**Specifica del software**

Si vuole realizzare un’applicazione desktop che rappresenti una variante di “*campo minato*”, classico videogioco a singolo giocatore per PC degli anni 90, in cui lo scopo del gioco è scoprire tutte le zone sicure del campo minato senza far esplodere le mine.

In particolare si vuole sviluppare una versione del gioco in grado di supportare due tipologie di campo minato: un **campo tradizionale** e un **campo hardcore**. Con il termine **campo tradizionale** si intende un classico campo minato in cui le mine sono generate successivamente la prima zona scoperta dal giocatore, mentre con il termine **campo hardcore** si intende un campo minato in cui le mine sono generate precedentemente la scoperta della prima zona.

L’applicazione da sviluppare si basa sulle seguenti caratteristiche e regole di gioco:

* Il campo di gioco consiste in un campo rettangolare (o quadrato) composto da molteplici zone quadrate con cui il giocatore può interagire cliccando su di esse con il tasto sinistro e destro del mouse.
* Il campo minato può avere delle dimensioni (lunghezza e altezza) minime di 8x8 e massime di 30x30, mentre per quanto riguarda le mine in esso contenute il loro numero può variare da un minimo di 10 a un massimo corrispondente all’80% delle zone costituenti il campo minato.
* Le zone del campo minato sono inizialmente tutte coperte, per poi essere progressivamente scoperte dal giocatore man mano che il gioco procede. Una volta che tutte le zone sicure sono state scoperte il giocatore vince la partita.
* Il giocatore può scoprire una zona cliccando su di essa con il tasto sinistro del mouse. Se la zona scoperta è minata si perde la partita, se invece è sicura viene visualizzato un numero (da 1 a 8) che indica la quantità di mine presenti attorno ad essa. Tale numero deve essere utilizzato dal giocatore per individuare le mine all’interno del campo minato e stabilire quali successive zone scoprire. Nell’eventualità che la zona sicura scoperta non abbia mine nelle sue immediate vicinanze il gioco scopre automaticamente le zone ad essa vicine fintanto che non vengono scoperte zone sicure che restituiscono un numero.
* Il giocatore può eventualmente anche contrassegnare (visivamente con una bandiera) una zona in cui crede sia presente una mina cliccando su di essa con il tasto destro del mouse. Fintanto che una zona è contrassegnata non può essere scoperta, di conseguenza premendo nuovamente il tasto destro su una zona già contrassegnata come minata questa viene “pulita” o riportata al suo stato originario.

L’applicazione deve inoltre prevedere quattro differenti modalità di partita, allo scopo di agevolare i neofiti e introdurre i più piccoli al gioco. Queste modalità fanno riferimento alle medesime regole di gioco e sono differenziate tra loro unicamente dal numero di tentativi disponibili, ossia dalla quantità di mine che è possibile far esplodere prima di perdere la partita. Nel dettaglio:

* la modalità **classica** prevede un singolo tentativo;
* la modalità **agevolata** prevede tre tentativi;
* la modalità **semplificata** prevede un numero di tentativi variabile (non inferiore a tre e corrispondente al 10% delle mine contenute nel campo minato);
* la modalità **sicura** non prevede un limite al numero di tentativi, impedendo di fatto di perdere la partita.

L’applicazione infine deve prevedere l’eventuale aggiunta in futuro di ulteriori tipologie di campo minato.

**Studio del problema**

*- Quali sono i punti critici?*

*- Come si sceglie di affrontarli?*

Dalla specifica è possibile individuare i seguenti punti critici:

* **Processo di costruzione di un campo minato**

Nonostante un campo minato tradizionale e uno hardcore si differenzino tra loro in come e quando le mine vengano generate, il processo di costruzione di un campo minato (che include al suo interno anche la fase di generazione delle mine) è uguale per tutte le tipologie di campo minato.

Di conseguenza, si ritiene opportuno implementare il design pattern **Template Method**, definendo lo scheletro dell’algoritmo di costruzione di un campo minato all’interno della classe base (rappresentante un generico campo minato) e lasciando alle sottoclassi (campo minato tradizionale e hardcore) i dettagli relativi alla generazione delle mine.

* **Riconfigurazione di un campo minato e delle sue zone**

Per evitare ad ogni nuova partita di istanziare un nuovo campo minato (della medesima tipologia) e potenzialmente fino a 900 zone si reputa ideale:

* fare in modo che il campo minato possa essere riconfigurato nelle sue caratteristiche (lunghezza, altezza e numero di mine) anche a seguito della sua creazione;
* creare fin da subito un campo minato di dimensioni massime, ma consentendo all’esterno di poter accedere unicamente alle zone che rientrano nelle dimensioni impostate per il campo minato;
* fare in modo che le zone del campo minato possano essere anch’esse riconfigurate successivamente la loro creazione.
* **Rappresentazione delle modalità di gioco**

Tenendo in considerazione che le varie modalità di gioco fanno riferimento alle medesime regole, che l’unica differenza tra loro è costituita dal numero di tentativi ad esse associate, e che il numero di tentativi della modalità *semplificata* è determinato dal numero di mine presenti nel campo minato, si è deciso di rappresentare questo concetto sotto forma di tipo enumerato, definito all’interno della classe che rappresenta le regole di una partita a campo minato.

* **Possibilità di aggiungere nuove tipologie di campo minato in futuro**

Vista la possibile aggiunta di nuovi tipi di campo minato è auspicabile che l’interfaccia grafica non sia né strettamente accoppiata a tipi specifici di campo minato né si occupi direttamente di istanziarne uno.

Di conseguenza, si reputa opportuno utilizzare il design pattern **Factory Method**, in grado di fornire un’interfaccia apposita per la creazione di oggetti, consentendo così di eliminare la dipendenza dell’interfaccia grafica dai tipi concreti di campo minato.

**Scelte architetturali**

*- Descrizione dell’architettura software comprensiva di una schema delle classi UML che descriva le componenti principali del sistema.*

*- Descrizione e motivazione dei design pattern utilizzati.*

**Documentazione sull’utilizzo**

*- (Se applicabile) Con quali parametri va eseguito il software una volta compilato?*

*- (Se applicabile) Ci sono passi particolari da eseguire per la compilazione?*

**Casi d’uso**

*- Descrivere i più significativi.*

*La relazione può includere altre sezioni qualora siano funzionali alla spiegazione.*

*La relazione non deve includere l’intero listato del codice ma può contenere alcune sezioni del programma se necessario per la spiegazione.*

*Il documento deve essere quanto più sintetico possibile e non deve superare le 12 pagine.*