Relazione Progetto Maratona

INDICE

Introduzione del lavoro 3

Descrizione consegna 3

Eventi da gestire 3

Suddivisione del lavoro 4

COSTRUZIONE SCHEMA UML CLASSI 4

Schema delle classi 5

Codice 6

Punti critici 12

PROBLEMI 12

Cosa abbiamo imparato ? 13

Introduzione del lavoro

Il lavoro assegnatoci era quello di creare un gioco che simulasse la maratona di New York tramite un'applicazione in ambiente Phyton.

Descrizione consegna

Il programma prevede che ci siano uno o più atleti a cui vengono inserite le loro generalità (nome, cognome, età, peso, ecc…).

Per partecipare alla gara bisogna soddisfare un tempo minimo richiesto, in questo caso di 5 minuti per km.

Il programma deve essere costruito integrando i Thread per così simulare più realisticamente lo svolgimento di una vera e propria maratona facendo gareggiare parallelamente i concorrenti.

Durante la gara vi saranno vari eventi gestiti in modo casuale che possono portare benefici o svantaggi all'atleta.

Gli eventi devono essere eseguiti ogni due secondi (ogni due km).

Eventi da gestire

1 – Scatto (il tempo a km viene ridotto del 30% per 2 secondi poi ritorna normale)

2 - Contrattura (il tempo a Km viene raddoppiato fino a prossimo evento: all'evento successivo alla contrattura si genera un numero random tra 1 e 2 per verificare se si scioglie la contrattura e si può tornare a correre normale. Se viene sorteggiato 1, la contrattura rimane e non deve essere associato alcun altro evento, se il numero è 2 la contrattura viene sciolta e si procede ad un nuovo evento)

Da 3 a 7 – Andatura normale (il tempo a Km rimane invariato)

8 – Stiramento (il tempo a Km viene quadruplicato fino alla fine, quindi si interrompe la gestione degli eventi)

9 – Ritmo in aumento (il tempo a Km viene ridotto del 10%)

10 – Stanchezza (il tempo a Km viene aumentato del 10%)

Ogni secondo un atleta ci mette il tempo a km che ha impostato, ogni secondo viene incrementato il tempo totale dell'atleta. Alla fine dei km si decreta il vincitore e la classifica in base a chi ha impiegato meno tempo. A fine gara, di ogni atleta deve venire salvato un report dell'andamento della gara, così da andare a verificare il suo percorso durante la competizione.

Suddivisione del lavoro

Il lavoro, svolto in coppie, è stato suddiviso in 4 parti:

1) Costruzione schema UML per le classi
2) Sviluppo software
3) Relazione del lavoro svolto in file Word
4) Presentazione preparata in PowerPoint

COSTRUZIONE SCHEMA UML CLASSI

Per la costruzione dello schema il lavoro è stato assegnato a Bastianelli Lorenzo il quale a prodotto una schematizzazione del lavoro inserendo le classi principali per il funzionameno dei Thread e la collezione degli atleti pertecipanti in gara.

Per lo sviluppo software il lavoro è stato assegnato a Fiore Enrico che ha implementato lo schema UML delle classi per costruire il software, aggiungedovi un main con annesso un menu per inserire i partecipanti o toglierli e per poi mostrare la gara e gli eventi che avvengono ogni 2 km di maratona.

