Metodologia di Programmazione
Specifica Progetti Giugno/Luglio/Settembre 2023

L'obiettivo del progetto è quello di realizzare una libreria Java che consenta la simulazione di uno *sciame di robot* che si muovono nello spazio.

L'ambiente dove i robot si muovono è costituito da una superficie piana, di dimensioni illimitate. All'interno dell'area sono presenti delle aree (di forma circolare o rettangolare) che indicano particolari condizioni. Ad ogni condizione è associata una etichetta (label). Ogni posizione nell'area è identificata per mezzo delle coordinate *(x,y)* e si assume come unità di misura i *metri*.

Ogni robot è in grado di:
- Percepire l'ambiente che li circonda;
- Segnalare una propria *condizione*;
- Muoversi nella direzione dei robot ad una certa distanza;
- Muoversi verso una certa direzione;
- Fermarsi.

Assumiamo che l'esecuzione di ogni singola istruzione richiede 1 secondo. E' possibile considerare un tempo di esecuzione "variabile" gestito attraverso le classi o l'interfaccia grafica.

Il linguaggio di programmazione dei Robot
Ogni robot esegue un programma costituito da una sequenza di istruzioni della seguente forma:

- `MOVE x y s`: si muove verso la direzione *(x,y),* dove x ed y sono valori compresi tra -1 ed 1 alla data velocità *s* espressa in *m/s,* si assume che al più una tra x ed y sia diversa da 0*;*
- `MOVE RANDOM x1 x2 y1 y2 s`: si muove alla velocità *s* espressa in *m/s* verso una posizione (x,y) scelta casualmente nell'intervallo *[x1, x2] e [y1, y2];*
- `SIGNAL label`: segnala una particolare condizione, *label* è un identificativo costituito da una sequenza di caratteri alfanumerici e dal simbolo '_'*;*
- `UNSIGNAL label`: termina la segnalazione della condizione*;*
- `FOLLOW label dist s`: si muove muove alla velocità *s* espressa in *m/s* una direzione che è la media delle posizioni dei robot che segnalano la condizione *label* e che di trovano ad una distanza minore o uguale a *dist. Se non esistono robot viene scelta una direzione a caso nell'intervallo [-dist, dist]x[-dist, dist].*
- `CONTINUE s`: continua a muoversi per *s* secondi con la medesima direzione e velocità*.*
- `STOP`: *arresta il proprio movimento. Il robot non si muove più ma continua a segnalare le proprie condizioni.*

Oltre alle istruzioni sopra, un programma può eseguire i seguenti comandi di selezione ed iterazione:

Ripete *n* volte una sequenza di istruzioni:

```
REPEAT n
cmd1
…
cmdn
DONE
```

Ripete delle istruzioni fino a quando una certa condizione non è percepita nell'ambiente (ossia il robot è all'interno di un'area con la label richiesta):

```
UNTIL label
cmd1
…
cmdn
DONE
```

Ripete un comportamento per sempre:

```
DO FOREVER
cmd1
…
cmdn
DONE
```

Specifica dell'ambiente
Per specificare l'ambiente dove i robot operano viene utilizzato un file di testo contenente una sequenza di linee della seguente forma:

label CIRCLE x y r
label RECTANGLE x y w h

Usate per rappresentare aree nella posizione (x,y) con l'etichetta data rispettivamente della forma circolare, con raggio r, o rettangolare con altezza e larghezza w. In ambedue i casi la posizione indica il punto centro della figura.

Le classi per il parsing del file con il programma del robot e la specifica dell'ambiente sono disponibili al seguente repository GitHub:

https://github.com/michele-loreti/followme

Il Progetto

Sviluppo Base (Valutazione massima 24)

Il progetto a questo livello di sviluppo dovrà:

- Definire una gerarchia di classi ed interfacce per rappresentare il comportamento dei robot;
- Definire una gerarchia di classi ed interfacce per descrivere l'area di movimento;
- Definire una gerarchia di classi ed interfacce per la simulazione del sistema. In particolare dovrà essere fornita la possibilità di costruire un insieme di robot posizionati in modo randomico all'interno dell'area che si muovono tutti seguendo uno specifico movimento. Dovrà essere poi data la possibilità di generare, a partire da una data configurazione, una simulazione del sistema entro un certo tempo t dove i comandi dei robot vengono eseguiti ogni dt secondi (in questo caso il tempo non è quello *reale*, ma quello di *simulazione*).

Sviluppo Avanzato (Valutazione massima 30 e Lode)

Realizzare un'interfaccia grafica che possa consentire di visualizzare l'evoluzione del sistema mostrando garantendo una *simulazione passo passo* o un'esecuzione per un determinato periodo di tempo/passi.

Griglia di valutazione

Il progetto verrà valutato secondo la seguente griglia di valutazione:

1. Definizione delle responsabilità, le responsabilità dovranno essere descritte in una breve relazione, in pdf, allegata al progetto (**NB: descrivere le responsabilità non vuol dire elencare le interfacce e le classi sviluppate**);
2. Coerenza dell'implementazione delle responsabilità. Per ogni responsabilità individuata al punto precedente indicare le interfacce e le classi che la implementano;
3. Rispetto dei principi SOLID;
4. Pulizia del codice ed assenza di Code Smell;
5. Estendibilità del codice prodotto;
6. Organizzazione del progetto (Gradle): il progetto gradle dovrà funzionare senza errori eseguendo i comandi "gradle build" e successivamente "gradle run" (quest'ultimo con eventuali parametri aggiuntivi).
7. Documentazione del codice (formado javadoc);
8. Presenza dei test (in proporzione al numero di classi e metodi testati);
9. Uso dei principi e metodologie viste a lezione;
10. Rispetto delle specifiche di consegna.

Modalità di Consegna

Il progetto dovrà essere consegnato in una cartella denominata CognomeNomeMatricola in un archivio *zip, tgz* o *tar.gz* con lo stesso nome. **L'uso di altri formati di consegna non è permesso! Nel caso il progetto non verrà valutato e considerato come non consegnato (non potendo quindi partecipare alla prova scritta)**.

All'interno della cartella CognomeNomeMatricola si dovrà trovare:

- I sorgenti del progetto *gradle*
- **Un pdf con la descrizione delle responsabilità assegnate, delle loro implementazioni, e delle istruzioni necessarie ad eseguire gli esempi consegnati necessari alla valutazione del progetto.**