Contratto con gli studenti

Il compito consiste nel realizzare un progetto che è costituito da una applicazione standalone realizzata in Java. Il progetto può essere realizzato da un gruppo composto da un minimo di 1 studente ad un massimo di 3 studenti. In generale, uno deve essere l'eccezione non la regola. Potete scegliere il progetto tra le tre possibili idee progettuali previste nel documento. Il progetto avrà un impatto del 50% sul voto totale.

Il restante 50% della valutazione consisterà in un colloquio orale singolo volto ad accertare le conoscenze acquisite durante il corso. Inoltre, nel colloquio ci sarà anche una discussione del progetto.

NOTA: Il progetto deve essere consegnato **5 giorni prima dell'appello** inviando una e-mail al docente manifestando la volontà di sostenere l'esame. Pertanto, il docente considererà il progetto sviluppato entro tale scadenza. Ciò vuol dire che qualunque modifica apportata successivamente non verrà presa in considerazione. Chiaramente bisogna effettuare la registrazione anche nel sistema di segreteria virtuale.

Form

Per la composizione dei gruppi è necessario compilare la Microsoft form disponibile nel canale teams.
La compilazione della form deve avvenire da parte di uno dei componenti del gruppo. Vi chiediamo cortesemente di compilare la form con molta attenzione.
Prima di compilare la form, tutti i componenti del gruppo devono creare un account su GitHub. Vi consigliamo di creare l'account con un nome che vi identifichi. GitHub sarà un vostro "biglietto" da visita virtuale per il vostro futuro accademico e lavorativo. La compilazione di tale form è obbligatoria.

Assignment

Una volta compilata la form si deve accedere al servizio classroom di github (https://classroom.github.com). Il docente all'interno di una classroom (https://classroom.github.com/classrooms/161450431-lpodisim2024-classroom-progetto) ha creato dei cosiddetti assignment (uno per ogni progetto). Di seguito sono elencati i link degli assignment per ogni progetto.
gioco-oca-univaq: https://classroom.github.com/a/2oBsXp76
risiko-univaq: https://classroom.github.com/a/OyE7CdI5
monopoli-univaq: https://classroom.github.com/a/LPLP_Uc8
I singoli componenti del team si devono iscrivere al relativo assignment/progetto scelto. Vi ricordiamo che uno dei componenti deve creare il team e, successivamente, gli altri componenti devono fare Join nel team.

Repository GitHub

Una volta che il team è stato creato, in GitHub troverete un repository per team con una struttura del progetto predefinita. Lo scheletro del progetto utilizza Apache Maven che chiaramente deve essere utilizzato per il progetto. Pertanto, tale repository dovrà essere utilizzata per gestire il codice sorgente del progetto. La repository conterrà il sistema da consegnare e di conseguenza permetterà di valutare il progetto durante l'esame. Inoltre, la repository conterrà una cartella doc dove potete trovare il template del progetto documentazione-progetto.docx da utilizzare per documentare il vostro progetto. Vi ricordo che tale documento è obbligatorio e costituisce parte integrante del progetto. Il repository deve

essere utilizzato da tutti i componenti del gruppo in modo uniforme durante lo sviluppo del progetto. Pertanto, ogni singolo componente deve effettuare un numero congruo di commit rispetto agli altri componenti del gruppo. Inoltre, i files contenuti in ogni singolo commit devono essere congrui rispetto agli altri.
NOTA importante: Qualora decidiate di abbandonare il gruppo dovete comunicarlo al docente inviando una e-mail con in copia carbone (cc) gli altri componenti del gruppo. Chiaramente per il/la rinunciatario/a dovrete ripetere il processo da capo, ovvero compilare la form e iscriversi al relativo assignment.

Regole comuni a tutti i progetti

- Il codice deve essere consegnato 5 giorni prima della data di esame, ad eccezione di quei gruppi che intendono sostenere l'esame durante il primo appello;
- Il codice deve essere compilabile ed eseguibile tramite comandi maven;
- Il codice deve essere sviluppato in maniera collaborativa usando git come VCS (Notabene: si analizzeranno il numero di commit e l'autore dello stesso)
- Il codice deve essere sviluppato in modo bilanciato dagli elementi del gruppo;
- Il progetto deve prevedere il riutilizzo di una libreria nota (ad esempio quelle fornite da apache foundation)

Analisi e Progettazione

Prima di iniziare la codifica, ogni gruppo dovrà sottomettere una breve analisi del gioco scelto, identificando le principali entità (classi) e le loro relazioni. Dovranno inoltre delineare l'architettura generale del sistema e le principali funzionalità.