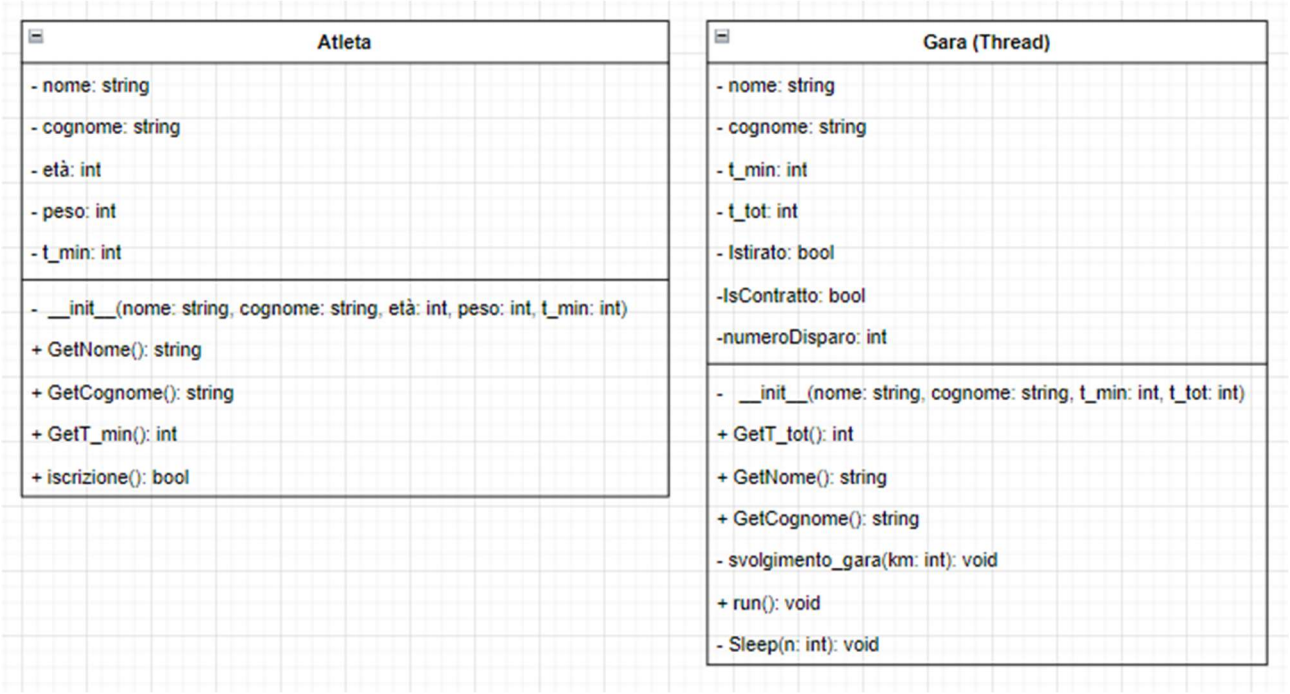
Per la relazione del lavoro in file Word il lavoro è stato assegnato a Bastianelli Lorenzo con la seguente supervisione di Fiore Enrico.

Per la preparazione della presentazione in PowerPoint il lavoro è stato assegnato a Fiore Enrico con la seguente supervisione di Bastianelli Lorenzo.

Il lavoro è stato svolto sia a casa che durante le ore scolastiche.

I file sono stati consdivisi usando GitHub.

