire

        /*ParallelBehaviour ComportParallel = new ParallelBehaviour(ParallelBehaviour.WHEN_ALL); //2ème façon de le faire
        ComportParallel.addSubBehaviour(new Dehors());  //Ajouter le 1er sous-comportement
        ComportParallel.addSubBehaviour(new ConsulterBoite());  //rajouter ce comportement pour consulter la boite aux lettres
        addBehaviour(ComportParallel);  //Ajouter le comportement séquetiel à l'agent*/
    }


  public class Dehors extends Behaviour{
      public void action() {
          etat = "dehors";
          System.out.println("Agent "+ getLocalName()+ " je suis dans l'état dehors ");
          block((int) (Math.random() * 10000));
      }
      public boolean done() {
          int val = (int) (Math.random() * 2);//generer une variable aléatoire qui prend soit la valeur 0 soit la valeur 1
          if (val == 0) return false; //je reste dans l'état dehors
          else{//je passe vers l'état demandeur
              addBehaviour(new Aquerir());//on rajoute ce comportement pour devenir demandeur et attendre que la variable jetonPrésent==1
              return true;
          }
      }
  }
```

```
public class Aquerir extends Behaviour{
    public void action() {
        if (etat.equals("dehors")){ // on a rajouté ce test car ces instructions doivent s'exécuter 1 seule fois
            etat = "demandeur";
            System.out.println("Agent "+ getLocalName()+ " je suis dans l'état demandeur ");

        }
        //System.out.println("Agent "+ getLocalName()+ " j'attend le jeton");
        block((int) (Math.random() * 10000));
        if (jetonPrésent == 1) { // attendre que la variable jetonPrésent == 1
            etat = "dedans";
            addBehaviour(new EnSC()); // on rajoute ce comportement pour entrer en Section Critique
        }
    }
    public boolean done() {
        if (jetonPrésent == 1)
            return true; //j'ai le jeton --> j'accède à la SC
        else
            return false; //je n'ai pas le jeton je refais l'exécution de la méthode action
    }
}


public class EnSC extends OneShotBehaviour{
    public void action() {
        for (int i = 0; i < 5 ; i++)
            System.out.println("********* Agent "+getLocalName()+" Je suis en SC");
        block((int) (Math.random() * 10000));
        addBehaviour(new Liberer()); //on rajoute ce comportement pour libérer la Section Critique
    }
}
public class Liberer extends OneShotBehaviour{
    public void action() {

        etat = "dehors";
        jetonPrésent = 0;

        ACLMessage msg = new ACLMessage(ACLMessage.INFORM); //envoyer le jeton au suivant
        msg.addReceiver(new AID(NomSuivant,AID.ISLOCALNAME));
        msg.setContent("Jeton");
        send(msg);

        System.out.println("Agent "+getLocalName()+" Je libère le jeton");
        block((int) (Math.random() * 10000));

        addBehaviour(new Dehors());
    }
}
```

```
    public class ConsulterBoite extends CyclicBehaviour{
        public void action() {

            ACLMessage msgRecu = receive();
            if (msgRecu != null ){
                if (msgRecu.getContent().equals("Jeton")){

                    System.out.println("Agent "+getLocalName()+" j'ai reçu "+ msgRecu.getContent()+
                              " de la part "+ msgRecu.getSender().getLocalName());

                    if (etat.equals("dehors")){  // je ne garde pas le jeton car je suis dans l'état dehors

                        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);  //envoyer le jeton au suivant
                        msg.addReceiver(new AID(NomSuivant,AID.ISLOCALNAME));
                        msg.setContent("Jeton");
                        send(msg);
                    }
                    else

                        jetonPrésent = 1;  // je garde le jeton car je suis dans l'état demandeur

                }
            }
            block((int) (Math.random() * 10000));
        }
    }
}
```

Classe test :

```
public class test {
    public static void main(String[] args) {
        String [] commande = new String[3];
        String argument ="";

        argument = argument+ "a:site(c,0)";
        argument = argument+";b:site(a,1)";
        argument = argument+";c:site(b,0)";

        commande [0]="-cp";
        commande [1]="jade.boot";
        commande [2]= argument;
        jade.Boot.main(commande);
    }
}
```