Relazione progetto sciame di robot

Autore: Jacopo Mucchietto

Descrizione delle responsabilità assegnate

- **Rappresentazione e gestione dell'ambiente**: la responsabilità di rappresentare e gestire l'ambiente è stata assegnata all' interfaccia I**Environment** che viene implementata dalla classe **Environment**. Environment tiene traccia dei Robot e delle Aree presenti al suo interno e fornisce metodi per restituire i robot e le aree che soddisfano determinate condizioni.
- **Rappresentare delle aree:** la responsabilità di rappresentare un'area è stata assegnata alla classe **Circle** e **Rectangle** che implementano l'interfaccia **IArea.** Circle rappresenta un'area di forma circolare e Rectangle di forma rettangolare**.** Entrambe le classi sono in grado di determinare se un punto è all'interno della loro area.
- **Rappresentazione di un punto 2D:** la responsabilità di rappresentare un punto in uno spazio bidimensionale è stata assegnata alla classe **Point.**
- **Rappresentazione delle label**: la responsabilità di rappresentare una label è stata assegnata all'interfaccia **ILabel** che viene implementa dalla classe **BasicLabel.** BasicLabel rappresenta una serie di caratteri alfanumerici e dal simbolo '_'.
- **Calcolare la distanza da percorrere in base alla velocità:** la responsabilità di determinare la posizione che deve raggiungere un robot, data la sua posizione iniziale, la direzione e la velocità è stata assegnata alla classe **DirectionCalculator.**
- **Rappresentazione dei comandi:** la responsabilità di rappresentare i comandi è stata assegnata all'interfaccia **ICommand** e ai comandi concreti che la implementano.
- **Rappresentazione dei comandi iterativi:** l'interfaccia **IIterativeCommands** che viene implementata da **DoForeverCommand, RepeatCommand** *e da* **UntilCommand** è responsabile di fornire un contratto per la rappresentazione dei comandi iterativi in modo che sia possibile la ripetizione dei comandi.
- **Rappresentare lo stato del Robot:** la classe **Robot** che implementa **IRobot** è responsabile di rappresentare il suo stato, contiene una lista di comandi che vengono eseguiti quando viene chiamato il metodo Execute().
- **Avviare la simulazione:** la responsabilità di avviare la simulazione è stata assegnata alla classe **Simulator** che implementa l'interfaccia **ISimulator.**
- **Configurazione della simulazione**: la responsabilità di configurare la simulazione, leggendo il file delle aree e del programma dei Robot è stata assegnata alla classe **SimulationConfigurator**.
- **Gestione del parsing dei comandi:** la responsabilità di gestire il parsing dei comandi è stata assegnata alla classe **ParserManager** e **ProgramParserHandler**. **ParserManager** esegue il parsing mentre **ProgramParserHandler,** che implementa l'interfaccia **FollowMeParser,** viene utilizzata per creare i comandi che sono stati parsati.
- **Creare le aree:** la responsabilità di determinare quale area deve essere creata durante il parsing è stata assegnata a **ShapeDataAreaFactory** che implementa **IAreaFactory** e a **CircleFactory** e **RectangleFactory**.

Estendibilità

Per garantire l'estendibilità del codice prodotto è stato utilizzato il Command Pattern, che rende più facile l'aggiunta di nuovi comandi tramite l'implementazione dell'interfaccia ICommand. L'interfaccia ICommand è stata progettata per rappresentare un comando generico, e le classi concrete che la implementano definiscono comportamenti specifici.

Esecuzione della simulazione

Per poter eseguire la simulazione è necessario eseguire il seguente comando - gradle run

Durante l'esecuzione di gradle run inoltre vanno inseriti i parametri per avviare la simulazione.

Una volta eseguito il comando bisogna inserire il numero di robot che devono essere presenti nell'environment, il time interval dei comandi e il tempo di simulazione. Il tempo di simulazione deve essere maggiore rispetto al time interval.

```
Enter the number of robots: 1
Enter the time for command execution: 1
Enter the total duration of the simulation: 10
```