Collegamenti utili

Una completa descrizione del gioco e delle sue regole è disponibile al seguente link.

É disponibile una repository GitHub del progetto al seguente link.

Specifica del Progetto

Formula 1 è un gioco di carta e matita che si gioca su un foglio di carta quadrettata, sul quale viene disegnato un circuito automobilistico di fantasia, con una linea di partenza e una linea di arrivo (che possono anche coincidere se il circuito è circolare). Il gioco simula una gara tenendo conto dell'inerzia dei veicoli (per esempio, è necessario frenare quando si affronta una curva).
L'obiettivo del progetto è quello di realizzare le classi che permettano di gestire una gara dove *giocatori interattivi* (umani) e *giocatori bot* (programmati) concorrono per vincere la gara.

Livello di Sviluppo

Sviluppo Base

Il progetto consegnato a questo livello di sviluppo dovrà mettere a disposizione le interfacce e le classi che implementano i seguenti elementi:

- Il *tracciato di gara*;
- I *giocatori bot*;
- Il *motore di gioco*.

L'applicazione sviluppata dovrà essere in grado di:

- Caricare da file la configurazione del tracciato insieme alla lista dei giocatori bot presenti (il formato deve essere definito dallo studente);
- Fornire il supporto alla *simulazione* di una gara;
- Mostrare tramite *console* l'avanzamento della gara (posizione dei giocatori nel tracciato e loro stato).

Relazione del Progetto

Introduzione

Il progetto è un gioco di simulazione che riproduce una gara automobilistica su un circuito rappresentato da una griglia su cui si muovono dei bot. Questi bot competono per raggiungere un traguardo mentre il motore di gioco gestisce le dinamiche di movimento dei bot e la verifica delle condizioni di vittoria o squalifica.

La versione di Java utilizzata è Java 21, e la versione di Gradle è Gradle 8.8.

Il progetto è dotato di commenti Javadoc scritti strettamente in lingua italiana per una chiara documentazione delle classi e dei metodi.

Per la verifica del funzionamento del codice del progetto, sono presenti le relative classi di Test con i test dei metodi implementati.

Responsabilità delle Classi e Implementazione

1. Coordinata

Responsabilità: La classe `Coordinata` è il blocco fondamentale di costruzione per il sistema di gioco, rappresentando ogni punto su un piano cartesiano con due coordinate intere, x e y. La sua funzione principale è quella di fornire un metodo unificato per identificare e manipolare le posizioni delle celle sulla mappa di gioco. In particolare, `Coordinata` viene utilizzata per definire e aggiornare le posizioni dei bot e per stabilire relazioni spaziali tra diversi elementi del gioco, come le linee di partenza e di arrivo, e i movimenti dei bot.

La classe non implementa interfacce, ma funge da punto di riferimento per le altre classi del sistema, come `Macchina`, `Linea`, e `Engine`, che utilizzano `Coordinata` per navigare la mappa e calcolare le interazioni tra bot e ambiente di gioco.

2. Celle

Responsabilità: `Celle` è un tipo enumerativo che rappresenta i diversi stati che una cella sulla mappa può avere. Le celle possono essere, ad esempio, celle di partenza, traguardi, celle

occupate dai bot, zone fuori pista o celle vuote. La classe `Celle` definisce questi stati attraverso valori predefiniti, consentendo al sistema di gioco di gestire e visualizzare il layout della mappa in modo chiaro e coerente. La classe non implementa interfacce, ma i suoi valori vengono utilizzati da classi come `Mappa` per determinare lo stato attuale della mappa e aggiornare la visualizzazione del gioco.

3. LeggiMappa

Responsabilità: `LeggiMappa` è dedicata alla lettura e al parsing della configurazione della mappa di gioco da un file di testo. Questa classe si occupa di convertire i dati del file passato, che rappresentano la mappa con vari stati delle celle, in una griglia di Celle che può essere utilizzata dal sistema di gioco per creare la Mappa.

La configurazione letta viene poi impiegata dalla classe `Engine` per inizializzare il campo di gioco e iniziare una nuova partita, e dalla classe `Partita`.

Questa classe è essenziale per il setup del gioco, assicurando che la mappa di gioco sia creata correttamente prima che inizi la partita.

4. LeggiBot

Responsabilità: `LeggiBot` gestisce la lettura dei nomi dei bot da un file di testo e fornisce questi nomi alla classe `Engine` per l'inizializzazione dei bot. La classe si occupa della logica per leggere i dati dal file e creare una lista di nomi che saranno poi assegnati ai bot alla loro creazione prima della partita.

