Metodologie di Programmazione
II semestre a.a. 2024/2025
Canale M-Z e prevalentemente a distanza (“teledidattica”)

Versione 1.0 del 23 Aprile 2025

Si propongono due progetti; un progetto più semplice (**JTressette),** ed uno più complesso (**JImpossibleMission)** che permettono di accedere ad un massimo di **28** e **30** punti rispettivamente.

In caso di plagio si annulleranno tutte le consegne coinvolte, e poi saranno presi provvedimenti.

Utilizzare strumenti di disegno per disegnare i diagrammi delle classi (ad es., draw.io, Google Draw, Google Presentation, PowerPoint, etc…).

Ricordo che durante gli appelli straordinari si consegnano progetti e si svolgono orali e non si tengono prove scritte.

Inoltre, si ricorda che il voto del progetto pesa il 40% del voto finale (il 60% dipende dal voto dello scritto).

Si prega di consultare le sezioni delle proprie pagine del corso (classroom per la presenza canale MZ e pagina del corso Unitelma per la Teledidattica) per le regole sulla valutazione di scritti e progetti.

Le scadenze delle consegne dei progetti sono riportate su infostud per ogni appello, e sono entro le 23:59 di 5 giorni prima la data di un appello scritto.

Il limite superiore per la consegna di questo progetto è l’appello straordinario di marzo/aprile 2026, poi bisognerà attendere il nuovo progetto dell’a.a. 2025-2026.

JTressette

Voto massimo per il progetto ottenibile: **28**
Numero massimo di membri del gruppo: **1**

Regole e Video Tutorial

Regole del gioco: **Tressette - Wikipedia**
Video tutorial: TRESSETTE | Gioco di Carte del Seme Obbligatorio

Risorse

Online si trovano moltissime immagini di carte da gioco, e siti con campioni audio di pubblico dominio.

Consegna

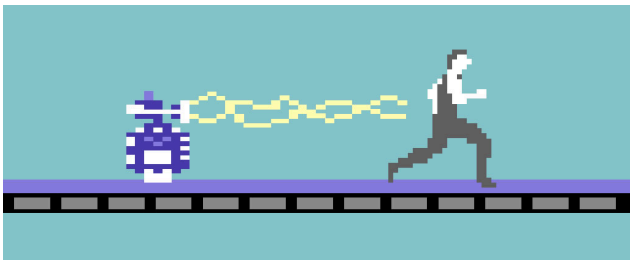
1) Consegnare il diagramma delle classi (esclusivamente in formato PDF)
2) Il progetto eclipse del gioco, con tutte le cartelle relative a codice sorgente e risorse (la classe JTressette deve contenere il main del gioco) (esclusivamente in formato ZIP e NON RAR o altri formati)
3) la documentazione completa generata con javadoc (nella forma di una cartella contenuta nel progetto eclipse del punto 2)
4) Una relazione **INDIVIDUALE** (esclusivamente in formato PDF) che descrive, almeno i seguenti punti **IMPORTANTE: UNA RELAZIONE DI UN PROGETTO SOFTWARE NON HA UN LIMITE SUPERIORE NEL NUMERO DI PAGINE (UNA RELAZIONE SERVE A VALORIZZARE IL VOSTRO LAVORO)**:
    a) Il numero di matricola
    b) corso (presenza MZ o Teledidattica)
    c) nome, cognome
    d) le decisioni di progettazione relative a ognuna delle specifiche (vedi sotto)
    e) I design pattern adottati, dove e perchè
    f) l'uso degli stream
    g) altre note progettuali e di sviluppo

Specifiche

1) Gestione del profilo utente, nickname, avatar, partite giocate, vinte e perse, livello …
2) Gestione di una partita completa con un giocatore umano contro 1, 2 o 3 giocatori artificiali

3) Uso appropriato di MVC [1,2], Observer Observable e di altri design pattern. **NON IMPLEMENTARE** MVC e/o ObservableObserver non è considerata una scelta appropriata, l'adozione è richiesta come **SPECIFICA DI PROGETTO.**
4) Adozione di Java Swing [2] o JavaFX [3] per la GUI
5) Utilizzo appropriato di stream
6) Riproduzione di audio sample (si veda appendice AudioManager.Java)
7) Animazioni ed effetti speciali (anche se limitati)

JImpossibleMission



Voto massimo per il progetto ottenibile: **30**
Numero massimo di membri del gruppo: **2**

Gameplay

C64 Longplay - Impossible Mission (complete)

gioca online: https://impossible-mission.krissz.hu/

Manuale delle istruzioni
https://archive.org/details/Impossible_Mission_1984_Epyx/mode/2up

Risorse

Online si trovano immagini con sprite del gioco e siti con campioni audio di pubblico dominio.

