Relazione del Programma Formula 1: Di Biase Michela (matricola: 113966)

Introduzione

Il programma descritto è un simulatore di gare automobilistiche (Formula 1) in cui i giocatori, sia umani che bot, competono su un tracciato di gara. Il progetto contiene diverse classi e interfacce che gestiscono vari aspetti del gioco, dalla gestione della gara alle macchine e ai giocatori. Il fulcro del programma è la classe `GestioneGara` che coordina il progresso del gioco attraverso vari round.

Classi e Responsabilità

1. `GestioneGara`

Questa classe implementa l'interfaccia `IGestioneGara` e ha il compito di gestire il progresso della gara, verificare la validità delle mosse, controllare lo stato delle macchine e determinare il vincitore.

Metodi:

- `void roundSuccessivo()`: aggiorna il round corrente, riportandolo a 1 dopo il terzo round.
- `int getRound()`: ritorna il numero del round corrente.
- `boolean controlloInGara(Macchina macchina, Locazione prossimaPosizione)`: verifica se una macchina è ancora in gara dopo uno spostamento.
- `boolean controlloMossa(ArrayList<Locazione> possibiliMosse, Locazione prossimaPosizione)`: verifica se una mossa è valida.
- `boolean controlloVittoria(Macchina macchina)`: verifica se una macchina ha raggiunto il traguardo.

2. `Giocatore`

Classe astratta che rappresenta un giocatore, sia esso umano o bot. Implementa l'interfaccia `IGiocatore` e contiene le informazioni e i metodi base per i giocatori.

-Metodi:

- `String getNome()`: ritorna il nome del giocatore.
- `TracciatoDiGara getTracciatoDiGara()`: ritorna il tracciato di gara.
- `void sceltaMacchina(Macchina macchina)`: permette di scegliere una macchina.
- `Macchina getMacchina()`: ritorna la macchina del giocatore.
- `ArrayList<Locazione> getProssimeMossePossibili()`: calcola le possibili mosse della macchina.
- `void spostaMacchinaIn(Locazione prossimaPosizione) throws Exception`: sposta la macchina in una nuova posizione se la mossa è valida e la macchina è ancora in gara.

3. `Bot`

Sottoclasse di `Giocatore` che rappresenta un giocatore automatico. Implementa un metodo per spostare la macchina automaticamente.

- Metodi:

  - `void spostamentoMacchina() throws Exception`: sceglie una mossa casuale tra le possibili mosse e sposta la macchina.

4. `Utente`

Sottoclasse di `Giocatore` che rappresenta un giocatore umano.

5. `Macchina`

Classe che rappresenta una macchina da corsa con posizione, colore, stato e capacità di movimento.

- Metodi:

  - `boolean controlloMovimento(int spostamentoX, int spostamentoY)`: verifica se un movimento è valido.

  - Vari metodi per calcolare le possibili mosse (e.g., `sinistra`, `altoSinistra`, `destra`, ecc.).

  - `ArrayList<Locazione> calcoloProssimeMosse(int altezza, int lunghezza)`: calcola tutte le possibili mosse.

  - `void spostamento(Locazione nuovaPosizione)`: aggiorna la posizione della macchina.

  - Getter e setter per gli attributi della macchina.

6. `TracciatoDiGara`

Classe che rappresenta il tracciato di gara, con informazioni sulla disposizione delle locazioni e metodi per gestire le macchine sul tracciato.

- Metodi:

  - `int getAltezza()`, `int getLunghezza()`: ritorna le dimensioni del tracciato.

  - `ArrayList<Locazione> getPosizioniIniziali()`, `ArrayList<Locazione> getTraguardo()`: ritorna le posizioni iniziali e il traguardo del tracciato.

  - `int getTipologiaLocazione(Locazione posizione)`: ritorna il tipo di locazione (0 se è fuori strada, 1 se è la pista, 2 se è il traguardo, 3 se è il punto di partenza).

  - `void inserisciMacchina(Macchina macchina)`, `void spostaMacchina(Locazione prossimaPosizione, Macchina macchina)`: gestisce l'inserimento e lo spostamento delle macchine sul tracciato.

  - `int getNumeroMacchina(Macchina macchina)`: ritorna un numero identificativo per la macchina basato sul colore.

  - `boolean controlloPosizione(Locazione locazione) `: verifica se la posizione è occupata da un’altra macchina.

7. `ReaderCircuito`

Classe per leggere e creare il tracciato di gara da un file di configurazione.

- Metodi:
  - `int[][] creazioneCircuito(String file)`: legge il file e crea la matrice del tracciato.
  - `void stampaCircuito(int row, int col, int[][] circuito)`: stampa il circuito per verificare il risultato.

Funzionamento del Programma

I passaggi principali includono:

1. Lettura del tracciato di gara da un file di configurazione.
2. Creazione del tracciato e inizializzazione della gestione della gara.
3. Creazione di macchine e giocatori (un utente e due bot).
4. Ciclo di gioco che alterna i turni tra i giocatori e i bot.
5. Controllo delle condizioni di vittoria o sconfitta e gestione del ciclo di gioco di conseguenza.

Il gioco termina quando una delle macchine raggiunge il traguardo o tutte le macchine sono fuori gara.