Luca Paulon Matricola: 1187028

Documento di architettura software

A seguito delle richieste avanzate da parte dei committenti si è deciso di procedere con delle precise scelte progettuali.
La mappa si compone di tre parti strettamente collegate:

- Una lista che contiene i pezzi elencati all'interno del file, dopo che sono stati letti, senza tenere conto di alcuna relazione d'ordine tra di essi. Questa viene utilizzata per eseguire le elaborazioni in cui si ricercano le celle con il maggiore valore di attacco e di difesa di "giorno" e di "notte". Si è creato ad hoc un tipo Pezzo (classe astratta) che rispecchia tutte le caratteristiche comuni di ogni pezzo. Come da richiesta dei committenti, si è poi creata un'apposita classe per ogni particolare pezzo: Elfo, Nano e Orco. In queste classi sono poi stati definiti degli appositi metodi necessari alla modifica dei parametri di attacco e difesa da utilizzare quando il pezzo si trova su una certa cella con determinate caratteristiche tipologiche oppure quando la mappa è in una certa fascia temporale. L'assegnazione di una fascia temporale alla mappa avviene solo in fase di elaborazione dei dati, inizialmente non è definita;
- Una matrice di celle. Si è creato uno specifico tipo Cella che rispecchia tutte le caratteristiche richieste, ovvero la possibilità di conteggiare ogni differente tipologia di pezzo con l'uso di tre differenti contatori, per ricercare poi il pezzo maggiormente presente sulla cella e un contatore generale utilizzato per testare il possibile raggiungimento della capienza massima della cella. Nel caso in cui questo accada, la cella stessa segnalerà l'errore attraverso un'apposita eccezione;
- Sei contatori: tre di essi servono per conteggiare rispettivamente quanti pezzi di ogni tipologia appartengono alla mappa, gli altri tre invece servono per conteggiare quanti pezzi, sempre divisi per tipologia, non appartengono alla mappa, cioè quanti pezzi hanno un tipo valido ma le cui coordinate sono eccessivamente grandi per appartenere alla mappa creata;

La gestione del dimensionamento della mappa è invece delegata alla classe Principale (in cui è contenuto il metodo main). Si controlla che i valori inseriti siano prima di tutto positivi, ma devono poi rimanere nelle dimensioni massime previste (vedere documento dei requisiti alla voce requisiti non funzionali). Sempre in questa classe vengono gestite tutte le possibili tipologie di errore che si possono presentare durante l'esecuzione: superamento della dimensione massima della cella, formato del file errato, file non presente, errore nell'input (quando s'inserisco dei caratteri al posto di valori numerici interi) e tipo errato (quando il formato del file è corretto ma nella lettura si legge un pezzo il cui tipo non è tra quelli predefiniti).

Si presentano di seguito le classi e gli enumeratori che compongono il software precisando per ognuno di essi: nome, dati membro, costruttori, funzioni membro e valori nel caso degli enumeratori.

Classi:

Principale	
Dati membro	
Costruttori	
Funzioni	
public static void main(String args[])	Punto di inizio del programma. La gestione della lettura da tastiera dei valori M e N viene fatta nel metodo main. Si richiede il reinserimento dei valori M ed N fino a quando non si inseriscono dei valori validi.
public static boolean verificaDimensioneMappa(int m, int n)	Verifica se la mappa rientra nelle dimensioni valide richieste.