Implementazione

Il gioco dovrà essere implementato seguendo i principi della programmazione orientata agli oggetti. È importante che il codice sia ben organizzato, commentato e conforme agli standard di stile del linguaggio scelto.

Documentazione

Ogni gruppo dovrà fornire una documentazione che include class-diagram, una descrizione delle principali classi e metodi, descrizione delle API (javadoc) e una riflessione sul design adottato.

- Riflessione sul Design: Analisi delle sfide incontrate durante lo sviluppo e delle soluzioni adottate.
- Design ad Oggetti: Utilizzo efficace della programmazione ad oggetti, inclusa l'ereditarietà e il polimorfismo.
- Javadoc per le API

Valutazione

Il progetto sarà valutato sulla base dei seguenti criteri:

- Qualità del Design: Chiarezza e appropriatezza della struttura ad oggetti e dell'architettura del sistema.

- Implementazione: Correttezza, efficienza, e stile del codice.
- Usabilità: Facilità di utilizzo e qualità dell'interfaccia utente basata su console.
- Innovazione e Creatività: Originalità delle soluzioni adottate e delle funzionalità implementate.
- Documentazione e Presentazione: Completezza e chiarezza della documentazione e efficacia della presentazione finale.
- Gestione delle eccezioni generate durante il gioco (ad esempi imput erronei e mosse non consentite).

Consigli per gli Studenti

- Iniziate presto: Non sottovalutate il tempo necessario per analizzare, progettare e implementare il vostro gioco.
- Collaborate: Comunicate costantemente all'interno del vostro gruppo e condividete le responsabilità in modo equo.
- Sperimentate: Non abbiate paura di esplorare nuove idee o di deviare leggermente dalle regole del gioco originale per rendere il vostro progetto più interessante o gestibile.
- Cercate feedback: Non esitate a chiedere feedback al vostro insegnante o ai vostri compagni durante lo sviluppo del progetto.

Descrizione risiko-univaq

Gli studenti potrenno effettuare delle personalizzazioni alle regole del gioco Risiko. Tali personalizzazioni dovranno:

- Essere concordate con il docente;
- Essere documentate nella relazione finale.

Obiettivo

- L'obiettivo di questo progetto è progettare e implementare il gioco da tavolo Risiko, utilizzando concetti di programmazione ad oggetti come classi, ereditarietà, polimorfismo, e gestione degli eventi. L'interfaccia utente sarà basata su console, ponendo enfasi sull'usabilità e sull'interazione utente.
- Linguaggio di Programmazione: Java (o altro linguaggio orientato agli oggetti concordato).
- Interfaccia Utente: Basata su console e/o java-fx,
- Numero di Giocatori: Da 2 a 6 giocatori.

Analisi e Progettazione

Entità Principali e Classi:

- Mappa: Gestisce le varie territori e continenti, inclusi i confini e le connessioni tra territori.
- Giocatore: Informazioni sul giocatore, incluse le armate a disposizione e i territori controllati.
- Territori: territori sulla mappa
- Continenti: I continenti raggruppano i territori
- Carte: Implementazione delle carte territorio, obiettivo, e jolly.

- Dado: Gestione dei lanci di dado per gli attacchi e le difese.
- Obiettivo: il raggiungimento di un obiettivo determina la vittoria della partita
- Stato del Gioco: Controllo delle fasi del gioco (distribuzione delle armate, attacco, spostamento, ecc.).

Relazioni e Interazioni

- Interazioni tra giocatori per attacchi e spostamenti di armate.
- Gestione delle fasi del gioco e delle azioni dei giocatori.

