# Metodologie di Programmazione
# II semestre a.a. 2022/2023
# Canale M-Z e prevalentemente a distanza (teledidattica)

Versione 1.1 del 30 Marzo 2023

Si propongono due progetti; un progetto più semplice (**JTrash),** ed uno più complesso (**JBomberMan)** che permettono di accedere ad un massimo di 27 e 31 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., Google Draw, Google Presentation, PowerPoint, draw.io, 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 massimo per la consegna di questo progetto è l'appello straordinario di marzo/aprile 2024, poi bisognerà attendere il nuovo progetto dell'a.a. 2023-2024.*

# JTrash

Voto massimo per il progetto ottenibile: **27**
Numero massimo di membri del gruppo: **1**

## Regole e Video Tutorial

Regole del gioco: **(https://www.wikihow.com/Play-Trash)**
Video tutorial: (**https://www.youtube.com/watch?v=vnLI2O5QuPk**)

## Risorse

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

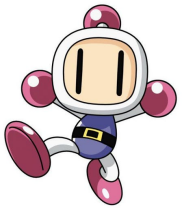
## Consegna

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 JTrash 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:
    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 1/2/3 giocatori artificiali
3) Uso appropriato di MVC [1,2], Observer Observable e di altri design pattern
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)

# JBomberman (JDynaBlaster)



Voto massimo per il progetto ottenibile: **31**
Numero massimo di membri del gruppo: **3**

## Gmeplay

https://www.youtube.com/watch?v=9-E4u4flsi0

**Manuale delle istruzioni**
https://drive.google.com/file/d/18RXnB6wbDFi7cBCBsH4SXPnrBWBUBMDe/view?usp=sharing

## Risorse

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

http://www.kousougames.com/sprites.php?show=Sprites%20from%20Super%20Bomberman
https://www.spriters-resource.com/snes/sbomber/

## 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 JBomberMan 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:
    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

## Team di 1 persone

1) gestione del profilo utente, nickname, avatar, partite giocate, vinte e perse, livello …
2) gestione di una partita completa con almeno due livelli giocabili, due tipi di nemici con grafica e comportamento di gioco differenti, con gestione del punteggio, delle vite, di 3 power up, shermata hai vinto,  game over, continua.
3) uso appropriato di MVC [1,2], Observer Observable e di altri Design Pattern
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

## Team di 2 persone

8) le specifiche da 1 a 7
9) almeno 5 livelli e 2 boss, 5 tipologie di nemici e con grafica e comportamento di gioco differenti, gioco anche con mouse (ad esempio, left click mi sposto, right click rilascio bomba)
10) 8 power up

## Team di 3 persone

11) le specifiche da 1) a 11)
12) almeno 8 livelli
13) l'editor dei livelli

# 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

# Appendice (AudioManager.Java)

```
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");
```