Ejercicio 1: CÁLCULO DE NÚMEROS PRIMOS

Se distribuirá el trabajo entre varios agentes y a cada uno de ellos se le asignará un rango de valores.

- **buscarTotalPrimos.java**: Agente principal que recibe un rango de números. Debe buscar en ese rango la cantidad de primos existente. Para ello busca agentes cuyo servicio sea “ayudar a calcular primos”: (1) obtiene los agentes que pueden ayudar con la tarea, (2) divide el rango principal en rangos según el número de agentes con esos servicios, (3) envía un mensaje indicando a cada uno de ellos el rango especifico que verificarán, (4) prepara la respuesta agrupando los resultados parciales de cada uno.
- **buscaPrimosRango.java:** Agente que recibe la propuesta de calcular los números primos existentes en un rango de números. En el mensaje, recibe un vector en el cual se le indica al agente cual fue el rango que le correspondió.
- **datosBusqueda.java:** clase que almacena los argumentos que usará cada agente para realizar la búsqueda en su rango específico: nombre agente (nombre), rango inicial (vi) y rango final (vf).

Descripción Básica del Funcionamiento:

- El agente **buscarTotalprimos**, utiliza el protocolo FIPA_CONTRACT_NET para contactar con todos los agentes que le ayudarán a buscar primos y un vector de objetos (**datosBusqueda**). Cada agente **buscaPrimosRango** busca en el vector qué objeto **datosBusqueda** le corresponde (según el nombre de agente) y a partir de los datos **vinicial** y **vfinal** procesa el rango para establecer el número de primos que contiene. Retorna el resultado a través del protocolo.
- El agente **buscarTotalprimos** agrega los resultados de cada uno de los agentes y muestra el número total de primos existente en todo el rango.

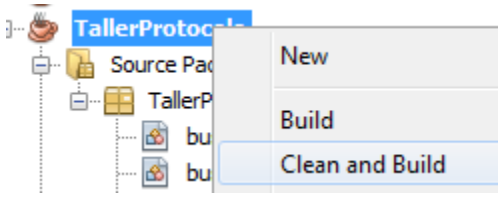
Pasos

a. Abrir el proyecto llamada TallerProtocolo, el cual contiene las clases .java : **buscarTotalPrimos.java, buscaPrimosRango.java, datosBusqueda.java.**
b. Configurar el proyecto con la librería jade.jar.
c. En la propiedad RUN, establecer los siguientes datos:

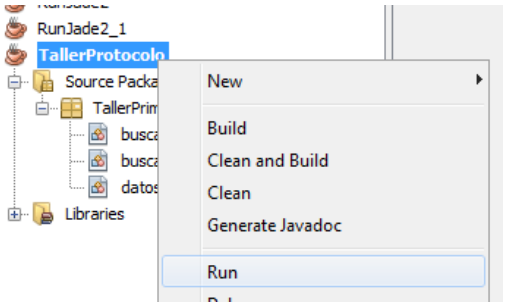
-gui ag2:TallerPrimos.buscaPrimosRango;ag3:TallerPrimos.buscaPrimosRango;ag4:TallerPrimos.buscaPrimosRango

En este caso, si queremos lanzar tres agentes buscadores entre rangos específicos.
Observemos que el agente principal no se lanza desde el RUN, lo lanzaremos desde la interfaz gráfica RMA.

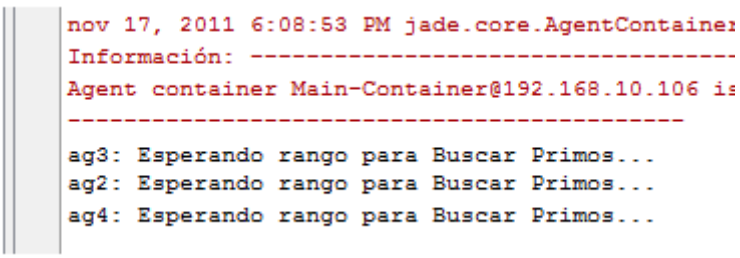
d. Dar clic derecho al proyecto y elegir **Clean and Build**



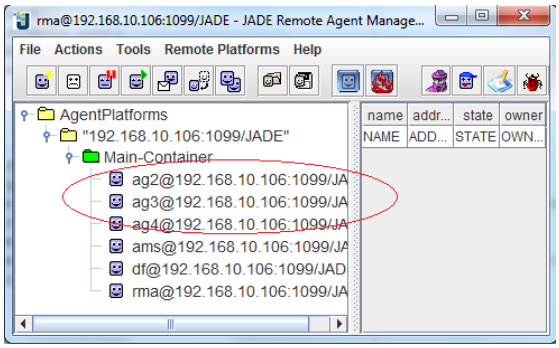
e. Dar Clid derecho al proyecto y elegir la opción RUN



f. Apreciamos que se lanzan los tres agentes buscadores de primos

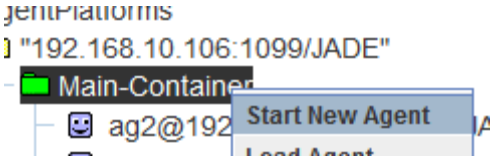


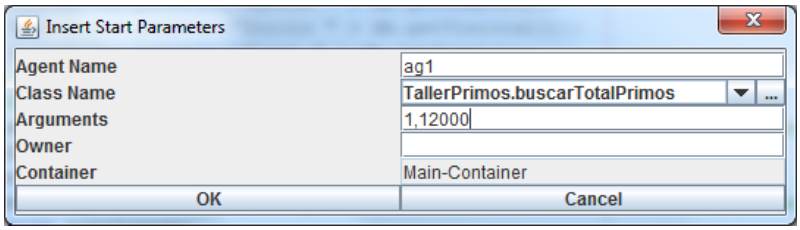
g. Aparece la interfaz gráfica RMA, mostrando igualmente los tres agentes que este ejemplo lanzamos



h. Procedemos a lanzar el agente principal **buscarTotalprimos,** que distribuirá el trabajo entre los tres actuales.

