Guida al primo utilizzo di Alchemist

Giorgio Audrito

28 maggio 2019

1 Installazione e preparazione

1. Scarica il progetto di esempio, disponibile al seguente indirizzo:

   bitbucket.org/gaudrito/alchemist-example

2. Installa la virtual machine Java SE 8 JDK:

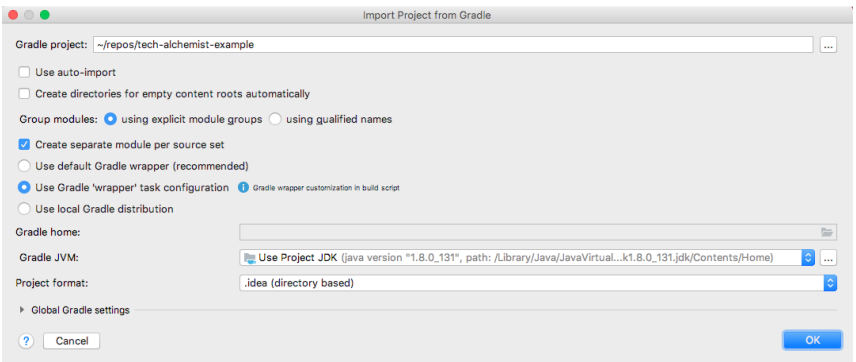
   `oracle.com/technetwork/java/javase/downloads`

3. Installa l'editor integrato IntelliJ (versione Community JBR 8):

   `jetbrains.com/idea/download`

   Durante l'installazione, seleziona il plugin per il supporto a Scala.

4. Apri IntelliJ e clicca su `File` > `Open`. Seleziona il file `build.gradle` presente nella cartella con il progetto di esempio scaricato. Clicca sul pulsante `Open as Project` e poi compila il form secondo questa schermata:

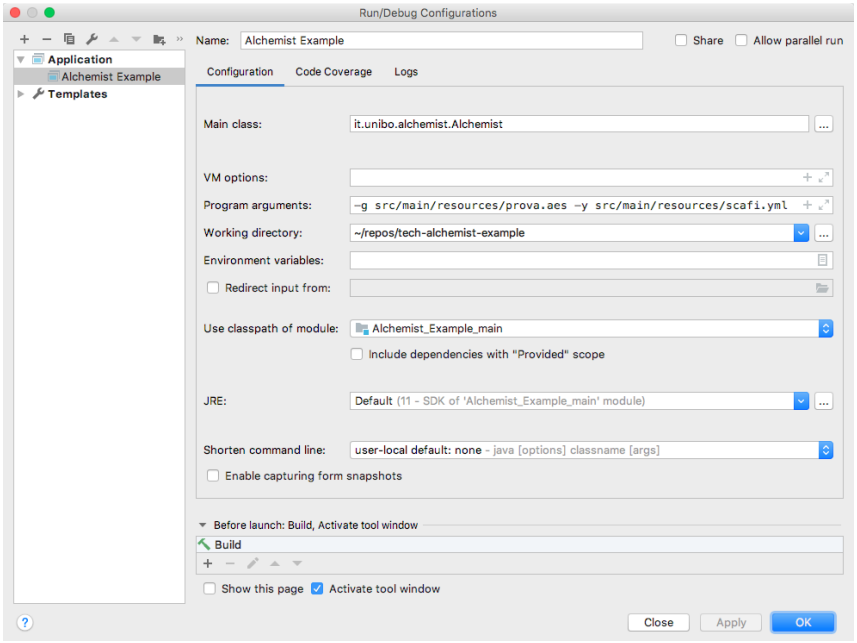


5. Apri il file `src/main/prova.scala`. Comparirà un banner giallo in alto, su cui puoi cliccare `Setup Scala SDK`. Se non comparisse, cerca la voce di menu `File > Project Structure > Global Libraries`. Compariranno poi delle finestre in cui potrai cliccare `Create > Download`, selezionare una versione `2.12.x` (per qualunque `x`) e poi premere `OK`. Attendi che lo scaricamento termini e poi premi ancora `OK`.

6. Esegui il progetto di esempio per accertarti che tutto abbia funzionato. Clicca su `Add Configuration > + > Application` e poi imposta:

   - Main class: `it.unibo.alchemist.Alchemist`
   - Program arguments: `-g src/main/resources/prova.aes -y src/main/resources/scafi.yml`
   - Use classpath of module: `Alchemist_Example_main`

   Il risultato dovrebbe somigliare a questa schermata:



   Premi quindi `OK` e poi il tasto play verde. Dopo qualche secondo si aprirà l'interfaccia di Alchemist, e potrai premere la lettera `P` per avviare e fermare la simulazione.