Schema delle classi



Codice

```
from threading import Thread
import time
import random

#Classe statistiche atleta
class Atleta:
    def __init__(self, nome, cognome, età, peso, t_min):
        self.nome = nome
        self.cognome = cognome
        self.età = età
        self.peso = peso
        self.t_min = t_min

    def GetNome(self):
        return self.nome
    def GetCognome(self):
        return self.cognome
    def GetT_min(self):
        return self.t_min

    def iscrizione(self) :
        if(self.t_min <= 5) :
            return True
        else :
            print(self.nome + " " + self.cognome + " non puoi parteciapare perchè
il tempo è maggiore di 5, tempo minimo atleta: " + str(self.t_min) + "\n")
            return False



#Classe con Thread
class Gara(Thread):

    def __init__(self,nome,cognome,t_min,t_tot):
        Thread.__init__(self)#creo Thread
        self.nome = nome
        self.t_min = t_min
        self.t_tot = t_tot
        self.cognome = cognome

    # variabili generali per salvare il dato del singolo maratoneta
    numeroDisparo = 0
    Istirato = False
    IsContratto = False
```

```
    def Sleep(n):
        time.sleep(n)

    def GetT_tot(self):
        return self.t_tot
    def GetNome(self):
        return self.nome
    def GetCognome(self):
        return self.cognome

    def svolgimento_gara(self,km):
        # il segno // divide per interi invece / è la divisione normale con la
virgola
        if(self.Istirato==False):
            if(self.IsContratto == False):
                if(km <= 1):# se è il primo km o meno lo indirizzo direttamente ad
una andatura normale
                    n_random = 3
                elif(km % 2 == 0):# ogni 2 km estraggo un numero casuale per
l'evento
                    n_random = random.randint(1,10)#numero casuale da 1 a 10
                    self.numeroDisparo  = n_random
                elif(km % 2 != 0 and km >2):# l'evento successo nel km precedente
(pari) continua anche per il km successivo (disparo)
                    n_random = self.numeroDisparo
                if(n_random == 1 and self.Istirato == False): #scatto
                    print(self.nome + " " + self.cognome + " ha fatto uno
scatto\n")
                    self.t_tot += self.t_min // 0.7
                    Gara.Sleep(2)
                    self.t_tot += self.t_min
                if(n_random == 2 and self.Istirato == False): #contrattura
                    print(self.nome + " " + self.cognome + " ha ricevuto una
cotrattura\n")
                    self.t_tot += self.t_min * 2
                    self.IsContratto = True
                if(n_random>=3 and n_random<=7 and self.Istirato ==
False):#andatura normale
                    print(self.nome + " " + self.cognome + " corre spensierato\n")
                    self.t_tot += self.t_min
                if(n_random == 8): # stiramento
                    print(self.nome + " " + self.cognome + " ha ricevuto
stiramento\n")
                    self.t_tot += self.t_min * 4
                    self.Istirato = True
                if(n_random == 9 and self.Istirato == False):#ritmo in aumento
                    print(self.nome + " " + self.cognome + " ha aumentato il
ritmo\n")
                    self.t_tot += self.t_min // 0.9
                if(n_random == 10 and self.Istirato == False):#stanchezza
```

```
                    print(self.nome + " " + self.cognome + " inizia a sentire la
stanchezza!\n")
                    self.t_tot += self.t_min * 1.1
            else:
                if(random.randint(1,2)==2): #non ha più la contrattura
                        self.t_tot -= self.t_min
                        print(self.nome + " " + self.cognome + " non ha più la
contrattura\n")
                        self.IsContratto = False
        else:
             print(self.nome + " " + self.cognome + " è ancora stirato\n")
             self.t_tot += self.t_min * 4
        Gara.Sleep(1)#da il tempo di un 1 secondo a km



    def run(self):
        print("è partito: " + self.nome + " " +self.cognome )
        for km in range(42):#simulazione di una maratona, ogni km succede qualcosa
            Gara.svolgimento_gara(self,km+1)
            print(self.nome + " " + self.cognome +" al chilometro: " + str(km+1) +
" di corsa\n")
            #print(self.nome + " " + self.cognome +" tempo: " + str(self.t_tot))




"""
#inserimento alteti
Lorenzo = Atleta ("Lorenzo", "Bastianelli", 17, 70, 5)
Enrico = Atleta ("Enrico", "Fiore", 18, 65, 7)
Gino = Atleta ("Gino", "Rossi", 19, 80, 3)
Pippo = Atleta ("Pippo", "Baudo", 20, 85, 8)
Poldo = Atleta ("Poldo", "Bianchi", 21, 85, 4)
Atleti.append(Lorenzo)
Atleti.append(Enrico)
Atleti.append(Gino)
Atleti.append(Pippo)
Atleti.append(Poldo)
"""
def mostraAtleti():
    j=1
    for i  in Atleti:
        print("Partecipante "+ str(j) + ": " + i.GetNome() + " " + i.GetCognome()+
"\n")
        j+=1

#MENU
def Menu():
    nelMenu = True #per entrare e rimanere nel menu
```

```
    while(nelMenu == True):
        print("Scegliere una tra le seguenti opzioni: \n1) Inserire atleta\n2) 
Eliminare atleta\n3) Iniziare gara\n4) Mostra partecipanti\n")
        while(True):
            try:
                scelta = int(input("Inserire il NUMERO dell'opzione: "))
                if scelta in (1, 2, 3,4):# se la scelta è 1 o 2 o 3 o 4 esci dal 
try catch
                    break
                else:
                    print("ATTENZIONE: bisogna inserire il numero corrispondente 
alla scelta")
            except ValueError:
                print("ATTENZIONE: bisogna inserire il numero corrispondente alla 
scelta")

        if( scelta == 1): #inserire atleta
            print("Per ISCRIVERE un atleta ci serve: NOME, COGNOME, ETA', PESO, 
TEMPO MINIMO A KM\n")
            print("Inserire NOME\n")
            nome = str(input("nome: "))
            print("\nInserire COGNOME\n")
            cognome = str(input("cognome: "))
            print("\nInserire ETA'\n")
            età = int(input("età: "))
            print("\nInserire PESO\n")
            peso = int(input("peso: "))
            print("\nInserire TEMPO MINIMO A KM\n")
            t_min = int(input("tempo minimo: "))
            nuovo_atleta = Atleta(nome,cognome,età,peso,t_min)#creazione nella 
classe Atleta
            Atleti.append(nuovo_atleta)# aggiunto nell'array
        elif(scelta == 2):# elimina atleta
            mostraAtleti()
            print("Per ELIMINARE un atleta ci serve: NOME, COGNOME\n")
            print("Inserire NOME\n")
            nome = str(input("nome: "))
            print("\nInserire COGNOME\n")
            cognome = str(input("cognome: "))
            nonTrovato =0
            for atleta in Atleti:
                if(atleta.GetNome() == nome and atleta.GetCognome() == cognome):
                    Atleti.remove(atleta)# rimosso atleta dall'array
                    print("\nL'Atleta"+ nome +" "+ cognome+" è stato rimosso con 
successo\n")
                    break
                else:
                    nonTrovato+=1 # per tenere traccia di quante volte il nome 
inserito non è stato trovato
```

```
            if(nonTrovato == len(Atleti)):# in caso il nome è stato inserito
sbagliato
                print("\nAtleta non trovo, provare a reinserire atleta\n")
        elif(scelta == 3):# inizia gara
            #controllo di chi può partecipare
            for i in Atleti :
                if i.iscrizione():
                    partecipanti.append(i)
            if(len(partecipanti)<2): #controllo che ci siano abbastanza atleti per
fare la gara (minimo 2)
                print("Ci sono ancora pochi alteti per iniziare la gara\n")
            else:
                nelMenu = False
        elif(scelta == 4):# mostra partecipanti
            mostraAtleti()


#MAIN
#t_min = tempo di corsa per gareggiare 5 per Kilometro

Atleti = [] #array atleti
partecipanti = [] # partecipanti gara
Menu()
#array in cui salvare risultati
risultati = []
#Partenza gara
for i in partecipanti:
    corridore = Gara(i.GetNome(),i.GetCognome(),i.t_min,0)
    corridore.start()
    risultati.append(corridore)
#Aspettando che tutti finiscano la gara
for thread  in risultati:
    thread.join()
print("------------------------------GARA FINITA------------------------------\n")
#stampa risultati
tempoVincente = 999#numero default
nomeVincitore= ""
cognomeVincitore =""
for corridori in risultati:
    print(corridori.GetNome() + " " + str(corridori.GetT_tot()))#controllo tempo
minore
    if(corridori.GetT_tot()< tempoVincente):
     tempoVincente = corridori.GetT_tot()
     nomeVincitore = corridori.GetNome()
     cognomeVincitore = corridori.GetCognome()


#stampa vincitore
print("Vincitore maratona: " + nomeVincitore + " "+ cognomeVincitore +" con un
tempo di: " + str(tempoVincente))
```

```
#ATTENZIONE!!!
#prendere il risultato in minuti [non in secondi]
```

