Unità didattica sulla programmazione ad oggetti

Alessandro Freda

15-06-2020

1 Titolo

Principi dell'OOP: un percorso verso la progettazione di un videogioco... con Bob!

2 Disciplina di riferimento

Informatica per un Liceo Scientifico indirizzo Scienze Applicate.

3 Classe

Questa UD è proposta per il quarto anno del Liceo Scientifico indirizzo Scienze Applicate. La classe è composta da 23 alunni. Nella classe si trovano grosse disparità tra gli studenti, alcuni eccellono altri, circa 6, sono poco attenti alle lezioni e non studiano costantemente. Il livello generale di conoscenza per la materia è medio, gli studenti fanno fatica a comprendere concetti più complessi e non riescono a risolvere esercizi più articolati in quanto non hanno buone fondamenta sulla programmazione.

4 Motivazione e Finalità

Essendo la programmazione ad oggetti un argomento della programmazione abbastanza complessa per i novizi, in particolare per degli studenti di un liceo scientifico che hanno a disposizione poche ore settimanali per l'informatica, è necessario creare un'unità didattica volta a far comprendere al meglio questo argomento agli studenti con nuove metodologie didattiche attraverso la creazione di un videogioco. Personalmente la programmazione ad oggetti è stata uno degli argomenti con cui ho avuto maggiori difficoltà nella comprensione durante il mio percorso di studi di informatica. Lo scopo di questa unità didattica è dunque quello introdurre i

concetti base della programmazione ad oggetti cercando di coprire gli aspetti fondamentali dell'argomento partendo da metodologie che permettano allo studente la piena comprensione soprattutto mediante la creazione di schemi mentali corretti, è importante dunque che le lezioni siano coinvolgenti e che permettano allo studente di partecipare attivamente alla lezioni in modo da avere i maggiori risultati.

5 Prerequisiti

Si presuppone che lo studente abbia familiarità con il linguaggio di programmazione Python. I requisiti necessari per questa unità didattica sono:

- Concetti base di programmazione
- Funzioni, definizione e chiamata
- Passaggio dei parametri

6 Contenuti

Nella seguente unità didattica verranno insegnati i principi base della programmazione ad oggetti come richiesto dalle linee guida del ministero dell'istruzione per i licei scientifici ad indirizzo scienze applicate. Della programmazione ad oggetti verranno introdotti i seguenti argomenti:

1. Definizione di classe e definizione di istanza
2. Costruttori e la parola chiave self
3. Variabili di istanza e variabili di classe
4. Metodi di istanza e metodi di classe
5. Concetti di base dell'ereditarietà

7 Obiettivi

7.1 Collegamento con i documenti ministeriali/proposte

La seguente unità didattica è volta ad insegnare l'argomento della programmazione ad oggetti seguendo le linee ministeriali presenti nel documento: *"Indicazioni nazionali riguardanti gli obiettivi specifici di apprendimento concernenti le attività e gli insegnamenti compresi nei piani degli studi previsti per i percorsi liceali di*

cui all'articolo 10, comma 3, del decreto del Presidente della Repubblica 15 marzo 2010, n. 89, in relazione all'articolo 2, commi 1 e 3, del medesimo regolamento." Nelle seguenti linee guida ministeriali la programmazione ad oggetti è posta nel secondo biennio, dunque quarto anno, sotto l'area tematica di AL (algoritmi e linguaggi)

7.2 Obiettivi

I seguenti obiettivi sono stati applicati sulla base della tassonomia qui presente (https://ibb.co/0ym4J8Z)

1. Il conoscere
    - Riprodurre la sintassi degli oggetti e delle classi in Python.
    - Classificare oggetti di una stessa classe o oggetti di classi diverse.
    - Definire il concetto di istanza/oggetto.
    - Riconoscere un oggetto e la sua classe corrispondente.
    - Riconoscere un problema risolvibile con la programmazione ad oggetti.
2. Il comprendere
    - Descrivere la soluzione ad un problema mediante l'utilizzo della programmazione ad oggetti.
    - Riconoscere un problema risolvibile con la programmazione ad oggetti.
    - Applicare procedimenti volti all'utilizzo di classi ed oggetti.
    - Verificare l'esecuzione del codice mediante un visualizer/tracer.
3. Pensiero convergente
    - Analizzare un esercizio svolto mediante la programmazione ad oggetti.
    - Schematizzare soluzioni per risolvere un problema con la programmazione ad oggetti.
4. pensiero divergente
    - Formulare ipotesi di soluzione con oggetti che hanno come variabili altri oggetti.
    - Riconoscere il problema dei modificatori di visibilità in una classe.
    - Tentare soluzione con la programmazione ad oggetti per problemi risolvibili anche senza di essa.

8 Metodologie didattiche

La seguente unità didattica è stata ideata sulla base di studi analizzati. La classe, per comprendere al meglio l'argomento della programmazione ad oggetti, sarà guidata attraverso un percorso che ricoprirà tutte le lezioni. Ogni lezione non sarà distinta dalle altre, ma saranno tutte collegate sulla base dello stesso principio: la creazione di un videogioco mediante la libreria PyGame `https://www.pygame.org/`. Gli studenti saranno invitati a creare gruppi nei quali applicheranno, con approcci di collaborative learning, i concetti esposti dal professore relativi al gioco. I componenti dei gruppi non saranno sempre gli stessi ma saranno scambiati più volte sulla base dell'andamento. Il docente esporrà esempi inerenti al videogioco e non esempi astratti. L'intento è quello di avvicinarsi ad una forma di costruzionismo che Papert definisce come "pensiero concreto" [1]. Mediante attività di "costruzione" (creazione di un videogioco) gli studenti saranno in grado di apprendere con maggiore efficacia. Si cerca di incentivare la fantasia degli studenti.
Gli studi sulla didattica della programmazione ad oggetti analizzati [2] e [3], sostengo la tesi secondo cui coinvolgere gli studenti attraverso la creazione di giochi interattivi ed interessanti permette di mantenere la motivazione nello studio e allo stesso tempo consolidare i concetti studiati. Attraverso un approccio top-down il docente insegna passo dopo passo le principali funzioni della libreria e permetterà agli studenti di esprimersi liberamente con gli strumenti messi a disposizione. Uno dei problemi dell'insegnamento di un corso di programmazione è che gli studenti non sono ugualmente preparati. Risulta complesso personalizzare i contenuti del corso sulla base della preparazione personale degli studenti. Lo sviluppo di un gioco o esercizi in laboratorio permette agli studenti, anche quelli meno preparati, di divertirsi imparando. Gli studenti sono incoraggiati ad affrontare sfide sempre più impegnative. Pertanto, tutti gli studenti imparano e rendono significativo il progresso ottenuto durante lo svolgimento degli esercizi indipendentemente dalla loro preparazione. In classe dunque si svolgeranno solo attività relative allo sviluppo del videogioco, per casa possono invece essere assegnati esercizi generali sulla programmazione ad oggetti. In questo modo gli studenti oltre ad imparare a risolvere esercizi con difficoltà graduale saranno spinti ad applicare i concetti della programmazione ad oggetti in nuove forme creative mediante la creazione di un videogioco anche concepito sulle passioni personali.
L'obiettivo principale di questa unità didattica è, inoltre, quello di evitare che gli studenti creino misconcezioni sull'argomento ed essendo questa una didattica sulla programmazione, si cerca di favorire gli studenti a creare i giusti schemi mentali necessari per la comprensione e la concreta applicazione della programmazione ad oggetti. Si farà largo uso di strumenti di code visualizer per la visualizzazione della macchina concettuale `http://www.pythontutor.com/`.