i. Damos entonces, clic derecho sobre Main-Container y elegimos la opción Start New Agent. Podemos llamarlo **ag1**; la clase en la cual está **TallerPrimos.buscarTotalPrimos;** y el rango debemos escribirlo en el campo argumentos, separado por comas, empezando por el valor inicial y luego el final (para el ejemplo 1,1200).





j. Vemos cómo los agentes calculan el número de primos de acuerdo al rango que le correspondió y al final, el agente principal agrega los valores

```
Información: ----------------------------------------
Agent container Main-Container@192.168.10.106 is ready.
--------------------------------------------
ag2: Esperando rango para Buscar Primos...
ag4: Esperando rango para Buscar Primos...
Calculando primos, tenemos 3 Agentes ayudando...
ag2: Peticion de calculo de primos recibida del ag1 ....
ag3: Peticion de calculo de primos recibida del ag1 ....
ag4: Peticion de calculo de primos recibida del ag1 ....
ag3: Calculando Primos entre 4000 y 7999 ...
ag4: Calculando Primos entre 8000 y 11999 ...
ag2: Calculando Primos entre 1 y 3999 ...
ag3: Encontre (457 Primos)
ag2: Encontre (550 Primos)
ag4: Encontre (431 Primos)
ag1: el total de primos es  1438:
```

Ejercicio 2: Agente chat que envía un mensaje a un conjunto de agentes

REVISAR

1) **Identificar la forma de recibir mensajes ACL**
2) **Identificar la estructura de enviar mensajes ACL por código**
3) **Identificar la forma de registrar servicios en la Plataforma**
4) **Diferencia entre comportamientos Ciclicos vs OneShotBehaviour (SimpleBehaviour)**

PASOS PARA RESOLVERLO

Modificar el ejemplo de TalkAgent

1) **En el setup del Agente TalkAgent, debe añadir un servicio para que en el Directorio Facilitador (DF) se informe a la plataforma de los agentes CHATS.**

```
// llamar a un método registrarservicio y tener en cuenta la configuración
    //  para luego buscarlo
    // ver el código fuente Inmobiliaria Línea 23 a la 36
    ServiceDescription servicio = new ServiceDescription();
     servicio.setType("CHAT");
     servicio.setName("CHAT");

     DFAgentDescription descripcion = new DFAgentDescription();
     descripcion.setName(getAID());
     descripcion.addLanguages("castellano");
     descripcion.addServices(servicio);

     try {
        DFService.register(this, descripcion);
     } catch (FIPAException e) {
        e.printStackTrace();
     }
```

2) **Añada un botón en el Formulario de VentanaChat, que cuando el usuario active esta opción invoque un método (actionperformed) para decirle al agente que procese un evento en la interfaz gráfica y envíe un mensaje a todos los agentes que tienen un servicio de chatear**

3) En el action performed del botón enviar TODOS

```
// configurar el GuiEvent como el ejemplo anterior
   // GuiEvent  evento = new GuiEvent (this, 1);
  //evento.addParameter(this.jTextField2.getText());
   //owner.postGuiEvent(evento);
   // verifique si necesita todos los campos
   // debe tener en cuenta el número de parámetros que va asociado al evento
```

4) En el evento OnGuiEvent se debe procesar el evento 1 (código para que busque todos los Agentes que ofrecen el servicio de CHAT y genere un mensaje para todos)

```
@Override
protected void onGuiEvent(GuiEvent ge) {

    if (ge.getType() == 0)// origen Evento
    {
        ACLMessage msg = new ACLMessage(ACLMessage.CONFIRM);
        AID x = new AID ((String)ge.getParameter(0),
                         AID.ISLOCALNAME);
        msg.addReceiver(x);
        msg.setContent((String)ge.getParameter(1));

        send(msg);
    }
    if (ge.getType() == 1)// Boton enviar a Todos
    {
     ACLMessage msg = new ACLMessage(ACLMessage.CONFIRM);
     // el contenido del mensaje tiene como argumento el texto que viene interfaz
    // VErificar este ejemplo
     //  msg.setContent((String)ge.getParameter(1));


    // Buscar el código de Buscar Servicio  -- Sin Techo desde la linea 39 hasta la  54

        // Generar el mensaje acL con todos los agentes
       // ciclo  ver linea 55 a la 60 en Sin TEcho
        //

        send(msg);
    }
```

Ejercicio 3: Agente JESS

1. **Descargar la librería jess.jar en**
   https://www.jessrules.com/jess/download.shtml

2. **Modelar el sistema de reglas médicas extraído de:**
   http://www.uco.es/users/sventura/misc/TutorialCLIPS/Practica1.htm