Introduzione

VectorRace è un progetto di simulazione di gara sviluppato in Java, in cui giocatori umani e bot competono su un tracciato predefinito, evitando ostacoli e cercando di raggiungere il traguardo per primi. Il progetto è stato progettato seguendo i principi SOLID per garantire un codice modulare, estensibile e facilmente manutenibile. Questa relazione descrive le responsabilità assegnate, le loro implementazioni, come implementare strategie avanzate per i bot e fornisce le istruzioni necessarie per eseguire gli esempi del progetto.

Ho progettato il gioco impostando una velocità massima di 3 unità, un valore che può essere modificato in base alla dimensione del tracciato o alle preferenze personali.
Il sistema di movimento consente di spostarsi in otto direzioni cardinali (su, giù, sinistra, destra e le diagonali). Inoltre, il giocatore può controllare la velocità selezionando una modalità di accelerazione: è possibile accelerare, decelerare o mantenere una velocità costante/invariata, a seconda delle esigenze o delle condizioni di gioco.
Questo approccio permette una gestione strategica dei movimenti, adattandosi alle diverse situazioni che il giocatore può incontrare durante la partita.

Responsabilita’

- Gestione del tracciato di gioco
    1. Definizione e Caricamento del Tracciato: Il tracciato di gioco viene caricato da un file.txt
    2. Verifica dello Stato delle Celle: Determina se una cella è libera, contiene un ostacolo o rappresenta il traguardo.
- logica di movimento e direzione
    1. Scelta delle Direzioni: Ogni giocatore, sia umano che bot, può scegliere una direzione di movimento tra quelle consentite in base alla sua velocità e alla direzione precedente.
    2. Calcolo delle Nuove Posizioni: Calcola le nuove posizioni dei giocatori in base alla direzione scelta e alla velocità corrente.
- Gestione del flusso di gioco
    1. Avvio e Terminazione della Gara: gestire l'inizio e la fine della gara, determinando quando terminare la partita in base a condizioni specifiche come il raggiungimento del traguardo da parte di un giocatore o il superamento del numero massimo di turni.
    2. Iterazione sui Turni: Gestisce l'ordine e la sequenza dei turni, assicurando che ogni giocatore (umano o bot) abbia la possibilità di agire in ogni turno.
- Gestione dei giocatori
    1. Creazione e Aggiunta dei Giocatori: Permette di aggiungere diversi tipi di giocatori, inclusi giocatori umani e bot, alla gara.
    2. Gestione delle Posizioni e Velocità: Tieni traccia delle posizioni attuali e delle velocità di ogni giocatore, aggiornandole in base alle azioni intraprese durante i turni.

Implementazione delle responsabilita’

GameEngine
- Gestione del flusso di gioco: turni, processazione delle azioni dei giocatori, gestione delle collisioni e conclusione della gara.

GameBoard
- Gestione dinamica delle posizioni dei giocatori e dell'interazione con il tracciato.

ITrack
- Rappresentazione e gestione del tracciato di gioco.

IVelocityCalculator
- Calcolare la velocità dei giocatori.

IInertiaManager
- Determinare le direzioni consentite per un giocatore in base alla sua velocità e alla direzione precedente.

IPlayer
- Definire le interfacce comuni per tutti i tipi di giocatori (umani e bot), inclusi metodi per scegliere direzioni e accelerazioni.

BasePlayer
- Fornire implementazioni di base comuni a tutti i giocatori, come la gestione del nome, della posizione e della velocità.

Position
- Rappresentazione delle coordinate di una posizione sul tracciato, operazioni sulle posizioni.

VectorDirection
- Definire le direzioni cardinali (N, NE, E, SE, S, SW, W, NW) utilizzate per il movimento dei giocatori.

Strategia giocatori/Bot
Per implementare una strategia Bot diversa da quelle qui presenti si possono ridefinire i metodi ChooseDirection e ChooseAccelleration in modo cambiare l’approccio decisionale di movimento e di accelerazione di un Bot. in questo modo si puo’ cambiare la decisione scelta del Bot senza interferire nella fisica e logica del gioco.

ChaserBot
- implementazione specifica di un bot che insegue un target, evitando collisioni e gestendo la velocita’.

SafeRunnerBot
- SafeRunnerBot cerca di correre verso il traguardo, ma allo stesso tempo evita di avvicinarsi troppo agli altri giocatori.

Entrambi questi Bot prendono in considerazione la conformazione del tracciato, le posizioni dei giocatori presenti in gara, tenendo conto della velocità del giocatore e della sua posizione.
All’interno del package Giocatori ho creato altri bot che implementano diverse strategie, e’ possibile aggiungerli nella classe Main per vedere il loro comportamento.

HumanPlayer
- Strategia di gioco di un giocatore interattivo, che permette di scegliere la direzione e l'accelerazione in maniera manuale.

Istruzioni per eseguire il progetto:

1. La classe Main configura e avvia l’intero gioco gestendo il caricamento del tracciato, l’inizializzazione delle componenti di gioco, la creazione e registrazione dei giocatori e l’avvio della gara.
2. Tramite il terminale Gradle:
    a. Compilare il progetto usando Gradle con il comando `gradle build`.
    b. Eseguire i test con `gradle test`.
    c. Per avviare l'applicazione, utilizzare il comando `gradle run`