8.1 Il linguaggio che si adatta meglio alla didattica sulla programmazione ad oggetti

Prima della creazione della seguente unità didattica ho riflettuto a lungo su quale linguaggio di programmazione utilizzare. La scelta era tra Java e Python. La decisione non è stata semplice, Java è un linguaggio la cui caratteristica identitaria è la programmazione ad oggetti tuttavia è anche famoso per la difficoltà dovuta all'eccessiva sintassi che porta ad un maggiore carico cognitivo per gli studenti. Python, al contrario, risulta essere un linguaggio con un minor carico cognitivo, la cui sintassi è molto più snella ed ha la possibilità di utilizzare molti strumenti che facilitano la didattica e la comprensione degli argomenti agli studenti. Nonostante ciò anche Python (come Java) ha i suoi lati negativi, uno tra i quali è proprio la programmazione ad oggetti, la quale a mio parere non risulta comprensibile e inequivocabile quanto Java.
Inoltre oggi nei licei molti professori avvicinano gli studenti al mondo della programmazione con Java e non con Python, una scelta che non è sempre condivisibile ma bisogna prenderne atto in quanto nella realtà ci si potrebbe ritrovare in una classe nella quale gli studenti hanno una maggiore padronanza del linguaggio Java e non Python. Ad un argomento complesso con un carico cognitivo notevole non si deve aggiungere il carico, già di per sé molto alto, del linguaggio di programmazione richiedendo agli studenti di impararne uno nuovo, seppur questo molto più semplice. Dunque è molto importante tenere conto del linguaggio di programmazione con il quale gli studenti hanno padronanza.
La scelta per questa unità didattica, in conclusione, è ricaduta su Python. In questo caso Python permette di costruire lezioni coinvolgenti e allo stesso momento molto educative. Il carico cognitivo è nettamente inferiore, una sintassi più snella porterà gli studenti ad una maggiore comprensione dell'argomento. Infine anche la parte di debugging sarà molto più semplice. Un altro aspetto importante da considerare è il tempo a disposizione per la didattica, essendo in questo caso molto poco, Python è il linguaggio che meglio si adatta a questo tipo di situazione. Ho inoltre riassunto in una tabella quelle che, per me, sono i pro e i contro di Python e Java per l'insegnamento della programmazione ad oggetti.

Tabella 1: Pro e contro dei linguaggi di programmazione Java e Python per la didattica sulla programmazione ad oggetti

	Pro	Contro
Java	• Particolarmente adatto alla programmazione ad oggetti • La chiarezza dei modificatori di accesso (o modificatori di visibilità) permette ai programmatori poco esperti di evitare di commettere errori di assegnazioni di variabili istanza o errori sui metodi istanza • La presenza dell'operatore unario new usato per creare oggetti di una classe è molto utile per la fase di comprensione della creazione di un oggetto, essendo questo operatore utilizzato solo a questo scopo lo studente individua senza dubbi la fase di creazione di un oggetto • Costruttori di una stessa classe con parametri diversi devono essere scritti separatamente (nella fase di comprensione è importante distinguere in maniera "visiva" il costruttore in base ai parametri)	• Eccessiva sintassi che porta ad un maggiore carico cognitivo durante la fase di comprensione della programmazione ad oggetti. • Regole troppo restrittive, presenza quasi obbligatoria dei modificatori di accesso e l'eccessiva sintassi porta lo studente a non concentrarsi sul concetto di programmazione ad oggetti e a distrarlo facilmente • Necessita di più tempo a disposizione per la didattica
Python	• Minore sintassi e minore carico cognitivo durante la fase di apprendimento • Maggiore libertà, non si richiedono modificatori di accesso • Più semplice da leggere e da scrivere • Sono disponibili molti tool per l'insegnamento • Nei casi in cui il tempo a disposizione per la didattica è poco Python è la migliore scelta	• Regole non rigide non aiutano lo studente a comprendere al meglio i concetti portandolo spesso a commettere errori ed eventualmente creare misconcezioni • Non esistono particolare restrizioni nell'accesso. Anche se un metodo è scritto con il doppio underscore come prefisso nel nome, il metodo non è mai realmente privato in quanto è facilmente accessibile utilizzando _nomeclasse_nomemetodoprivato

9 Tempi

La seguente unità didattica è stata pensata per durare circa 3 mesi comprese le verifiche di apprendimento. Data la continuità delle lezioni, il numero può sensibilmente aumentare sulla base di vari fattori: l'andamento generale della classe, la comprensione dei concetti teorici, la svolgimento completo dei progetti. In un liceo scientifico scienze applicate il tempo messo a disposizione per la materia dell'informatica è di 2 ore alla settimana. Dato il grande spazio di tempo tra una lezione e l'altra verranno assegnati esercizi di consolidamento degli argomenti ad ogni fine lezione. Ogni lezione ha la durata di 2 ore. Per questo argomento sono state ipotizzate:

- 6 lezioni
- 1 verifica in itinere senza valutazione
- Eventuali lezioni di recupero/ripasso
- 1 verifica finale

10 Spazi

Tutte le lezioni sono composte da una parte teorica ed una pratica. Per questo motivo il laboratorio si rende necessario per tutte le lezioni. Per evitare che gli studenti si distraggano durante la lezione teorica, si ipotizza di svolgere queste parti in aula e durante la pausa spostarsi in laboratorio.
Per le lezioni teoriche è necessario un proiettore collegato ad un computer. Le lezioni che comprenderanno lo svolgimento dei progetti si svolgeranno in laboratorio. Nel caso in cui fosse possibile, gli studenti possono portare i proprio dispositivi ed utilizzarli in classe. Si incentiva inoltre l'utilizzo del raspberry [7], il quale integra perfettamente la libreria di Python PyGame che sarà utilizzata in tutte le lezioni. In questo caso è necessario avere a disposizione almeno 6/7 raspberry collegati a dei monitor in modo da creare gruppi da 3 nel quale gli studenti oltre a applicare la programmazione ad oggetti. Nel caso in cui non fosse possibile reperire dei raspberry l'esercizio finale può essere svolto in laboratorio con computer comuni.

11 Materiali e Strumenti

- Durante le lezioni saranno forniti agli studenti materiali prodotti dal docente (esercizi svolti, spiegazioni dettagliate e compiti da svolgere per la lezione successiva)

- Code visualizer online per la visualizzazione della macchina concettuale (`http://www.pythontutor.com/`)
- Qualsiasi IDE per l'ambiente di sviluppo, se è possibile si incentiva l'utilizzo di Raspberry Pi
- Risorse online (wikipedia, esercizi svolti, documentazione Python e PyGame)

12 Percorso didattico

12.1 Lezione 1

Prerequisiti

Conoscenza della programmazione di base
Proiettore e computer
Ide e libreria PyGame

Finalità

Gli studenti verranno introdotti al concetto della programmazione ad oggetti mediante una spiegazione teorica e pratica. Attraverso l'elaborazione della prima parte del videogioco gli studenti comprenderanno il concetto di classe e di istanza e saranno in grado di definire il concetto di istanza ed, inoltre, sapranno leggere il codice relativo alla programmazione ad oggetti in modo consapevole e critico. Si affronterà il concetto di istanza e variabile istanza al fine di far comprendere agli studenti la definizione di classe.

