Progetti Finali per il Corso di Metodologie di Programmazione: specifiche, regole e linee guida

Premessa

Attualmente, l'esposizione dei progetti si è conclusa con la loro descrizione, lasciando a Mercoledì 5 Maggio l'ultima parte relativa alle regole e alle linee guida. Nel frattempo, è consigliata la lettura approfondita di quanto riportato nelle righe successive per comprendere al meglio i progetti e i suoi criteri di valutazione, utilizzando Classroom per segnalare bug, errori, dubbi e domande attinenti.

Introduzione

Il seguente documento ha lo scopo di fornire tutte le informazioni relative allo sviluppo dei progetti. Le informazioni qui riportate riguarderanno tutti gli aspetti inerenti al ciclo di vita del software, partendo dalle specifiche fino ad arrivare alle modalità di consegna (*rilascio*).

Specifiche dei Progetti

A seguire, verranno mostrate le specifiche di sviluppo per ciascun progetto, la cui scelta dipenderà dal numero di componenti nel gruppo.

1) Forza 4 con visualizzazione su console (singolo) o tramite GUI (gruppo da 2)

Il gioco Forza 4 consiste in una sfida tra due giocatori secondo le seguenti regole:

1. Due giocatori operano su una griglia, disposta verticalmente e chiusa sul fondo, che ha dimensione di 6x7 (7 in verticale e 6 in orizzontale) caselle
2. I due giocatori dispongono di pedine diverse tra loro
3. I due giocatori a turno inseriscono una pedina in una delle colonne verticali della griglia: la pedina scende fino al fondo della colonna oppure fino ad appoggiarsi all'ultima pedina precedentemente inserita nella stessa colonna. In questo modo la pedina va ad occupare la casella più bassa della colonna, se la colonna era vuota, oppure la casella libera immediatamente superiore

alle caselle già occupate in precedenza. È vietato inserire pedine in una colonna che sia completamente occupata.

4. Lo scopo dei giocatori è allineare in orizzontale, in verticale o in diagonale, quattro pedine dello stesso colore: la partita termina quando uno dei due giocatori raggiunge tale obiettivo o quando la griglia è completamente piena.

   Il programma dovrà richiedere i nomi dei due giocatori, assegnandogli due diversi colori. Successivamente, la partita verrà avviata e il funzionamento sarà regolato dai dei turni, che verranno alternati tra i due giocatori. In ogni turno infatti verrà richiesta la mossa al giocatore attivo, **visualizzando conseguentemente lo stato della griglia a video.** Il programma dovrà impedire qualsiasi mossa vietata e rileverà automaticamente quando uno dei due utenti avrà raggiunto la vittoria, segnalandolo opportunamente, oppure se la partita si sarà conclusa in parità. Si dovrà offrire inoltre la possibilità di sospendere la partita, attribuendole un nome e salvandone lo stato attuale. Coerentemente, il programma dovrà dare la possibilità di riprendere l'esecuzione di una partita precedentemente salvata a partire dal punto in cui era stata sospesa.

2) Battaglia Navale con visualizzazione su console (singolo) o tramite GUI (gruppo da 2)