2 Il progetto di esempio

2.1 Ispezionare e modificare il progetto

Nel project explorer (sulla sinistra), espandete la voce del progetto di esempio, e le sotto-voci `src/main/resources`, `src/main/scala`. In quelle cartelle si trovano i file principali di questo progetto.

- `prova.aes`: Gli effetti grafici usati dalla visualizzazione interattiva.
- `prova.scala`: Il programma che viene eseguito, in linguaggio Scafi.
- `prova.pt`: Il programma che viene eseguito, in linguaggio Protelis.
- `scafi.yml`: La descrizione dello scenario di simulazione per Scafi.
- `protelis.yml`: La descrizione dello scenario di simulazione per Protelis.

Esaminiamoli uno per uno.

2.1.1 Il programma

Il programma contiene innanzitutto la definizione di una funzione `constant` per ottenere valori che restano costanti al passare dei round, poi la funzione `gradient` che calcola distanze, seguita dalla funzione principale `main`. Quest'ultima inizia con due blocchi di codice (uno per impostare il nodo sorgente, e uno per salvare le stime di distanza). Infine ci sono due blocchi per regolare il movimento dei dispositivi: nel primo si determina un momento `timeToGo` in cui iniziare a muoversi, e nel secondo si determina un `target` se è arrivato il momento di muoversi. Cercate di identificare questi blocchi e capire come funzionano.

2.1.2 Lo scenario di simulazione

Lo scenario di simulazione contiene diverse sezioni. All'inizio, vengono impostate alcune informazioni generali: la scelta del linguaggio (Scafi o Protelis), dei semi per il generatore di numeri casuali, il modello di connessione dei device (nel nostro caso, se la distanza Euclidea è minore di 1).

Nella sezione `pools` vengono descritti gli eventi da eseguire (esecuzione del programma, movimento Browniano, movimento verso un obiettivo), assieme con una distribuzione temporale (frequenza) e altri parametri.

Infine nella sezione `displacements` vengono disposti i dispositivi secondo delle forme geometriche (punto, cerchio, rettangolo) e gli vengono associati degli eventi da eseguire.

2.1.3 Gli effetti grafici

Il file degli effetti grafici non è fatto per essere editato manualmente. Potete impostare gli effetti direttamente dall'interfaccia grafica di Alchemist, cliccando sul riquadro rettangolare con su scritto "DS" (in alto a sinistra a fianco delle frecce). Nella finestra di dialogo che compare, vedrete scritto che l'effetto attuale fa tuning dei colori dei nodi usando i valori della "molecola" `gradient` (valore esportato nel programma tramite la funzione `put`), assieme a dei parametri che regolano la scala di colori.

3 Esercizi proposti

Modificate il file `prova.scafi`, incrementalmente, per calcolare in ogni device le seguenti cose.

1. il numero di device vicini
2. il massimo numero di vicini che il device corrente ha mai avuto
3. il massimo numero di vicini che un qualunque device della rete ha mai avuto
4. muoversi verso il vicino che ha meno vicini
5. muoversi lontano dal vicino che ha più vicini
6. combinare gli ultimi due punti: ogni device attratto dal vicino con meno vicini, e respinto dal vicino con più vicini.

3.1 Suggerimenti

- Nei primi esercizi, ragionate dove usare `nbr` ("guardo i vicini") e `rep` ("guardo il passato").
- Per muovere un device, bisogna memorizzare una posizione `List(x,y)` nella molecola `target`:

  ```
  node.put("target", ...)
  ```

  Sostituite la porzione di codice che ora si occupa del movimento con quanto richiesto.
- Si può ottenere la posizione corrente di un device tramite il metodo `getCoordinates()`.

4 Risorse utili

- La pagina ufficiale di Scafi:

  `scafi.github.io`
- La pagina ufficiale di Protelis:

  `protelis.github.io`
- La pagina ufficiale di Alchemist:

  `alchemistsimulator.github.io`
- Un introduzione all'Aggregate Programming, con presentazione:

  `www.sti.uniurb.it/events/sfm16quanticol/slides/beal_1.pdf`
  `www.sti.uniurb.it/events/sfm16quanticol/slides/beal_2.pdf`
  `www.sti.uniurb.it/events/sfm16quanticol/slides/beal_3.pdf`

  e con articolo:

  `web.mit.edu/jakebeal/www/Publications/`
  `QUANTICOL16-AggregateProgramming.pdf`