Fasi dell'attività didattica: spazi e tempi

1. Introduzione sul lavoro da svolgere nella lezione e sul percorso della creazione del videogioco (20 minuti)
2. Spiegazione teorica sui concetti base di classe ed istanza con esempi pratici basati sul videogioco (40 minuti)
3. Discussione della classe sugli oggetti (15 minuti)
4. Svolgimento in gruppi della prima parte del videogioco (40 minuti)

La prima parte della lezione può essere svolta in classe, la seconda parte in laboratorio.

Cosa ho programmato di fare

1. Il docente fornisce il materiale ed espone alla classe gli argomenti dell'unità didattica e della lezione 1, mostrerà la libreria PyGame e il funzionamento del python tutor.

2. Il docente introduce la programmazione ad oggetti mediante esempi concreti sul videogioco da creare nell'ora successiva. Nella spiegazione teorica si introducono i concetti base: creazione di una classe e definizione di istanza. Viene introdotto il concetto di costruttore. Vengono forniti esempi pratici sui concetti spiegati in precedenza, tali esempi potranno essere applicati nello svolgimento del videogioco. I concetti introdotti mediante il codice saranno visualizzati con l'utilizzo del code visualizer.

3. Il docente invita gli studenti a fornire esempi di classi ed oggetti relativi ad un videogioco anche illustrando le idee alla lavagna, al fine di comprendere se ci sono misconcezioni ed eventualmente chiarire alcuni dubbi.

4. Il docente chiede agli studenti di formare gruppi di massimo 3 persone e fornisce loro tutto il materiale utilizzato nella prima parte della lezione. Spiega agli studenti come consultare la libreria di PyGame e spiega loro come muovere i primi passi verso la prima parte del videogioco.

Cosa prevedo facciano gli alunni

1. Gli studenti raccolgono il materiale del professore ed ascoltano l'introduzione.

2. Gli studenti ascoltano la spiegazione. In caso di dubbi chiedono chiarimenti e spiegazioni al docente.

3. Gli studenti forniscono esempi pratici di programmazione ad oggetti, anche mediante l'utilizzo della lavagna.

4. Gli studenti creano gruppi e cominciano a programmare, sulla base della spiegazione precedente e sul materiale fornito dal professore, il videogioco. Muovono i primi passi con PyGame ed applicano concretamente gli esempi, modificando gli oggetti e i nomi e creando il proprio videogioco. Leggono il codice fornito dal professore e cercano di interpretare il funzionamento del codice chiedendo al docente di spiegare con maggiore accuratezza le parti meno chiare.

Materiali e strumenti

Verranno forniti materiali inerenti alla lezione con relativo codice ed esempi, verranno forniti anche materiali supplementari online prodotti dal docente (video spiegazioni e esempi con codice)

Attività alternative / approfondimenti

La lezione introduttiva può essere approfondita con svariati esempi sugli oggetti e sull'uso della programmazione ad oggetti nelle applicazioni, al fine di far comprendere agli studenti l'importanza e l'utilità dell'argomento.

Eventuali compiti a casa

Sarà chiesto agli studenti di svolgere un esercizio composto da tre problemi da risolvere e da consegnare la settimana successiva. Gli studenti sono invitati a studiare il materiale prodotto dal docente ed integrare (non obbligatorio) il videogioco con nuove funzionalità.

In caso di studenti assenti

Gli studenti assenti saranno invitati a studiare il materiale fornito dal professore ed a creare il proprio videogioco (la parte iniziale) a casa. In caso di dubbi potranno chiedere spiegazioni nella lezione successiva.

12.2 Lezione 2

Prerequisiti

Aver studiato il materiale fornito nella lezione 1 ed aver svolto la prima parte del videogioco
Proiettore e computer
Necessario un qualsiasi ide per python e un browser web per il code visualizer

Finalità

Mediante l'elaborazione della seconda parte del videogioco gli studenti comprenderanno il concetto di creazione di diverse istanze di una stessa classe o di classi diverse. Si affronterà inoltre la parola chiave self, i costruttori e il concetto della visibilità delle variabili istanza e metodi istanza di una classe e l'accesso e modifica di quest'ultimi. La prima parte della lezione ha lo scopo di introdurre i concetti

in modo teorico anche mediante l'utilizzo di codice ed esempi pratici relativi al videogioco. I concetti introdotti mediante il codice saranno visualizzati con l'utilizzo del code visualizer. Nella seconda parte della lezione gli studenti formeranno gruppi per l'elaborazione della seconda parte del videogioco, attraverso l'approccio del collaborative learning gli studenti, si confrontano tra di loro collaborando al fine di trasformare l'apprendimento in un processo attivo e coinvolgente [4]. Lo scopo del videogioco è, oltre a quello di suscitare l'interesse degli studenti, quello di evitare che gli studenti creino misconcezioni riguardo concetti della programmazione ad oggetti. Inoltre la collaborazione tra studenti crea anche competizione tra gruppi, creando uno spirito giocoso con la finalità di far partecipare attivamente tutti gli studenti della classe [3].

Fasi dell'attività didattica: spazi e tempi

1. Introduzione agli argomenti della lezione (15 minuti)
2. Spiegazione teorica sul concetto di multiple istanze, parola self, costruttori, accesso e modifica di variabili e metodi istanza con esempi pratici basati sul videogioco (40 minuti)
3. Svolgimento in gruppi della seconda parte del videogioco (40 minuti)
4. Discussione sui progetti (20 minuti)

La prima parte della lezione può essere svolta in classe, la seconda parte in laboratorio.

Cosa ho programmato di fare

1. Il docente fornisce il materiale ed espone alla classe gli argomenti della lezione.
2. Il docente introduce i nuovi concetti attraverso una lezione teorica composta anche da esercizi svolti ed esempi pratici su oggetti del videogioco.
3. Il docente invita gli studenti a ricomporre i gruppi ed iniziare ad applicare i concetti spiegati alla seconda parte del videogioco. Invita inoltre gli studenti ad implementare il proprio videogioco in maniera creativa utilizzando gli esempi forniti e avvalendosi dell'aiuto della documentazione di PyGame presente su internet.
4. Il docente chiede agli studenti di esporre le loro idee progettuali ed i principali problemi avuti e li aiuta a risolverli davanti alla classe.

Cosa prevedo facciano gli alunni

1. Gli studenti raccolgono il materiale del professore.
2. Gli studenti ascoltano la spiegazione. In caso di dubbi chiedono chiarimenti e spiegazioni al docente.
3. Gli studenti in gruppi integrano il codice del proprio gioco sulla base della spiegazione fatta dal professore e sulla base del materiale fornito in precedenza. Leggono il materiale cercano di interpretare il funzionamento del codice chiedendo al docente di spiegare con maggiore accuratezza le parti meno chiare. Implementano nuove funzionalità del videogioco con l'aiuto della documentazione di PyGame su internet.
4. Gli studenti, un gruppo alla volta, espongono i principali problemi avuti e presentano le loro idee di implementazione.