Mappa	
Dati membro	
private int m	Contiene il numero di righe della mappa.
private int n	Contiene il numero di colonne della mappa.
private Cella[][] mappa	Matrice contenente le cella della mappa.
private List<Pezzo> pezzi	Lista contenente solo i pezzi effettivamente presenti. Si utilizza per le elaborazioni dei dati.
private int countOrchi	Conteggia il totale di orchi presenti.
private int countElfi	Conteggia il totale di elfi presenti.
private int countNani	Conteggia il totale di nani presenti.
private int countNotInOrchi	Conteggia gli orchi con coordinate fuori mappa.
private int countNotInElfi	Conteggia gli elfi con coordinate fuori mappa.
private int countNotInNani	Conteggia i nani con coordinate fuori mappa.
Costruttori	
public Mappa(Int M, Int N)	Inizializza una mappa con i valori M e N (validi e già controllati) letti da standard input. Crea la matrice di Celle inizializzando ogni cella ad un determinato tipo.
Funzioni membro	
public void init()	Apre il file, legge i pezzi presenti, li dispone sulla matrice che rappresenta la mappa e nella lista dei pezzi. Se il formato del file è errato si termina il programma lanciando l'eccezione NumberFormatException. Se il pezzo non appartiene alla mappa ma il tipo è valido conteggia i pezzi non appartenenti alla mappa. Se il tipo non corrisponde a quelli inseriti si notifica un errore nel file con l'eccezione InvalidTypeException e si termina il programma.

	Se si supera la capienza massima di una cella pari a 5 pezzi si lancia l'eccezione MaximumCellSizeException, e si termina il programma.
public void printNumeroPezzi()	Stampa sullo schermo il totale dei pezzi presenti sulla mappa divisi per tipologia, ed il totale dei pezzi fuori mappa divisi per tipologia.
public void maggioreDifesa(Temporalita temporalita)	Stampa la cella o le celle con maggiore valore di difesa quando il riferimento temporale è "giorno" oppure "notte".
public void maggioreAttacco(Temporalita temporalita)	Stampa la cella o le celle con maggiore valore attacco quando il riferimento temporale è "giorno" o "notte".
public void maggioreNumeroPezziStessoTipo()	Stampa la cella o le celle in cui c'è il maggior numero di pezzi tutti dello stesso tipo.
private void inizializzaCelle()	Inizializza ogni singola cella della matrice tramite l'invocazione del costruttore.
private Pezzo gestisciOrco(Pezzo cur, Temporalita temporalita)	Metodo di utilità utilizzato in fase di elaborazione per creare un clone di un orco nel caso in cui si debbano modificare i valori di attacco e difesa dell'orco senza modificare il dato della lista.
private Boolean giaInserita(List<Cella> celle, Cella c)	Verifica che in fase di registrazione di molteplici celle che soddisfano una stessa proprietà la cella corrente non sia già presente. In tal caso non verrà considerata.

Cella	
Dati membro	
private int x	Contiene la coordinata X della cella.
private int y	Contiene la coordinata Y della cella.
private TipoCella luogo{ pianura, bosco, montagna}	Enumeratore che contiene la tipologia della cella.
private int count	Conteggia il totale dei pezzi sulla cella.
private int countOrchi	Conteggia il numero di orchi presenti sulla cella.
private int countElfi	Conteggia il totale di elfi presenti sulla cella.
private int countNani	Conteggia il totale di nani presenti sulla cella.
Costruttori	
public Cella(Int X, Int Y)	Crea una cella di coordinate X, Y. Inizializza tutti i contatori ed assegna in tipo casuale alla cella
Funzioni membro	
public void addPezzo(Pezzo p)	Incrementa il contatore generale e quello relativo al tipo inserito. Se si supera la dimensione massima si lancia l'eccezione MaximumCellSizeException e si termina il programma.

public int findMaxType()	Ritorna il massimo numero di pezzi tra quelli dello stesso tipo presenti su una cella. In caso di parità non fa differenze.
private void initCella()	Inizializza la tipologia della cella attraverso un generatore di numeri casuali rispettando l'associazione definita nel documento dei requisiti.
public TipoCella getLuogo()	Ritorna la tipologia della cella.
public int getX()	Ritorna la coordinata X della cella.
public int getY()	Ritorna la coordinata Y della cella.
public boolean isEmpty()	Ritorna vero se la cella è vuota, cioè se non ci sono pezzi posizionati su di essa.
public boolean equals(Cella cella)	Override del metodo equals. Due celle sono uguali se hanno la stessa coordinata X e la stessa coordinata Y.