5. Mappa

Responsabilità: La classe `Mappa` rappresenta il campo di gioco stesso, strutturato come una griglia di celle. Essa gestisce la visualizzazione e lo stato della mappa, permettendo di accedere alle celle e modificarle, ad esempio, aggiornando la posizione dei bot o impostando le celle per il traguardo. Implementa l'interfaccia `PuntiMappa`, che definisce metodi per ottenere informazioni sulla mappa, come recuperare una cella da una coordinata, verificare la validità di una posizione, e accedere alle linee di partenza e di arrivo.

La classe `Mappa` è centrale per la gestione dello stato del gioco, poiché tiene traccia di tutte le celle e coordina le interazioni tra bot e ambiente di gioco. È utilizzata sia dalla classe `Engine` per gestire la partita, sia dalla classe `StampaMappa` per visualizzare la mappa sulla console.

6. Macchina

Responsabilità: `Macchina` rappresenta una singola pedina di gioco per un Bot e gestisce le sue operazioni di movimento all'interno della mappa. Implementa l'interfaccia `Spostamento`, che definisce metodi per calcolare la velocità, determinare le mosse iniziali e standard, e spostare la pedina da una coordinata a un'altra.

La classe `Macchina` è responsabile per la logica di movimento e per l'aggiornamento della posizione della pedina del bot durante il gioco. Si occupa di tutte le azioni che un bot può compiere nel corso della partita, inclusa l'implementazione delle mosse basate sulla posizione corrente e le regole del gioco.

7. Bot

Responsabilità: `Bot` rappresenta un giocatore programmato dal sistema per compiere spostamenti randomici tra quelli possibili della sua pedina (macchina) che partecipa alla gara. Implementa l'interfaccia `movimentoPlayer`, che definisce metodi per gestire il turno del bot e le sue azioni nel gioco. La classe `Bot` decide quale mossa effettuare e interagisce con `Macchina` per eseguire le azioni necessarie per progredire nel gioco.

La responsabilità principale di `Bot` è quella di simulare le decisioni e le strategie di un giocatore, utilizzando logiche di gioco per determinare le azioni da intraprendere e gestire i turni all'interno della partita.

8. Linea

Responsabilità: `Linea` gestisce le intersezioni e le relazioni geometriche tra segmenti di linee sulla mappa di gioco. Implementa l'interfaccia `Intersection`, che fornisce metodi per determinare se due linee si incrociano e per calcolare se un punto si trova su una linea.

La classe `Linea` è essenziale per determinare le dinamiche della gara, come verificare se i bot attraversano linee importanti come il traguardo o se ci sono intersezioni tra percorsi dei bot.

9. Engine

Responsabilità: `Engine` è il cuore del gioco, gestendo il flusso complessivo della partita. Implementa l'interfaccia `MotoreDiGioco`, che definisce metodi per creare i bot, calcolare le mosse, aggiornare la mappa e verificare lo stato della partita. `Engine` coordina tutte le componenti del gioco, iniziando la partita e gestendo il ciclo di vita del gioco, incluse le condizioni di vittoria e squalifica.

Questa classe è centrale per l'esecuzione del gioco, poiché controlla l'interazione tra i bot, la mappa, e le regole del gioco, assicurandosi che tutto proceda secondo le specifiche del gioco.

10. StampaBot

Responsabilità: `StampaBot` è responsabile per la visualizzazione delle informazioni sui bot durante la partita. Implementa l'interfaccia `PlayerPrinter`, che definisce metodi per annunciare i bot che partecipano alla gara e aggiornare la visibilità dei bot sulla mappa.

La classe `StampaBot` è importante per il feedback visivo del gioco, offrendo una rappresentazione chiara dei bot in gara.

11. StampaMappa

Responsabilità: `StampaMappa` si occupa della visualizzazione della mappa di gioco sulla console. Implementa l'interfaccia `MapPrinter`, che fornisce un metodo per stampare la mappa corrente.

Questa classe è essenziale per fornire una rappresentazione visiva del campo di gioco, permettendo agli utenti di vedere lo stato della mappa e le posizioni dei bot, e contribuendo a rendere il gioco più interattivo e comprensibile.

Descrizione delle Interfacce Utilizzate

1. PuntiCoordinata

Responsabilità: PuntiCoordinata gestisce le operazioni relative alle coordinate, come ottenere i punti vicini e verificare la validità della coordinata.

- **Classe:** Coordinata implementa PuntiCoordinata.
- **Metodi:**
  - ottoVicini() restituisce una lista delle otto coordinate adiacenti.
  - isPositive() verifica se entrambe le coordinate x e y sono non negative.