Materiali e strumenti

Verranno forniti materiali inerenti alla lezione con relativo codice ed esempi, verranno forniti anche materiali supplementari online prodotti dal docente (video spiegazioni e esempi con codice).

Attività alternative / approfondimenti

Gli studenti sono invitati ad approfondire il progetto svolto in aula integrandolo a casa con nuove funzionalità (non obbligatorio). Nel materiale fornito dal docente saranno indicati siti sui quali gli studenti possono approfondire gli argomenti trattati nella lezione. Sarà, inoltre, cura del docente fornire materiale autoprodotto al fine di fornire un maggior numero di esempi pratici e spiegazioni esaustive (anche mediante videolezioni).

Eventuali compiti a casa

Per la lezione successiva sarà richiesto agli alunni di ripassare e studiare il materiale prodotto dal docente. Verranno assegnati esercizi da svolgere a casa e da consegnare al docente.

In caso di studenti assenti

Gli studenti assenti saranno invitati a studiare il materiale fornito dal professore e in caso di dubbi potranno chiedere spiegazioni nella lezione successiva. Potranno

inoltre, riprendere il progetto ed aggiungere nuove funzionalità in linea con gli argomenti trattati nella seguente lezione.

12.3 Lezione 3

Prerequisiti

Aver studiato il materiale fornito nella lezione 2 ed aver svolto la seconda parte del videogioco
Proiettore e computer
Necessario un qualsiasi ide per python e un browser web per il code visualizer

Finalità

La terza lezione ha lo scopo di consolidare gli argomenti trattati ed introdurre nuovi argomenti come la differenza tra variabili e metodi di istanza e variabili e metodi di classe. Verranno trattati gli argomenti sulle differenze tra variabili e metodi di istanza e di classe .La prima parte della lezione ha lo scopo di introdurre gli argomenti in modo teorico anche mediante l'utilizzo di codice ed esempi pratici relativi al videogioco. I concetti introdotti mediante il codice saranno visualizzati con l'utilizzo del code visualizer. Nella seconda parte della lezione gli studenti, in gruppi da 3, applicheranno i concetti introdotti dal professore nel videogioco integrando il lavoro fatto fino ad allora.

Fasi dell'attività didattica: spazi e tempi

1. Introduzione alla lezione (15 minuti)
2. Spiegazione teorica sulle differenze tra variabili e metodi di istanza e di classe con esempi pratici basati sul videogioco (40 minuti)
3. Svolgimento in gruppi della terza parte del videogioco (40 minuti)
4. Discussione sui progetti (20 minuti)

La prima parte della lezione può essere svolta in classe, la seconda parte in laboratorio.

Cosa ho programmato di fare

1. Il docente fornisce il materiale ed espone alla classe gli argomenti della lezione.

2. Il docente introduce i nuovi concetti sulle differenze tra variabili e metodi di istanza e di classe attraverso una lezione teorica composta anche da esercizi svolti ed esempi pratici su oggetti del videogioco.

3. Il docente invita gli studenti a ricomporre i gruppi ed iniziare ad applicare i concetti spiegati alla terza parte del videogioco. Il docente può decidere di scambiare i componenti dei gruppi sulla base dell'andamento del progetto.

4. Il docente chiede agli studenti di esporre le loro idee progettuali ed i principali problemi avuti e li aiuta a risolverli davanti alla classe mostrando anche il codice svolto.

Cosa prevedo facciano gli alunni

1. Gli studenti raccolgono il materiale del professore ed ascoltano l'introduzione della lezione.

2. Gli studenti ascoltano la spiegazione. In caso di dubbi chiedono chiarimenti e spiegazioni al docente.

3. Gli studenti in gruppi integrano il codice del proprio gioco sulla base della spiegazione fatta dal professore e sulla base del materiale.

4. Gli studenti espongono i principali problemi avuti e presentano le loro idee di implementazione alla classe.

Materiali e strumenti

Verranno forniti materiali inerenti alla lezione con relativo codice ed esempi, verranno forniti anche materiali supplementari online prodotti dal docente (video spiegazioni ed esempi con codice).

Attività alternative / approfondimenti

Gli studenti sono invitati ad approfondire il progetto svolto in aula integrandolo a casa con nuove funzionalità (non obbligatorio). Nel materiale fornito dal docente saranno indicati siti sui quali gli studenti possono approfondire gli argomenti trattati nella lezione.

Eventuali compiti a casa

Per la lezione successiva sarà richiesto agli alunni di studiare il materiale fornito dal docente. Verranno assegnati 3 esercizi da svolgere a casa e da consegnare al docente per la lezione successiva.

In caso di studenti assenti

Gli studenti assenti saranno invitati a studiare il materiale fornito dal professore e in caso di dubbi potranno chiedere spiegazioni nella lezione successiva. Potranno inoltre, riprendere il progetto ed aggiungere nuove funzionalità in linea con gli argomenti trattati nella seguente lezione.

12.4 Lezione 4

Prerequisiti

Aver studiato il materiale fornito nella lezione 3 ed aver svolto la terza parte del videogioco
Proiettore e computer
Necessario un qualsiasi ide per python e un browser web per il code visualizer

Finalità

Nella quarta lezione gli studenti applicheranno mediante lo sviluppo del proprio videogioco il concetto di ereditarietà. Lo scopo di questa lezione è insegnare loro ad applicare l'ereditarietà a problemi la cui soluzione richiede questo argomento. La prima parte della lezione ha lo scopo di introdurre i concetti in modo teorico anche mediante l'utilizzo di codice ed esempi pratici relativi al videogioco. I concetti introdotti mediante il codice saranno visualizzati con l'utilizzo del code visualizer. Nella seconda parte della lezione gli studenti formeranno gruppi per l'elaborazione della quarta parte del videogioco, al fine di applicare concretamente il concetto di ereditarietà.

Fasi dell'attività didattica: spazi e tempi

1. Introduzione sugli argomenti da trattare nella lezione e sulle modalità dello svolgimento della quarta parte del videogioco (15 minuti)

2. Spiegazione teorica sull'ereditarietà con esempi pratici basati sul videogioco (40 minuti)

3. Svolgimento quarta parte del videogioco (40 minuti)

4. Discussione sui progetti (20 minuti)

La prima parte della lezione può essere svolta in classe, la seconda parte in laboratorio.

Cosa ho programmato di fare

1. Il docente fornisce il materiale ed espone alla classe gli argomenti della lezione.

2. Il docente introduce i nuovi concetti sull'ereditarietà attraverso una lezione teorica composta anche da esercizi svolti ed esempi pratici.

3. Il docente invita gli studenti a ricomporre i gruppi ed iniziare ad applicare i concetti spiegati alla quarta parte del videogioco.

4. Il docente chiede agli studenti di esporre le loro idee progettuali ed i principali problemi avuti e li aiuta a risolverli davanti alla classe.

Cosa prevedo facciano gli alunni

1. Gli studenti raccolgono il materiale del professore.

2. Gli studenti ascoltano la spiegazione. In caso di dubbi chiedono chiarimenti e spiegazioni al docente.

3. Gli studenti in gruppi integrano il codice del proprio gioco sulla base della spiegazione fatta dal professore e sulla base del materiale, per aggiungere nuove funzionalità consultano la documentazione di PyGame e in caso di dubbi chiedono aiuto al docente e nei minuti finali espongono i problemi avuti alla classe.