Logica di Gioco e requisiti

- Regole per la distribuzione iniziale dei territori e delle armate.
- Meccanismo per le fasi di attacco, spostamento e rinforzo.
- Calcolo dei rinforzi basato sul controllo dei territori e dei continenti, oltre al possesso di carte.
- Interazioni Utente:
- Input dell'utente per la selezione dei territori, l'esecuzione degli attacchi, e lo spostamento delle armate (opzionale).
- Visualizzazione dello stato della mappa, delle armate, e delle carte possedute.
- Il progetto deve permettere di salvare e riprendere una partita.
- Il progetto deve fa uso di **Collections**. Inoltre è fortemente consigliato l'uso di stream ed espressioni lambda sulle liste.
- i pogetti devono scrivere su un file la lista delle azioni svolte (e.g., Giocatore X attacca Giocetore Y territorio X: 6, Giocatore X sceglie 3 cararmati, etc)

Regole del risiko

http://risiko.it/wp-content/uploads/2017/10/Regolamento-Risiko.pdf

Descrizione gioco-oca-univaq

Gli studenti potrenno effettuare delle personalizzazioni alle regole del gioco proposto. Tali personalizzazioni dovranno:

- Essere concordate con il docente;
- Essere documentate nella relazione finale.

Obiettivo

- L'obiettivo di questo progetto è progettare e implementare il gioco da tavolo gioco dell'oca con alcuni modifiche al regolamento, utilizzando concetti di programmazione ad oggetti come classi, ereditarietà, polimorfismo, e gestione degli eventi. L'interfaccia utente sarà basata su console, ponendo enfasi sull'usabilità e sull'interazione utente.
- Linguaggio di Programmazione: Java (o altro linguaggio orientato agli oggetti concordato).
- Interfaccia Utente: Basata su console e/o java-fx,
- Numero di Giocatori: Da 2 a 6 giocatori. Il gioco deve prevedere giocatori non umani.

Analisi e Progettazione

- Gioco: Classe principale che gestisce le regole del gioco, il flusso di gioco, e le interazioni tra le altre classi.
- Tabellone: Rappresenta il tabellone del gioco, composto da caselle. Ogni casella può avere un effetto diverso sul giocatore che vi si ferma. Il tabellone e il numero di caselle deve essere scelto al momento di inizio della partita. Così come il numero di facce del dado.
- Giocatore: Gestisce le informazioni del giocatore, come la posizione attuale sul tabellone e il turno di gioco.
- Casella: Classe astratta da cui derivano diverse tipologie di caselle (normale, oca, ponte, labirinto, prigione, ecc.). Esistono caselle speciali che abilitano mini game.

Logica di Gioco e requisiti

- Meccanismo per le fasi di minigame
- Interazioni Utente
- Visualizzazione dello stato del tabellone.
- Il progetto deve permettere di salvare e riprendere una partita.
- Il progetto deve fa uso di **Collections**. Inoltre è fortemente consigliato l'uso di stream ed espressioni lambda sulle liste.
- i pogetti devono scrivere su un file la lista delle azioni svolte (e.g., Giocatore X attacca Giocetore Y territorio X: 6, Giocatore X sceglie 3 cararmati, etc)

Regole speciali

Il dato in questo gioco ha **M facce (valore deve essere preso da un file di properties).**
Si gioca su un tabellone sul quale è disegnato un percorso a spirale **composto da N caselle (valore deve essere preso da un file di properties)**. Visto che il numero di caselle deve variare, anche la distribuzione delle caselle normali e speciali e quelle dei mini game saranno assegnate randomicamente sul tabellone.

A questo tradiozionale gioco si voglio aggiungere 2 tipologie di caselle speciali che dovranno essere piazziate durante il persorso (più di una per ogni tipologia).
Tipologia - Gioco della morra:
Passa ultimo: questa casella obbligerà il passante a combattere con l'ultimo della scacchiera al gioco della morra (carta, sasso, forbice) alla meglio di 3. In caso di pareggi i posti rimarranno inalterati. Se il giocatore che passa su questa casella è l'ultimo combatte con il primo. La gestione della segretezza del simbolo scelto si demanda al buon senso dei giocatori.
Tipologia - Tris:
questa casella obbligerà il passante a combattere con il primo della scacchiera a tris alla meglio di 3. In caso di pareggi le posizioni sulla scacchiera rimarrano inalterate. Se il giocatore che passa su questa casella è il primo combatte con l'ultimo.