Commodore 64 - Impossible Mission - Player - The Spriters Resource
https://www.reddit.com/r/c64/comments/122baww/impossible_mission_a_study_of_the_map_and_objects/?rdt=55205

Consegna

1) Consegnare il diagramma delle classi (esclusivamente in formato PDF)
2) Il progetto eclipse del gioco, con tutte le cartelle relative a codice sorgente e risorse (la classe JImpossibleMission deve contenere il main del gioco) (esclusivamente in formato ZIP e NON RAR o altri formati)
3) la documentazione completa generata con javadoc (nella forma di una cartella contenuta nel progetto eclipse del punto 2)
4) Una relazione **INDIVIDUALE** (esclusivamente in formato PDF) che descrive, almeno i seguenti punti **IMPORTANTE: UNA RELAZIONE DI UN PROGETTO SOFTWARE NON HA UN LIMITE SUPERIORE NEL NUMERO DI PAGINE (UNA RELAZIONE SERVE A VALORIZZARE IL VOSTRO LAVORO)**:
    a) Il numero di matricola
    b) corso (presenza MZ o Teledidattica)
    c) nome, cognome, **e composizione del gruppo**
    d) le decisioni di progettazione relative a ognuna delle specifiche (vedi sotto)

e) I design pattern adottati, dove e perchè
f) l'uso degli stream
g) altre note progettuali e di sviluppo

Specifiche

Team di 1 persona

1) gestione del profilo utente, nickname, avatar, partite giocate, vinte e perse, livello …
2) gestione di una partita completa con almeno 8 livelli giocabili, 2 tipi di nemici con grafica e comportamento di gioco differenti, con gestione del punteggio, delle vite, delle tessere nascoste, dell'attivazione degli ascensori, del blocco temporaneo dei nemici, game over, continua, classifica….
3) uso appropriato di MVC [1,2], Observer Observable e di altri Design Pattern; l'adozione è richiesta come **SPECIFICA DI PROGETTO.**
4) adozione di Java Swing [2] o JavaFX [3] per la GUI
5) utilizzo appropriato di stream (Stream<T>)
6) riproduzione di audio sample (si veda appendice AudioManager.Java)
7) animazioni ed effetti speciali

Team di 2 persone

8) le specifiche da 1 a 7
9) almeno 16 livelli, tutti i tipi di nemici più 2 nemici nuovi con nuovi attacchi e comportamenti,
10) L'editor dei livelli di gioco dove poter specificare:
    - posizione e tipologia delle piattaforme
    - posizione e tipologia dei nemici
    - posizione e tipologia degli oggetti (librerie, computer …)

Riferimenti

[1] https://it.wikipedia.org/wiki/Model-view-controller
[2] Java Swing e MVC Tutorial (Attenzione questa implementazione di MVC non prevede l'adozione di Observer Observable, mentre è richiesto di adottare anche Observer Observable per la gestione delle notifiche provenienti dal Model) : https://www.youtube.com/watch?v=-NiKk9UqUoo&list=PLU8dZfh0ZlUn7-TDZfSmX9QRnBgmdJJWD
[3] Tutorial del corso di Metodologie di Programmazione
https://github.com/sapienza-metodologie-di-programmazione/guide?tab=readme-ov-file

Appendice (AudioManager.Java)

Provate una delle due versioni, il funzionamento dipende dalle distribuzioni di JRE.

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import sun.audio.AudioPlayer;
import sun.audio.AudioStream;

public class AudioManager {
	private static AudioManager instance;

	public static AudioManager getInstance() {
		if (instance == null)
			instance = new AudioManager();
		return instance;
	}
	private AudioManager() {

	}
	public void play(String filename) {
		try {
			InputStream in = new FileInputStream(filename);
			AudioStream sound = new AudioStream(in);
			AudioPlayer.player.start(sound);
		} catch (FileNotFoundException e1) {
			e1.printStackTrace();
		} catch (IOException e1) {
			e1.printStackTrace();
		}
	}
```

}

Esempio di riproduzione di un sample audio

```
AudioManager.getInstance().play("resources/audio/hit.wav");
```

Altra versione di AudioManager (JDK>9)

```
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class AudioManager {

	private static AudioManager instance;

	public static AudioManager getInstance() {
		if (instance == null)
			instance = new AudioManager();
		return instance;
	}

	private AudioManager() {

	}

	public void play(String filename) {

		try {
			InputStream in = new BufferedInputStream(new
FileInputStream(filename));
			AudioInputStream audioIn = AudioSystem.getAudioInputStream(in);
			Clip clip = AudioSystem.getClip();
			clip.open(audioIn);
			clip.start();
		} catch (FileNotFoundException e1) {
			e1.printStackTrace();
		} catch (IOException e1) {
			e1.printStackTrace();
		} catch (UnsupportedAudioFileException e1) {
			e1.printStackTrace();
		} catch (LineUnavailableException e1) {
			e1.printStackTrace();
		}
	}
}
```