4. Espongono le difficoltà avute durante lo svolgimento del videogioco e chiedono chiarimenti al professore mostrando il loro codice alla classe in modo che chiunque possa partecipare.

Materiali e strumenti

Verranno forniti materiali inerenti alla lezione con relativo codice ed esempi, verranno forniti anche materiali supplementari online prodotti dal docente (video spiegazioni ed esempi con codice).

Attività alternative / approfondimenti

Gli studenti sono invitati ad approfondire il progetto svolto in aula integrandolo a casa con nuove funzionalità (non obbligatorio). Nel materiale fornito dal docente saranno indicati siti sui quali gli studenti possono approfondire gli argomenti trattati nella lezione.

Eventuali compiti a casa

Per la lezione successiva sarà richiesto agli alunni di studiare il materiale prodotto dal docente. Verranno assegnati 3 esercizi da svolgere a casa e da consegnare al docente per la lezione successiva.

In caso di studenti assenti

Gli studenti assenti saranno invitati a studiare il materiale fornito dal professore e in caso di dubbi potranno chiedere spiegazioni nella lezione successiva.

12.5 Lezione 5

Prerequisiti

Aver studiato il materiale fornito nella lezione 4 ed aver svolto la quarta parte del videogioco
Proiettore e computer
Necessario un qualsiasi ide per python e un browser web per il code visualizer

Finalità

La quinta ed ultima lezione prima della presentazione dei progetti ha lo scopo di introdurre gli argomenti dei metodi statici e del polimorfismo. Gli studenti concluderanno il progetto di gruppo e prepareranno la presentazione per la lezione successiva.

Fasi dell'attività didattica: spazi e tempi

1. Introduzione agli argomenti della lezione odierna (15 minuti)
2. Spiegazione teorica sui metodi statici e sui concetti base del polimorfismo con esempi pratici basati sul videogioco (40 minuti)
3. Svolgimento parte finale del videogioco (1 ora)

La prima parte della lezione può essere svolta in classe, la seconda parte in laboratorio.

Cosa ho programmato di fare

1. Il docente fornisce il materiale ed espone alla classe gli argomenti della lezione.

2. Il docente introduce i nuovi concetti sui metodi statici e sui concetti base del polimorfismo attraverso una lezione teorica composta anche da esercizi svolti ed esempi pratici sul videogioco "Il mondo di Bob".

3. Il docente invita gli studenti a ricomporre i gruppi ed iniziare ad applicare i concetti spiegati alla quarta parte del videogioco.

Cosa prevedo facciano gli alunni

1. Gli studenti raccolgono il materiale del professore.

2. Gli studenti ascoltano la spiegazione. In caso di dubbi chiedono chiarimenti e spiegazioni al docente.

3. Gli studenti in gruppi integrano il codice del proprio gioco sulla base della spiegazione fatta dal professore e sulla base del materiale.

Materiali e strumenti

Verranno forniti materiali inerenti alla lezione con relativo codice ed esempi, verranno forniti anche materiali supplementari online prodotti dal docente (video spiegazioni ed esempi con codice).

Attività alternative / approfondimenti

Gli studenti sono invitati ad approfondire il progetto svolto in aula integrandolo a casa con nuove funzionalità (non obbligatorio). Nel materiale fornito dal docente saranno indicati siti sui quali gli studenti possono approfondire gli argomenti trattati nella lezione.

Eventuali compiti a casa

Per la lezione successiva sarà richiesto agli alunni di creare slide riassuntive sul progetto svolto e di preparare una presentazione di gruppo.

In caso di studenti assenti

Gli studenti assenti saranno invitati a studiare il materiale fornito dal professore e in caso di dubbi potranno chiedere spiegazioni nella lezione successiva. Dovranno inoltre preparare la presentazione sul progetto svolto anche se incompleto.

12.6 Lezione 6

Prerequisiti

Lavagna e proiettore
Aver concluso il videogioco

Finalità

Gli studenti attraverso l'esposizione del proprio videogioco si confrontano con la classe e con il professore. La presentazione davanti alla classe ha lo scopo di preparare gli studenti ad esporre, anche attraverso semplici slide, il lavoro svolto in modo da acquisire padronanza dell'argomento. Gli altri studenti che in quel momento non presentano possono intervenire proponendo una loro soluzione e allo stesso tempo confrontano, anche se indirettamente, il loro lavoro con quello degli altri.

Fasi dell'attività didattica: spazi e tempi

1. Esposizione videogiochi (2 ore)

La lezione sarà interamente svolta in aula.

Cosa ho programmato di fare

1. Correggere alla classe gli eventuali errori e commentare le scelte progettuali.

Cosa prevedo facciano gli alunni

1. Ciascun gruppo espone il videogioco attraverso poche slide nelle quali riassume in punti ciò che è stato fatto, gli studenti sono invitati a partecipare e a fare commenti. In caso di dubbi gli studenti chiedono chiarimenti sia agli studenti che espongono sia al docente che risponde ai quesiti ed eventualmente corregge gli errori.

Materiali e strumenti

Per questa lezione non sono previsti materiali. Gli studenti presenteranno il videogioco attraverso slide sul computer della classe.

Attività alternative / approfondimenti

Gli studenti sono invitati a modificare eventuali errori o ad aggiungere nuove features al videogioco.

Eventuali compiti a casa

Non saranno assegnati nuovi compiti, gli studenti dovranno eventualmente modificare il codice sulla base di errori riscontrati nella presentazione dal professore o dalla classe. Gli studenti dovranno prepararsi per la verifica finale.

13 Valutazione

Gli studenti saranno valutati sulla base della loro capacità di esprimere, mediante esercizi pratici di programmazione e domande teoriche, di aver compreso i concetti base della programmazione ad oggetti. Essendo questo un argomento di programmazione non verranno effettuate interrogazioni orali, ma si valuterà il percorso dello studente anche attraverso l'esposizione orale finale del videogioco. La sufficienza si otterrà mostrando di aver compreso gli esercizi base dell'argomento. Progressivamente la valutazione migliorerà in base alla qualità sintattica del codice e alle soluzioni più efficienti.

13.1 Valutazione in itinere / formativa / per l'apprendimento

Il docente terrà conto dei progressi dei singoli studenti sulla base del loro percorso durante le lezioni. Una verifica in itinere verrà svolta senza valutazione per permettere al docente di capire l'andamento della classe e correggere eventuali misconcezioni venutesi a creare in alcuni studenti. Mediante la verifica in itinere il docente sarà in grado di capire quali argomenti risultano più difficili e potrà modificare le lezioni sulla base degli errori degli studenti. La valutazione in itinere è dunque uno strumento molto importante per il docente anche per individuare gli studenti con maggiori difficoltà ed eventualmente concentrare parte delle lezioni a incentivare questi studenti ad esporre i problemi sorti durante lo svolgimento degli esercizi e dei progetti.

13.2 Valutazione sommativa / finale / dell'apprendimento

Si prevede una verifica finale con voto. Gli studenti potranno studiare dal materiale prodotto dal docente e dagli esercizi svolti durante il percorso e potranno anche consultare il libro di testo. La valutazione della verifica sarà composta da brevi esercizi da risolvere e domande teoriche. La valutazione finale terrà conto sia della verifica che del percorso dello studente.