abstract Pezzo	
Dati membro	
protected int x	Contiene la coordinata X del pezzo.
protected int y	Contiene la coordinata Y del pezzo.
protected double attacco	Contiene il valore dell'attacco del pezzo.
protected double difesa	Contiene il valore della difesa del pezzo.
Costruttori	
protected Pezzo(Int X, Int Y, double a, double d)	Crea un pezzo nella cella di coordinate X, Y con i valori di attacco pari ad "a" di difesa pari a "d".
public int getX()	Ritorna il valore della coordinata X.
public int getY()	Ritorna il valore della coordinata Y.
public double getAttacco()	Ritorna il valore di attacco del pezzo.
public double getDifesa()	Ritorna il valore della difesa del pezzo.

Elfo	
Dati membro	
Dati ereditati dalla classe Pezzo.	
private static final double defAtt	Costante che contiene il valore di default dell'attacco.
private static final double defDif	Costante che contiene il valore di default della difesa.
Costruttori	
public Elfo(Int X, Int Y)	Crea un nuovo Elfo i cui valori di attacco e difesa sono quelli definiti di default. Richiama il costruttore della classe Pezzo passando i valori di attacco e difesa di default e le coordinate X, Y.
Funzioni membro	
Funzioni ereditate dalla classe pezzo.	

public void modificaAttDifBosco()	Modifica il valori di attacco e difesa se è posizionato su una cella dove la tipologia è bosco.

Nano	
Dati membro	
Dati ereditati dalla classe Pezzo.	
private static final double defAtt	Costante che contiene il valore di default dell'attacco.
private static final double defDif	Costante che contiene il valore di default della difesa.
Costruttori	
public Nano(Int X, Int Y)	Crea un nuovo Nano i cui valori di attacco e difesa sono quelli definiti di default. Richiama il costruttore della classe Pezzo passando i valori di attacco e difesa di default e le coordinate X, Y.
Funzioni membro	
Funzioni ereditate dalla classe pezzo.	
public void modificaAttDifMontagna()	Modifica il valori di attacco e difesa se è posizionato su una cella dove la tipologia è montagna.

Orco	
Dati membro	
Dati ereditati dalla classe Pezzo.	
private static final double defAtt	Costante che contiene il valore di default dell'attacco.
private static final double defDif	Costante che contiene il valore di default della difesa.
Costruttori	
Orco(Int X, Int Y)	Crea un nuovo Orco i cui valori di attacco e difesa sono quelli definiti di default. Richiama il costruttore della classe Pezzo passando i valori di attacco e difesa di default e le coordinate X, Y.
Funzioni membro	
Funzioni ereditate dalla classe pezzo.	
void decrementaAttDifGiorno()	Modifica il valori di attacco e difesa se il rifermento temporale è "giorno".
void incrementaAttDifNotte()	Modifica i valori di attacco e difesa se il riferimento temporale è "notte".

InvalidTypeException
Dati membro
Dati ereditati dalla classe RuntimeException.
Costruttori

Costruttori ereditati dalla classe RuntimeException.
Funzioni membro
Funzioni ereditate dalla classe RuntimeException.

MaximumCellSizeException
Dati membro
Dati ereditati dalla classe RuntimeException.
Costruttori
Costruttori ereditati dalla classe RuntimeException
Funzioni membro
Funzioni ereditate dalla classe RuntimeException.

InvalidFileFormatException
Dati membro
Dati ereditati dalla classe RuntimeException.
Costruttori
Costruttori ereditati dalla classe RuntimeException
Funzioni membro
Funzioni ereditate dalla classe RuntimeException.

Enumeratori:

TipoCella
Valori
bosco, pianura, montagna

Temporalita
Valori
giorno, notte