Le regole del Gioco dell'oca

Il gioco dell'oca è adatto a tutti, a partire dai sei anni circa. Le regole sono abbastanza semplici. Ecco, in sintesi, quali sono.
Il dato in questo gioco ha **M facce (valore deve essere preso da un file di properties).**
Si gioca su un tabellone sul quale è disegnato un percorso a spirale **composto da N caselle (valore deve essere preso da un file di properties)**. Le caselle sono contrassegnate con numeri o altri simboli. I giocatori devono scegliere una pedina e, dopo aver pagato una posta pattuita, partono dalla casella di partenza. Ogni giocatore deve procedere aspettando il proprio turno, percorrendo il numero di caselle ottenuto tramite il lancio dei dadi.
Lo scopo finale del gioco è quello di raggiungere la casella centrale della spirale. Se la casella di arrivo è occupata dalla pedina di un altro giocatore, la pedina di arrivo ne prende il posto.
È necessario sapere che alcune caselle hanno un "valore" speciale. Nel gioco tradizionale, infatti, le caselle decorate con delle oche permettono al giocatore di spostarsi subito in avanti di un numero di caselle pari a quelle coperte dal movimento appena effettuato.

Le altre caselle "speciali"

Ci sono anche altre caselle nel Gioco dell'oca che possono essere definite "speciali", ecco quali:
il ponte, dove si paga la posta e si va alla casella 12;
casa" o locanda, impone il pagamento della posta e si rimane fermi per un turno;
pozzo e prigione impongono al giocatore di rimanere fermo fino a quando non arriva alla casella interessata un'altra pedina, che viene a sua volta "imprigionata";
labirinto, costringe il giocatore a pagare la posta e a tornare alla casella X;
scheletro, impone di pagare la posta e di ricominciare, dalla casella di partenza.
La casella d'arrivo **N** deve essere raggiunta con un lancio di dadi esatto; altrimenti, giunti in fondo, si retrocede dei punti in eccesso.

Descrizione monopoli-univaq

Gli studenti potrenno effettuare delle personalizzazioni alle regole del gioco. Tali personalizzazioni dovranno:

- Essere concordate con il docente;
- Essere documentate nella relazione finale.

Obiettivo

- L'obiettivo di questo progetto è progettare e implementare il gioco da tavolo monopoli con alcuni modifiche al regolamento, utilizzando concetti di programmazione ad oggetti come classi, ereditarietà, polimorfismo, e gestione degli eventi. L'interfaccia utente sarà basata su console, ponendo enfasi sull'usabilità e sull'interazione utente.
- Linguaggio di Programmazione: Java (o altro linguaggio orientato agli oggetti concordato).
- Interfaccia Utente: Basata su console e/o java-fx,

- Numero di Giocatori: Da 2 a 6 giocatori. Il gioco deve prevedere giocatori non umani.

Analisi e Progettazione

- Gioco: Classe centrale che coordina il corso del gioco, le regole e le interazioni tra le altre classi.
- Tabellone: Rappresenta il tabellone di gioco, organizzato in caselle che includono proprietà, imprevisti, probabilità, tasse, stazioni, e servizi. Il tabellone dovrà essere dinamico e prevedere diverse settaggi in base al numero di giocatori. Ad esempio 2 giocatori 4 terzine di proprietà 2 stazioni, 6 giocatori 10 terzine di proprietà, 6 stazioni/servizi). Il punteggio delle somme dovute da più stazioni/servizi potete deciderlo voi.
- Giocatore: Gestisce le informazioni e le azioni di ogni giocatore, come la posizione sul tabellone, denaro, proprietà possedute e decisioni di gioco. Considerare la presenza di giocatori gestiti dal pc.
- Casella: Classe astratta che può esser ereditata per creare diversi tipi di caselle (proprietà, tassa, probabilità, ecc.).
- Carta: Classe per le carte probabilità e imprevisti, che influenzano il gioco in modi variabili.

Logica di Gioco e requisiti

- Meccanismo per le fasi di minigame
- Interazioni Utente
- Visualizzazione dello stato del tabellone.
- Il progetto deve permettere di salvare e riprendere una partita.
- Il progetto deve fa uso di **Collections**. Inoltre è fortemente consigliato l'uso di stream ed espressioni lambda sulle liste.
- i pogetti devono scrivere su un file la lista delle azioni svolte (e.g., Giocatore X attacca Giocetore Y territorio X: 6, Giocatore X sceglie 3 cararmati, etc)

Regolamento del gioco

Se cercate online riuscite facilmente a trovare le regole del gioco. Altrimenti chiedere al docente.