Bibliografia

[1][Papert, S., 1986] *Constructionism: A new opportunity for elementary science education.* NSF Grant Application.

[2][Yan, Lu. 2009] *Teaching Object-Oriented Programming with Games.* ITNG 2009 - 6th International Conference on Information Technology: New Generations.

[3][W. Chen and Y. C. Cheng 2007] *"Teaching Object-Oriented Programming Laboratory With Computer Game Programming,"* in IEEE Transactions on Education, vol. 50, no. 3, pp. 197-203

[4][Yulia and R. Adipranata, 2010], *"Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment,"* 2nd International Conference on Education Technology and Computer, Shanghai.

[5][Umberto Costantini, 2018] *"Apprendimento della programmazione: analisi delle difficolta e sviluppo di strategie didattiche".* Tesi di laurea

[6][Lodi, 2014] *Imparare il pensiero computazionale, imparare a programmare.* Tesi di laurea, Università di Bologna, 2014.

[7][Davoli, 2014] *Solo Software Libero nella scuola!* 2014. `http://aptiva.v2.cs.unibo.it/wiki/index.php/Solo_Software_Libero_nella_scuola!`.

A Appendice: Materiali didattici per gli studenti

A.1 (Lezione 1) Codice fornito dal professore per la realizzazione della prima parte del videogioco

```
1
2  import pygame
3  import random
4  #grandezze espresse in pixel
5  LARGHEZZA = 800
6  ALTEZZA = 600
7  #Codici rgb
8  BIANCO = (255, 255, 255)
9  BLU = (0, 0, 255)
10 ROSSO = (255, 0, 0)
11
12 game_display = pygame.display.set_mode((LARGHEZZA,ALTEZZA))
13 pygame.display.set_caption('Il mondo di Bob')
14 clock = pygame.time.Clock()
15
16
17 class Giocatore:
18     #Costruttore
19     def __init__(self, nome, sesso):
20         self.x = random.randrange(0, LARGHEZZA)
21         self.y = random.randrange(0, ALTEZZA)
22         self.nome = nome
23         self.sesso = sesso
24
25     #metodo per mostrare sullo schermo l'immagine dell'istanza
26     def mostra(self, screen, immagine):
27         screen.blit(immagine, (self.x, self.y))
28
29
30 class Ostacolo:
31     #Costruttore
32     def __init__(self, numero, pericolosit , movimento):
33         self.x = random.randrange(0, LARGHEZZA)
34         self.y = random.randrange(0, ALTEZZA)
35         self.numero = numero
36         self.pericolosit  = pericolosit
37         self.movimento = movimento
38
39
40     #metodo per mostrare sullo schermo l'immagine dell'istanza
41     def mostra(self, screen, immagine):
```

```
42          screen.blit(immagine, (self.x, self.y))
43
44
45
46  def start():
47      #creo istanza bob ed istanza ostacolo1 appartenenti
48      #rispettivamente alle classi Giocatore ed Ostacolo
49      bob = Giocatore('Bob', 'M')
50      ostacolo1 = Ostacolo(1,5,False)
51
52      #Carico le immagini da attribuire alle istanze
53      bob_img = pygame.image.load('bob.png')
54      ostacolo1_img = pygame.image.load('ostacolo1.png')
55
56
57      while True:
58          for event in pygame.event.get():
59              if event.type == pygame.QUIT:
60                  pygame.quit()
61                  quit()
62          #definisco lo sfondo del videogioco del colore bianco
63          game_display.fill(BIANCO)
64          #mostro sullo schermo bob e l'ostacolo
65          bob.mostra(game_display, bob_img)
66          ostacolo1.mostra(game_display, ostacolo1_img)
67          #richiamo il metodo muovi della classe Giocatore per
    muovere bob
68          pygame.display.update()
69          #definisco gli fps
70          clock.tick(60)
71
72
73  start()
```

Listing 1: Codice fornito dal professore per la realizzazione della prima parte del videogioco

A.2 (Lezione 2) Materiale del professore

Le classi, come abbiamo visto nella lezione precendente, permettono di includere variabili e funzioni in modo tale da essere riutilizzabili attraverso la creazione di istanze. Abbiamo creato la classe Giocatore con delle variabili istanza (x, y, nome, sesso) ed una classe Ostacolo con le variabili istanza (x, y, numero, pericolosità, movimento).

```
class Giocatore:
    #Costruttore
    def __init__(self, nome, sesso):
        self.x = random.randrange(0, LARGHEZZA)
        self.y = random.randrange(0, ALTEZZA)
        self.nome = nome
        self.sesso = sesso
```

Listing 2: Classe Giocatore

```
class Ostacolo:
    #Costruttore
    def __init__(self, numero, pericolosit , movimento):
        self.x = random.randrange(0, LARGHEZZA)
        self.y = random.randrange(0, ALTEZZA)
        self.numero = numero
        self.pericolosit  = pericolosit
        self.movimento = movimento
```

Listing 3: Classe Ostacolo

Abbiamo dunque creato l'istanza della classe Giocatore, denominandola Bob. Abbiamo poi creato l'istanza della classe Ostacolo, creando un vero e proprio ostacolo posizionato nel mondo di Bob.

```
    #creo istanza bob ed istanza ostacolo1 appartenenti
    #rispettivamente alle classi Giocatore ed Ostacolo
    bob = Giocatore('Bob', 'M')
    ostacolo1 = Ostacolo(1,5,False)
```

Listing 4: Istanze delle classi

Applicandolo al videogioco con la libreria PyGame abbiamo mostrato graficamente le due istanze.

