Relazione Progetto Maratona

INDICE

[Introduzione del lavoro 3](#_Toc152694137)

[Descrizione consegna 3](#_Toc152694138)

[Eventi da gestire 3](#_Toc152694139)

[Suddivisione del lavoro 4](#_Toc152694140)

[Schema delle classi 5](#_Toc152694141)

[Codice 6](#_Toc152694142)

[Punti critici 7](#_Toc152694143)

[Cosa abbiamo imparato ? 8](#_Toc152694144)

# 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

L’alunno Fiore Enrico si è occupato principalmente della parte del codice con la supervisione e la comprensione dell’alunno Bastianelli Lorenzo. Bastianelli ha sviluppato lo schema UML.

L’alunno Fiore Enrico si è occupato

# Schema delle classi

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

# 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))

#t\_min = tempo di corsa per gareggiare 5 per Kilometro

#inserimento alteti

Atleti = [] #array atleti

partecipanti = [] # partecipanti gara

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)

#controllo di chi può partecipare

for i in Atleti :

    if i.iscrizione():

        partecipanti.append(i)

#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)

#    if(i == Poldo):#ultimo a partire e quindi chiude il Thread

#        corridore.join()

#print("ultimo corridore: " + str(corridore))

#corridore.join()

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

I punti critici che abbiamo riscontrato durante questo lavoro sono stati la costruzione del codice facendo finire contemporaneamente tutti i Thread

# Cosa abbiamo imparato ?

Durante questo lavoro abbiamo imparato a lavorare insieme ed a suddividerci il lavoro in parti uguali.

Questo lavoro ci ha aiutato a capire meglio come programmare con il linguaggio Python tramite l’utilizzo dei Thread e come usare la piattaforma Github, condividendo il codice e alcune cartelle con altre persle vedi le modifiche one.