Da *Wikipedia*:
La battaglia navale è un gioco, originariamente nato per carta e penna, che vede due giocatori scontrarsi attraverso l'abbattimento completo della flotta nemica. Per giocare, sono necessarie quattro tabelle (due per giocatore) di egual dimensione. I quadretti di ciascuna tabella sono identificati da coppie di coordinate, corrispondenti a riga e colonna; tradizionalmente si usano lettere per le colonne e numeri per le righe (perciò le celle sono "A-1", "B-6", e così via). All'inizio, i giocatori devono "posizionare le proprie navi" segnandole su una delle loro due griglie (che terranno nascoste all'avversario per tutta la durata del gioco).
Una "nave" occupa un certo numero di quadretti adiacenti in linea retta (orizzontale o verticale) sulla tabella. Due navi non possono toccarsi. I giocatori si accordano preliminarmente su quante navi disporre e di quali dimensioni. Si può notare che molti giocatori utilizzano (anche non sempre in modo consistente) una particolare terminologia per riferirsi alle navi delle varie dimensioni; per esempio un sottomarino è di solito una nave di dimensione 3, insieme all'incrociatore, un cacciatorpediniere è di dimensione 2 e le navi di lunghezza superiore sono corazzate (dimensione 4) e così via. Una volta posizionate le navi, il gioco procede a turni. Il giocatore di turno "spara un colpo" dichiarando un quadretto (per esempio, "B-5"). L'avversario controlla sulla propria griglia se quella cella è occupata da una sua nave. In caso affermativo risponde "colpito!" e marca quel quadretto sulla propria tabella; in caso negativo risponde "acqua" o "mancato". Sulla seconda tabella in dotazione i giocatori prendono nota dei colpi che hanno sparato e del loro esito. Quando un colpo centra l'ultimo quadretto di una nave non ancora affondata, il giocatore che subisce il colpo dovrà dichiarare "colpito e affondato!" e la nave si considera persa. Vince il giocatore che fa affondare tutte le navi dell'avversario per primo.

Partendo dalla sua definizione, si vuole implementare un programma in Java che permetta di giocare a battaglia navale contro un avversario rappresentato dal computer. Il campo da gioco di ciascun giocatore corrisponderà ad una griglia 10x10 come nella spiegazione del gioco. Ogni giocatore ha in dotazione una flotta di 10 navi, così composta:

- 1 Portaerei con 5 caselle
- 1 Corazzata con 4 caselle
- 2 Crociere con 3 caselle
- 3 Sottomarini con 3 caselle
- 3 Navi d'assalto con 2 caselle

Per convenzione, l'utente avrà sempre il primo turno della partita, mentre il computer seguirà al turno successivo.
All'inizio di una nuova partita, l'utente disporrà le proprie navi sul proprio campo di gioco mentre il calcolatore procederà allo schieramento automatica della propria flotta. Fatto ciò, la partita avrà inizio.

All'inizio di ogni turno dovrà essere visualizzata la griglia dell'utente, seguita da quella inerente agli attacchi effettuati. I due giocatori indicheranno una casella del campo di gioco avversario tramite una coppia composta da una lettera e un numero. Per esempio, (A, 2) indica la seconda casella da sinistra della prima riga del campo di gioco. Ai fini della selezione della casella, il computer selezionerà un punto in modo casuale tra la lista di coloro i quali non sono stati ancora selezionati. Se la casella indicata è occupata da una nave, essa verrà colpita, affondandola nel caso in cui tutte le celle appartenenti a tale nave siano state colpite. L'esito dell'attacco dovrà essere indicato con un opportuno messaggio, in linea con il tipo di visualizzazione grafica che dovrà essere applicata nel progetto.

Vince il giocatore che affonderà per primo l'intera flotta avversaria.

3) Gestore di un Ristorante (gruppi da 3)

Si vuole realizzare il gestore di un piccolo ristorante con lo scopo di digitalizzare la gestione delle ordinazioni e l'emissione degli scontrini. Gli utilizzatori di questo progetto appartengono a 4 differenti ruoli all'interno del ristorante:

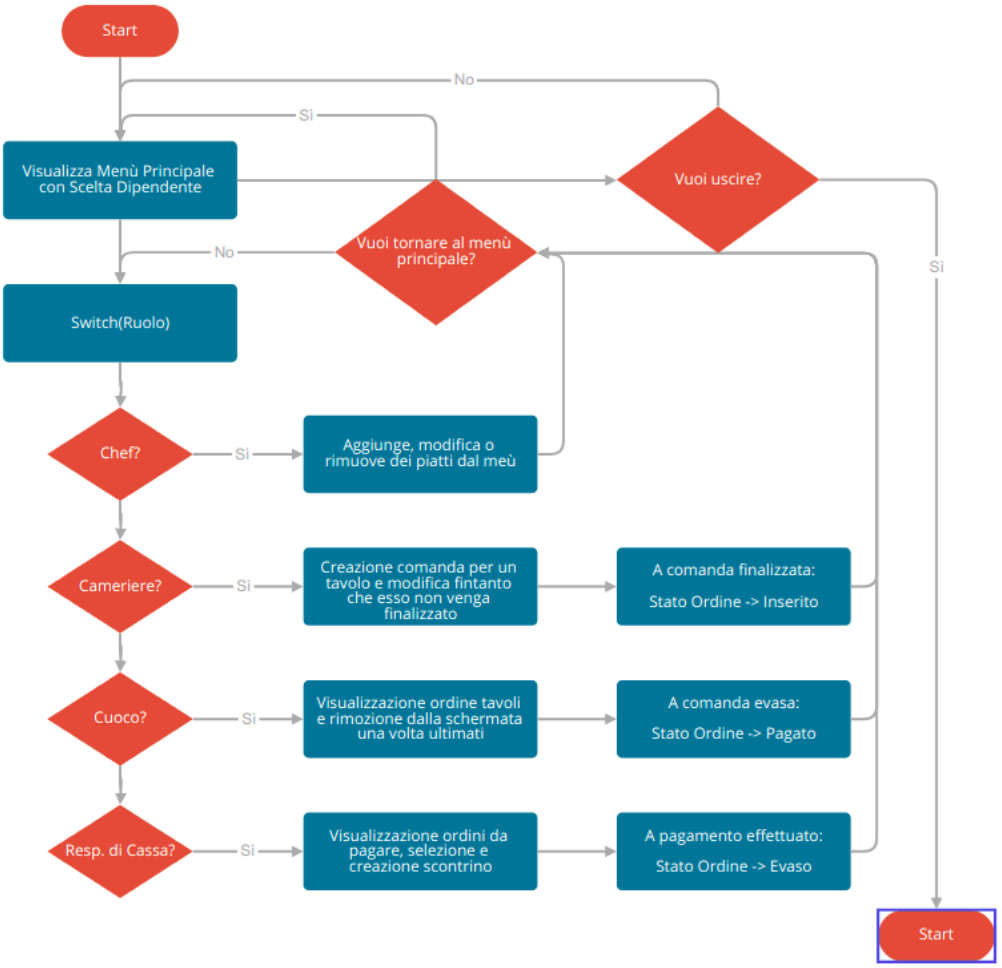
1) **Chef:** dovrà avere la possibilità di aggiungere, modificare o rimuovere i piatti all'interno del menù
2) **Cameriere**: dovrà avere la possibilità di prendere le ordinazioni dei clienti di ciascun tavolo, scegliendo tra i prodotti presenti in un menù
3) **Cuoco**: dovrà avere la possibilità di ricevere le ordinazioni provenienti dai camerieri per poi evaderle una volta che i piatti sono stati evasi

4) **Responsabile di cassa**: dovrà avere la possibilità di elargire uno scontrino sulla base delle ordinazioni effettuate da un tavolo

Il progetto dovrà soddisfare le necessità appena descritte. Da un punto di vista più tecnico, il programma dovrà:

- **Menù Principale:** visualizzare un menù iniziale, mostrato all'avvio del programma, che permetta sia l'accesso alle schermate relative ai diversi dipendenti che l'uscita dall'applicazione
- **Chef:** offrire la possibilità di aggiungere, modificare e rimuovere le pietanze all'interno di un menù, identificate da un *nome,* univoco, ed un *prezzo*. La lista delle pietanze all'interno di un menù dovrà essere persistita all'interno di un file, in modo tale che le successive esecuzioni del programma non debbano ricostruire nuovamente il menù
- **Cameriere:** gestire la creazione di un ordine attraverso una schermata in cui venga visualizzato il menù e nel quale, cliccando su una pietanza, venga aggiunta all'interno dell'ordine con la quantità desiderata. Poiché un cliente può cambiare idea durante l'ordinazione, la creazione di una comanda dovrà prevedere la modifica delle quantità e l'eventuale rimozione di un piatto. Ad ordine finalizzato non sarà più possibile effettuare una modifica
- **Cuoco:** fornire una schermata per la cucina in cui vengano visualizzati tutti gli ordini di ciascun tavolo, rimuovendo un ordine una volta che tutti i piatti al suo interno sono stati evasi
- **Responsabile di cassa:** fornire una schermata che permetta il pagamento del conto, selezionandolo da una lista di conti di tutti i tavoli che non hanno ancora effettuato il pagamento. In aggiunta a ciò, dovrà essere simulata la creazione di uno scontrino sotto forma di file di testo. A pagamento effettuato, l'ordine di quel tavolo non dovrà più essere visibile.
- All'interno della schermata di ciascun dipendente deve essere data la possibilità di tornare al menù principale

Ai fini di una maggior comprensione, a seguire viene fornito uno schema generale del ciclo di vita dell'applicazione:



4) Web Scraper Wikipedia sugli Imperatori Romani (gruppi da 4 o più)

Il seguente progetto verterà sulla creazione di uno o più alberi genealogici relativi a ciascuna dinastia di Imperatori Romani, con lo scopo di visualizzare le diverse relazioni parentali che intercorrono.

Poiché tale processo dovrà essere il più automatico possibile andando ad attingere da diverse fonti dati presenti su Wikipedia, ai fini della realizzazione dovrà essere impiegata la tecnica del web scraping. Essa sarà necessaria per l'ottenimento delle informazioni relative alle diverse interazioni familiari di ciascun membro delle dinastie. Un esempio di quanto appena introdotto è mostrato in questa immagine. Come si può notare, l'infografica rappresenta l'albero genealogico della dinastia Giulio-Claudia con le relazioni parentali che intercorrono tra i diversi membri. Il nome dei sovrani è descritto tramite una formattazione in grassetto del loro nome, insieme alla data di inizio e fine regno.

Da un punto di vista tecnico, il progetto richiederà l'utilizzo di Selenium, un framework open-source multipiattaforma impiegato per il testing applicazioni web e,

come nel nostro caso, per effettuare operazioni di web scraping. Per familiarizzare con il prodotto è consigliata la lettura di uno dei tanti tutorial presenti su internet. Ad esempio, è possibile trovare dei Tutorial per IntelliJ o per Eclipse.

A livello di specifiche, il progetto utilizzerà il web scraping e le interfacce grafiche al fine di:

- ottenere gli imperatori di ciascuna dinastia, partendo dalla pagina di Wikipedia fornita all'inizio della descrizione
- visitare la pagina di ogni imperatore di ciascuna dinastia al fine di prelevare le sue relazioni di parentela. Ad esempio, riferendosi alla pagina Wikipedia di Augusto:

Coniuge	Clodia Pulcra[26] (fino al 40 a.C.) Scribonia[26] (40–38 a.C.) Livia Drusilla[26] (38 a.C.-14 d.C.)
Figli	Giulia maggiore[29] Adottivi: Lucio Cesare[30] Gaio Cesare[30] Marco Vipsanio Agrippa Postumo (poi ripudiato e mandato in esilio)[31] Tiberio[31]
Dinastia	Giulio-claudia
Padre	Gaio Ottavio[27] Gaio Giulio Cesare (adottivo)
Madre	Azia maggiore[28]

- creare un albero genealogico per ciascuna dinastia mediante gli strumenti grafici visti a lezione

Come si può notare, l'estensibilità e la creatività per questo progetto sono dei punti lasciati volutamente aperti poiché, in un contesto di gruppi ben assortiti come questo, è possibile assegnare ad una o più persone una particolare area dello sviluppo (es: creazione dell'infrastruttura per il download dei dati, sviluppo dell'interfaccia grafica, etc.). Pertanto, è incoraggiata la personalizzazione del proprio operato fintanto che i requisiti di base precedentemente espressi siano soddisfatti.