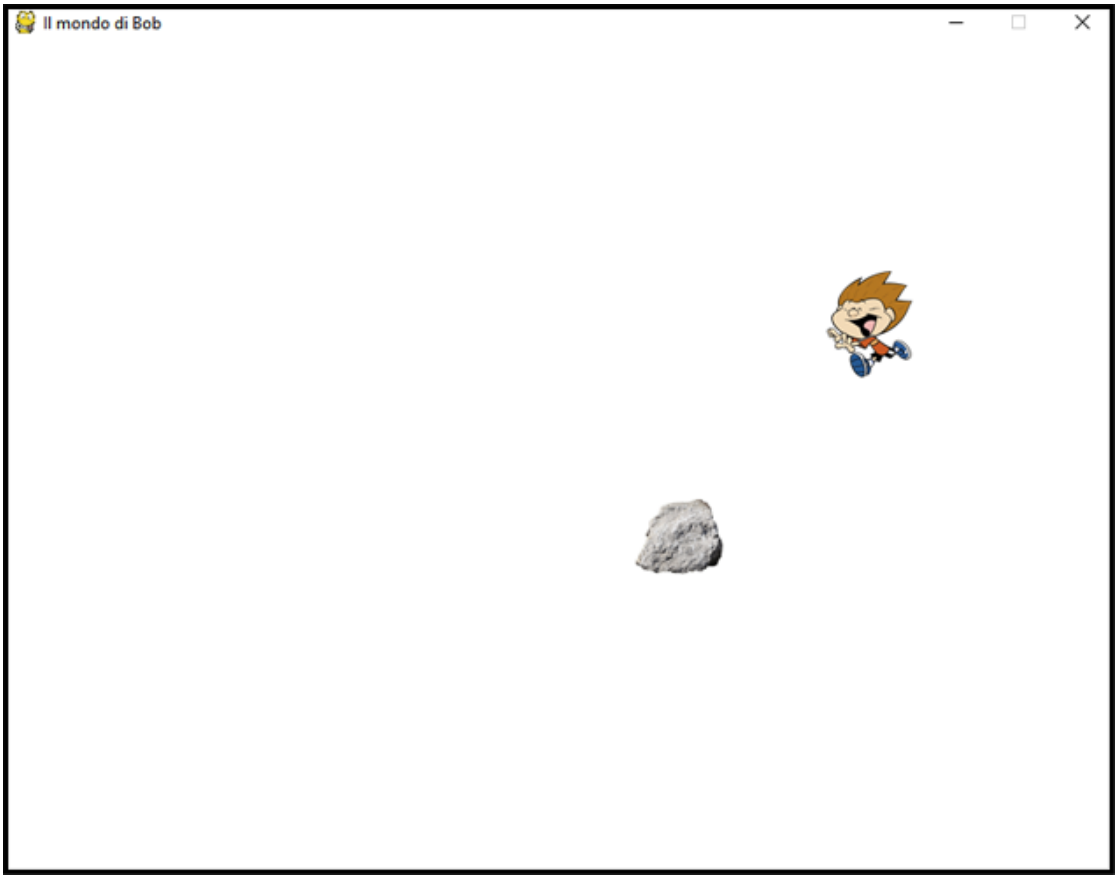

Figura 1: Videgioco il mondo di Bob

In questa lezione introduciamo alcuni concetti più avanzati sulle classi. Quando è stato introdotto il costruttore si è parlato della parola chiave self, vediamo nello specifico a cosa si riferisce. Nella lezione precedente si è affermato che all'interno di una classe non possono coesistere due variabili che hanno lo stesso nome, di conseguenza anche all'interno di un metodo non possono esserci due variabili dello stesso nome. Tuttavia se una di queste variabili è una variabile istanza questo è possibile.
La classe giocatore ha infatti le variabili istanza nome e sesso e queste sono precedute dalla parole chiave self (letteralmente "se stesso") in quanto rappresenta l'oggetto che riceve l'invocazione del costruttore, mentre i parametri sono le variabili alle quali verranno associate le variabili istanza. Un costruttore è dunque un metodo che viene invocato quando viene creato un oggetto di una classe. Un costruttore fornisce il valore iniziale delle variabili di istanza dell'oggetto appena creato.
Essendo dunque una classe come uno stampo per la costruzione di oggetti (istan-

ze) si possono creare multiple istanze di una stessa classe, vediamo un esempio. Nel mondo di Bob potremmo aggiungere un nuovo Player un amico ad esempio di nome Tom creando una nuova istanza della classe Giocatore in questo modo:

```
1     bob = Giocatore('Bob', 'M')
2     tom = Giocatore('Tome', 'M')
```

Listing 5: Due istanze della classe Giocatore

Oppure è possibile creare molti ostacoli nel mondo di bob, creando una lista che li memorizza. Ad esempio possiamo creare una lista lista_ostacoli che raggruppa tutti gli oggetti (istanze di tipo ostacolo):

```
1     lista_ostacoli = []
2     for i in range(20):
3         lista_ostacoli.append(Ostacolo(i,10,True))
```

Listing 6: Multiple (20) istanze della classe ostacolo

In questo modo possiamo mostrale sul videogioco per verificare la creazione delle diverse istanze, Mostrando graficamente le istanze possiamo notare le 20 istanze della classe ostacolo nel mondo di Bob:

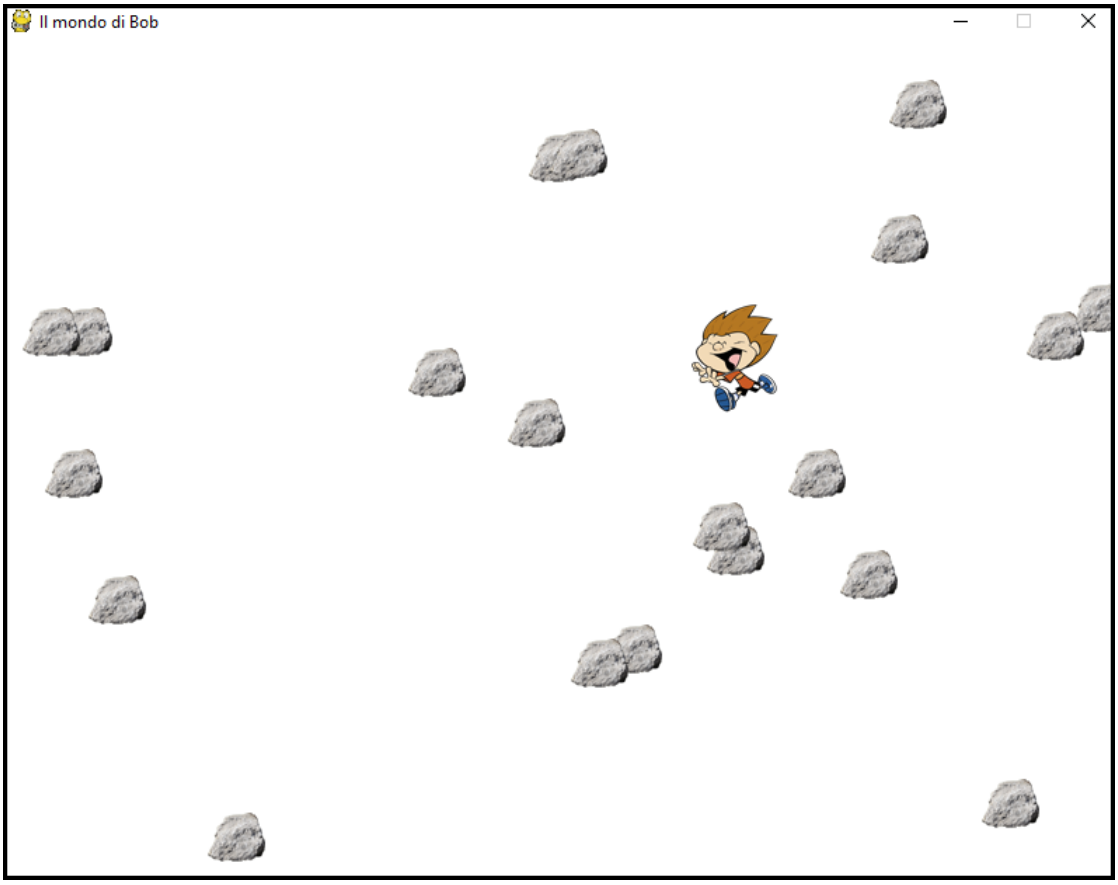

Figura 2: Multiple (20) istanze della classe ostacolo mostrate graficamente

Come introdotto nella prima lezione un metodo è un insieme di istruzioni con un nome la cui invocazione comporta l'esecuzione delle istruzioni in esso definite. Un metodo istanza è un metodo che viene invocato su un oggetto è che può manipolare lo stato dell'oggetto stesso.Un metodo di istanza definito in una classe viene invocato usando un oggetto di quella classe. L'oggetto prende il nome di oggetto chiamante (calling object). L'invocazione viene effettuata scrivendo il nome dell'oggetto, per esempio bob, seguito da un punto e dal nome del metodo, per esempio bob.muovi() ed infine da una coppia di parentesi che possono contenere i parametri per il metodo. Tutte le definizioni di metodo istanza compaioni della definizione della classe alla quale appartengono. Questi metodi infatti possono essere usati solo con oggetti della classe in cui il metodo è definito.

```
def muovi(self):
        #nella variabile key_input memorizza il pulsante premuto
        key_input = pygame.key.get_pressed()
        #sposta l'istanza di 5 pixel
```

```
5           if key_input[pygame.K_LEFT]:
6               self.x -= 5
7           if key_input[pygame.K_UP]:
8               self.y -= 5
9           if key_input[pygame.K_RIGHT]:
10              self.x += 5
11          if key_input[pygame.K_DOWN]:
12              self.y += 5
```