2. PuntiMappa

Responsabilità: PuntiMappa gestisce le operazioni sulle celle della mappa, inclusa la verifica delle posizioni e la gestione della linea di partenza e di arrivo.

- **Classe:** Mappa implementa PuntiMappa.
- **Metodi:**
  - getCellaFromCoordinata(Coordinata coordinata) restituisce la cella associata a una coordinata.
  - setCella(Coordinata c, Celle cella) associa una cella a una coordinata.
  - isNotValid(Coordinata c) verifica se una coordinata è valida per la mappa.
  - isPosition(Coordinata c, Celle cella) verifica se una coordinata è associata a una certa cella.
  - getStart() restituisce le coordinate della linea di partenza.
  - getFinish() restituisce le coordinate della linea di arrivo.
  - getFinishLine() restituisce la linea di arrivo.

3. Spostamento

Responsabilità: Spostamento gestisce il movimento delle macchine, calcolando la velocità e determinando le mosse disponibili.

- **Classe:** Macchina implementa Spostamento.
- **Metodi:**
  - velocita() aggiorna la velocità della macchina.
  - mosseIniziali() calcola le mosse iniziali disponibili.
  - mossaStandard() fornisce una mossa standard.
  - mosse() crea una lista di tutte le mosse possibili.
  - moveTo(Coordinata coordinata) sposta la macchina a una nuova coordinata.

4. Intersection

Responsabilità: Intersection gestisce le intersezioni tra linee e verifica se un punto è su una linea o se due linee si incrociano.

- **Classe:** Linea implementa Intersection.
- **Metodi:**
  - interseca(Linea altra) verifica se due linee si intersecano.
  - intersezione(Coordinata c1, Coordinata c2, Coordinata q1, Coordinata q2) calcola l'intersezione tra due segmenti.
  - direzione(Coordinata a, Coordinata b, Coordinata c) determina la direzione rispetto a un segmento.
  - sullaLinea(Coordinata a, Coordinata b, Coordinata c) verifica se un punto è sulla linea.

5. MapPrinter

Responsabilità: MapPrinter si occupa della visualizzazione della mappa di gioco.

- **Classe:** StampaMappa implementa MapPrinter.
- **Metodi:**
  - stampaMappa() visualizza la mappa di gioco corrente.

6. MotoreDiGioco

Responsabilità: MotoreDiGioco gestisce l'intero processo di gioco, dalle mosse dei bot alla verifica dello stato della partita.

- **Classe:** Engine implementa MotoreDiGioco.
- **Metodi:**
  - creaPlayers() crea i bot che parteciperanno alla partita.
  - mossePossibili(Bot giocatore) calcola le mosse possibili per un bot.
  - checkPlayer(Bot giocatore) verifica se un bot è squalificato.
  - playerState() verifica lo stato dei bot.
  - matchState(Bot giocatore) verifica lo stato della partita.
  - aggiornaMappa() aggiorna la mappa con le posizioni dei bot.
  - startPartita() inizia una nuova partita.

7. PlayerPrinter

Responsabilità: PlayerPrinter gestisce la stampa delle informazioni sui bot in gara.

- **Classe:** StampaBot implementa PlayerPrinter.
- **Metodi:**
  - stampaStart() annuncia i bot che partecipano alla gara.
  - stampaPlayers() aggiorna la visibilità dei bot sulla mappa.

8. movimentoPlayer

Responsabilità: movimentoPlayer gestisce le azioni e i turni dei bot durante la partita.

- **Classe:** Bot implementa movimentoPlayer.
- **Metodi:**
  - move() gestisce il movimento del bot.
  - turno() gestisce un turno di gioco per il bot.

Esempi di Utilizzo del Progetto

Per eseguire il progetto e testarne il funzionamento, vanno seguiti questi passaggi:

Preparare i File di Configurazione:

a. Crea un file **txt** di configurazione per la mappa, ad esempio map.txt, che contiene una griglia di valori interi.
b. Crea un file **txt** di nomi per i bot, ad esempio bot_names.txt, che contiene una lista di nomi per i bot.

Eseguire il Codice:

c. Avvia il progetto utilizzando la classe Partita con i nomi dei file preparati. Assicurati che i file si trovino nella stessa directory del progetto o fornisci il percorso corretto.

Verificare il Funzionamento:

d. Dopo aver avviato Partita, osserva la console per vedere la mappa di gioco e le mosse dei bot. La classe Partita gestisce la partita e la visualizzazione dello stato della gara.