Regole

Lo sviluppo e la conseguente valutazione del progetto verteranno sulle regole riportate qui a seguire. L'eventuale inadempienza a tali criteri comporterà delle penalità sulla valutazione finale o l'insufficienza del progetto stesso dove espressamente indicato.

Correttezza

Il codice sorgente all'interno del progetto dovrà compilare ed essere eseguito correttamente secondo le specifiche riportate nella sua descrizione. *Il criterio di compilazione è una condizione necessaria ai fini della sua valutazione.*

Poiché ciascun progetto verrà testato sia nella sua correttezza che nella gestione delle casistiche non previste dal normale funzionamento, è richiesta la predisposizione di una o più classi che fungano da *tester* per le funzionalità sviluppate.

Applicazione dell'Object Oriented Programming (OOP)

Il progetto dovrà essere sviluppato rispettando l'OOP affrontato nei suoi vari aspetti durante il corso. A tale scopo, la valutazione verterà sui seguenti punti:

- Utilizzo della Modularità, specialmente nel caso di progetti con interfacce grafiche in cui dovrà essere applicata una separazione tra la parte grafica e quella logica
- Utilizzo dell'Incapsulamento
- Utilizzo dell'ereditarietà
- Utilizzo del polimorfismo volto ad incentivare l'estensibilità delle classi.

È consigliata l'attinenza ai principi SOLID introdotti come durante il corso. Essa dovrà essere giustificata all'interno della relazione spiegando opportunamente quali principi sono stati rispecchiati da ciascuna classe.

Funzionalità Riportate

Le funzionalità richieste all'interno del ciascun progetto dovranno essere sviluppate secondo le specifiche riportate. La valutazione di questa sezione verterà quindi sulla quantità e sulla qualità di funzioni sviluppate.

Documentazione del Codice

La documentazione del codice dovrà rispettare gli standard introdotti durante il corso. In particolar modo, il progetto dovrà essere corredato dalla corrispettiva documentazione *JavaDoc* per ogni classe presente al suo interno. Verrà inoltre richiesta la presenza di variabili il cui nome sia *parlante* e la scrittura di attributi e metodi secondo la corretta *naming convention* utilizzata in Java.

Qualità dell'Interfaccia Grafica (GUI)

Per tutti i progetti che dovranno introdurre l'utilizzo di un'interfaccia grafica sarà valutato il corretto utilizzo del framework *Swing*.

Autenticità del codice sorgente

Il codice sorgente relativo al progetto dovrà essere frutto di un pensiero critico, di ricerca e, nel caso di un gruppo, di cooperatività. L'introduzione significativa di codice proveniente da fonti esterne sarà oggetto di annullamento della prova orale per tutto il gruppo con effetto retroattivo su quella scritta.

Relazione

Lo sviluppo del progetto dovrà essere finalizzato dalla redazione di un documento scritto in LaTeX (preferibilmente) o con un normale editor di testo. Tale documento dovrà essere scritto in modo chiaro e comprensibile e dovrà contenere le seguenti sezioni:

- **Introduzione:** al progetto, documentando la ripartizione del lavoro tra i componenti del gruppo
- **Descrizione delle classi:** dovrà essere fornita una descrizione delle classi con i loro attributi e metodi. La gerarchia delle classi dovrà essere documentata in modo opportuno tramite diagramma UML. Una particolare attenzione dovrà essere dedicata al polimorfismo, descrivendone l'eventuale impiego.
- **Descrizione delle funzionalità:** ciascun comportamento richiesto dalle specifiche dell'applicazione dovrà essere opportunamente documentato. L'idea è quella di ottenere una sezione in cui ciascuna funzionalità viene descritta mostrando il flusso di esecuzione (tramite screenshot o esempi di risultato) insieme a tutta la parte tecnica e decisionale necessaria ai fini dello sviluppo.
  NB: la scrittura di questa parte non dovrà contenere ingenti quantità di codice. Tale responsabilità dovrà essere lasciata ai commenti stessi presenti nel codice e all'impiego di codice parlante. Questa parte dovrà invece mostrare, motivare e documentare gli sviluppi e le conseguenti scelte che sono state prese alla loro base
- **Manuale della GUI:** in caso di progetto contenente una parte grafica, dovrà essere redatto un documento in cui venga spiegato il significato e l'utilità di ogni schermata, insieme ad una breve descrizione di ogni elemento contenuto al suo interno
- **Referenti di sviluppo:** per ogni funzionalità e/o schermata sviluppata dovrà essere indicato il corrispettivo referente o all'interno del gruppo
- **File README:** ogni progetto deve essere corredato dalla presenza di un file *readme* in cui viene descritto tutto ciò che è utile alla compilazione del progetto ed alla sua esecuzione.

Modalità di Consegna

La consegna dovrà avvenire via email indicando il prof. Quattrociocchi come destinatario.

Rispetto dei Termini di Consegna

I progetti dovranno **tassativamente** essere consegnati entro 5 giorni dalla data dell'orale. Non saranno accettate motivazioni di alcun tipo allo scadere dei termini di consegna, né modifiche.

Prova Orale

La prova orale consisterà nella discussione del progetto, con una serie di domande sia sullo sviluppo che sulla teoria sottostante ai fini di valutare la corretta proprietà intellettuale e conoscenze del lavoro svolto. Per questo motivo, è richiesto che il progetto sia pronto per essere eseguito, se richiesto.

È possibile preparare delle slide per esporre il proprio lavoro.

Strumenti di Collaborazione e Versionamento del Codice

Ai fini di una collaborazione produttiva all'interno dei vari gruppi e/o per una corretta gestione delle versioni del proprio codice ottenute durante lo sviluppo, è incoraggiato l'utilizzo di git e GitHub. Maggiori informazioni in merito verranno fornite durante il corso.

Per i gruppi interessati a scrivere il report del proprio progetto in LaTeX, è consigliato l'utilizzo di Overleaf, uno strumento collaborativo volto alla scrittura di documenti scientifici in LaTeX. Esso semplificherà la redazione di tali documenti in quanto non sarà necessario installare nessun eseguibile e/o libreria appartenente a LateX. In aggiunta a ciò, Overleaf è in grado di gestire le diverse versioni degli elaborati in base alle modifiche apportate, rendendo la scrittura collaborativa meno soggetta a errori.

FAQ

A seguire verranno riportate le domande fatte su Classroom con le relative risposte.

1. ***Per il progetto del Forza 4 dobbiamo tenere in conto che si possono mettere in pausa più partite o una sola?***
   Quindi nel "carica partita" dovremmo mettere uno o più slot?
   - Il programma per come è stato descritto dovrà gestire una partita alla volta. La scelta di mettere in pausa una partita e tornare ad un eventuale menù principale o di andare avanti sta al gruppo. Questo dovrebbe rispondere al dubbio relativo al numero di partite da poter mettere in pausa
   - La voce "Carica Partita" dovrà gestire più slot di salvataggio dunque, poiché ad una esecuzione possono essere associate diverse partite, ognuna con un suo possibile salvataggio
2. **Qual è la versione consigliata per la JDK?**
   È consigliata la 8 poiché comprende l'uso delle lambda expression.
3. **Nell'utilizzo di Selenium ricevo l'errore *x,y,z*. Come posso risolverlo?**
   Google e Stack Overflow. Non verrà fornito nessun tipo di aiuto in merito in quanto l'interfacciamento con framework esterni è parte del progetto.
4. **È possibile utilizzare JGraphX?**
   Sì.
5. **È possibile condividere il proprio lavoro tramite Google Drive?**
   Sì, a patto che l'accesso alla cartella venga reso disponibile anche ai membri che non appartengono al gruppo.