Punti critici

Durante questo lavoro non abbiamo riscontrato enormi problemi, ma ciò non vuol dire che i problemi siano stati 0.

I problemi principali cha abbiamo avuto sono stati fondalmente 3 ma che per nostra fortuna siamo riusci a risolvere velocemente.

PROBLEMI

1) Mostrare i risultati quando tutti i partecipanti hanno finito la gara
2) Gestire le classi
3) Implementare i Thread

Il problema per quanto riguarda il join dei Thread, era un problema per il quale venivano mostrati i risultati della gara nonostante alcuni Thread dovevano completare la maratona.

Tale problema è stato risolto facendo un join per ogni Thread attivo così da dover far temrinare tutti i Thread prima di far finire la maratona.

Il problema della gestione delle classi è un problema riguardante all'utilizzo e alla schematizzazione delle classi essendo esse abbastanza differenti da come siamo abituati con il C#.

Il problema dell'implementazione dei Thread era un problema riguardante al capire come i Thread dov'essero essere implementati per riuscire a far pertire gli atleti contemporaneamente.

Cosa abbiamo imparato ?

Durante questa collaborazione, abbiamo affrontato sfide e compiti distribuendoli equamente, unendo le nostre competenze per padroneggiare il linguaggio Python. L'uso dei Thread ci ha permesso di comprendere meglio il concetto di concorrenza nel codice, apprendendo come gestire l'esecuzione di più operazioni contemporaneamente.

Inoltre, la nostra scelta di condividere il lavoro con GitHub ci ha permesso di imparare più approfonditamente come può essere un'ambiente di lavoro per lo sviluppo sofware. Abbiamo imparato a utilizzare questa piattaforma per collaborare in modo efficiente. Grazie a GitHub, abbiamo condiviso il nostro codice, tracciato le modifiche e lavorato su diverse parti dello stesso progetto contemporaneamente, mantenendo sempre una chiara visione delle nostre attività e delle modifiche apportate.

Questa esperienza ha non solo consolidato le nostre conoscenze di programmazione Python, ma ci ha anche dotato di competenze fondamentali nell'utilizzo di strumenti collaborativi essenziali per lo sviluppo software. La capacità di dividere compiti, risolvere problemi e comunicare in modo efficace è stata fondamentale per il successo di questo progetto.

In sintesi, attraverso il nostro lavoro con Python, l'utilizzo dei Thread e l'esperienza con GitHub, abbiamo acquisito un metodo di lavoro nella programmazione e nella gestione collaborativa del codice, preparandoci in modo significativo per affrontare sfide più complesse e progetti futuri con sicurezza e competenza.