Guida tecnica del progetto:

Titolo del progetto: farmAdventure.

Creato da: Team Schiavi Pellicani (Daniele Schiavoni e Paolo Pellicanò), due studenti del corso Java organizzato dalla scuola bitCamp.

Scopo: Creare un gioco gestionale in cui il giocatore veste i panni di un contadino, e deve lavorare nella sua fattoria. Potrà comprare nuovi animali, sfamarli e curarli. Comprare nuovi campi, in cui seminare colture da cui ottenere frutta e verdura, e potrà accedere al suo magazzino, dove potrà vendere la frutta e la verdura raccolta.

Il gioco è sviluppato interamente in Java, con la seguente struttura:

1. Classi:
   a) Fattore: la classe fattore è dotata di due attributi: String nome e int denaro. Oltre ai costruttori, ai metodi setter e getter, e toString, dispone di funzioni per guadagnare e spendere denaro (rispettivamente guadagnaDenaro(int) e spendiDenaro(int)) e una funzione che ritorna un valore booleano haSoldi(int), per controllare, di fronte a un determinato prezzo, se il fattore può permetterselo.
   b) Animale: la classe Animale è dotata di tre attributi: String tipo, int salute e int fame. Oltre ai costruttori, ai metodi setter e getter, e toString, dispone di due metodi personalizzati: mangia(int) e cura(int). Entrambe le funzioni servono a migliorare le statistiche dell'animale, con cura che aumenterà il livello di salute e mangia che diminuirà il livello di fame. Sono da implementare in futuro ulteriori metodi, che possano consentire effettivamente all'animale di affamarsi, ammalarsi e morire. Da implementare anche metodi ulteriori per ottenere dei prodotti dall'animale (ad esempio latte, lana, o uova).
   c) Campo: la Classe campo ha due attributi, String tipoColturae int quantita, e, oltre ai costruttori e setter e getter, dispone del metodo specifico raccogli(int), per ridurre la quantità di campi raccogliendo i prodotti. Al momento questa funzione però non è implementata nel gioco, perché si è scelto di andare in un'altra direzione, facendo uso della funzione raccogliColtura nella classe Fattoria.
   d) Fattoria: la classe Fattoria ha quattro attributi: tre ArrayList chiamate rispettivamente animali, campi e prodotti, e un Fattore, che sarà il proprietario della classe. Oltre ai costruttori, i getter e i setter, la classe dispone di diverse funzioni:
      1. aggiungiAnimale(Animale animale)
      2. aggiungiCampo(Campo campo)

3. aggiungiProdotto(Prodotto prodotto) - Queste tre funzioni servono ad aggiungere rispettivamente oggetti Animale, Campo e Prodotto ai rispettivi ArrayList animali, campi e prodotti
4. cercaCampoPerColtura(String tipoColtura): cerca un determinato oggetto Campo nell'array utilizzando un for. Restituisce un Campo o null
5. seminaColtura(String tipoColtura, int quantita, Campo campo): per seminare una coltura. Non è stata utilizzata all'interno dell'applicativo, perché ridondante, comportandosi, di base, come un costruttore di Campo.
6. raccogliColtura(int indiceCampo, int quantita, int prezzo): aggiunge un prodotto da un campo.
7. vendiProdotto(int indiceProdotto): trova il prodotto nell'array,lo rimuove, e con la funzione guadagnaDenaro dà i soldi al fattore
8. prodottoPresente(int indiceProdotto)
9. animalePresente(int indiceAnimale): le due funzioni restituiscono un valore booleano dopo aver controllato se l'indice dell'animale o del prodotto è presente all'interno dei rispettivi array

2. Main: Nel main è presente la struttura effettiva del gioco(le funzionalità specifiche del menù possono essere trovate nel documento "tutorialFaq" presente sul sito del gioco). I metodi del main sono:
   a) int checkInt(): Utilizzando hasNext() e un ciclo, controlla che l'imput di int sia effettivamente un int;
   b) void animali(): il menù dedicato agli animali
   c) void campi(): il menù dedicato ai campi
   d) void magazzino: menù dedicato ai prodotti e ai metodi toString

Implementi futuri:

- Implementare metodi per gestire l'affamarsi, l'ammalarsi e il morire degli animali.
- Aggiungere metodi per ottenere prodotti dagli animali (latte, lana, uova).
- Espandere le funzionalità di gestione dei campi.
- Migliorare l'interfaccia utente per una migliore esperienza di gioco.
- Pulire il codice: sono state create molte funzioni native delle classi che poi non sono state implementate nel main.

Test:

I test principali eseguiti per il corretto funzionamento del gioco sono stati:

1. Corretta creazione del personaggio: all'apertura dell'applicativo, inserire il nome scelto per il fattore. Confermare la creazione del personaggio, e nel menu di avvio, premere il tasto 4(guardati allo specchio). Confermare che il nome corrisponda a quello scelto, e che il denaro iniziale sia di venti monete
2. Acquisizione di un animale: nel menu di avvio, premere il tasto 1(Vai dai tuoi animali), e poi di nuovo 1(Compra un animale). Dopo aver inserito il tipo di animale, premere 4(vedi i tuoi animali), per confermare che l'animale è stato aggiunto, e poi di nuovo 4(guardati allo specchio) dopo essere tornati al menu di avvio, per confermare che le monete del giocatore sono scese di 5
3. Semina del campo: dal menu di avvio premere 2(vai ai tuoi campi), 1(aggiungi campo) e acquisire un campo. Ansare poi nella sezione 3(Vedi i tuoi campi) e confermare la corretta acquisizione. Andare poi nella sezione 2(Semina campo) per confermare la possibilità di seminare il campo corretto.
4. Gestione dei prodotti: dopo aver seminato un campo, dal menu principale premere 3(Vai nel magazzino) e confermare la presenza del prodotto ottenuto con 1(vedi i tuoi prodotti). Provare a vendere il prodotto dal testo 2(vendi i tuoi prodotti) e inserire l'indice corretto del prodotto. Confermare dal menu principale il corretto incremento dei soldi.

Esperienze e lezioni imparate (Daniele):

Il progetto ha messo a dura prova la mia capacità di gestione di un lavoro, costringendomi a mettermi di fronte ad alcune mie limitazioni: realizzo di non essere bravo nel pianificare a lungo termine, e nel lavoro normalmente tendo a pensare alle cose man mano che le faccio. Ho dovuto però rendermi conto che questo mio approccio non è assolutamente funzionale in frangenti come questo: in primis, essendo un lavoro di gruppo, non posso permettermi di cambiare in corso d'opera alcuni elementi del progetto senza tener conto dei miei colleghi. In secondo luogo, è molto difficile lavorare a un progetto con più classi e documenti aperti, senza avere prima un quadro chiaro della situazione. Questo mi ha portato, nella stesura del main, ad avere difficoltà nell'integrare i metodi delle classi: mi sono reso conto solo dopo aver finito, infatti, di aver tralasciato di inserire i metodi creati, aggirandoli con linee di codice ridondanti inserite all'interno del main stesso.