Listing 7: Metodo istanza muovi() della classe Giocatore

Allo stesso modo possiamo mettere in movimento anche gli oggetti ostacolo, definendo un movimento random.

```
1 def muovi(self):
2           if self.movimento:
3               self.x += random.randrange(-4,5)
4               self.y += random.randrange(-4,5)
5
6               #controllo per evitare che si muovano oltre i bordi
7               if self.x < 0: self.x = 0
8               elif self.x > LARGHEZZA: self.x = LARGHEZZA
9
10              if self.y < 0: self.y = 0
11              elif self.y > ALTEZZA: self.y = ALTEZZA
12          else:
13              print("L'ostacolo non si pu  muovere")
```

Listing 8: Metodo istanza muovi() della classe Ostacolo

Vediamo ora come accedere ai valori delle variabili istanza di un oggetto. Abbiamo visto come un oggetto di una classe è un elemento contenente al proprio interno una serie di variabili di istanza. Le variabili istanze in python sono accessibili e modibicabili anche esternamente alla definizione della classe. Bisogna porre dunque molta attenzione quando si modifica o accede ad una variabile di una istanza in quanto è possibile modificarla in modo molto semplici attraverso la sintassi nomeoggetto.variabileIstanza.
Facciamo un esempio pratico, preso il nostro esempio di Bob vogliamo mostrare a video la posizione di Bob durante lo spostamento. Quello che dobbiamo fare è prendere i valori delle coordinate x ed y dell'istanza Bob e mostrarla a video. Per fare ciò creaiamo due variabili posizioneBob_x e posizioneBob_y nelle quali memorizziamo le posizioni delle coordinate x ed y di bob.

```
1           posizioneBob_x = bob.x
2           posizioneBob_y = bob.y
```

Listing 9: Accesso al valore delle variabili istanza

Possiamo ora mostrare in tempo reale le coordinate di bob nella mappa stampando i valori delle variabili memorizzate:

```
1         #mostro sulla schermo le coordinate di Bob
2         textsurface = myfont.render(f'Bob si trova nelle
    coordinate: x: {str(posizioneBob_x)} y: {str(posizioneBob_y)}',
    False, (0, 0, 0))
3         game_display.blit(textsurface,(450,550))
```

Listing 10: Codice per mostrare sulla mappa le coordinate di Bob

Vediamo ora il risultato finale:

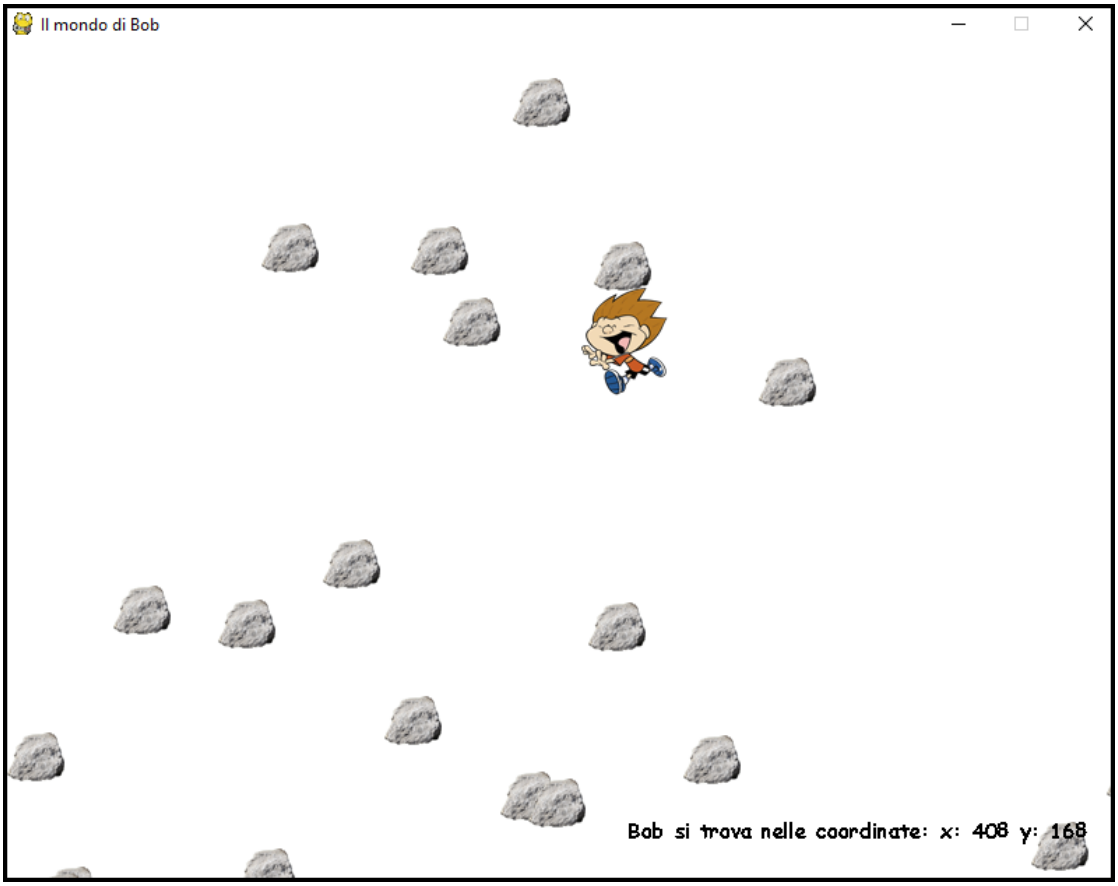

Figura 3: Coordinate di bob mostrate graficamente in modo dinamico

Nella lezione di oggi abbiamo visto nello specifico il concetto di classe che possiede delle variabili istanza per memorizzare dati e definizioni di metodi che eseguono azioni. Abbiamo poi visto cosa è un costruttore, ovvero un metodo che,

invocato, crea ed inizializza un oggetto di una classe. Le variabili istanza e i metodi istanza possono essere usati da qualunque posizione. Abbiamo poi visto nello specifico la parola chiave self , la quale quando è usata all'interno della definizione di classe rappresenta l'oggetto che riceve l'invocazione di un metodo. Sulla base di questi concetti teorici abbiamo poi creato un videogioco interattivo composto da un giocatore Bob (istanza della classe Giocatore) ed una serie di ostacoli (20 istanze della classe Ostacolo) che si muovono nel mondo di Bob ed ostacolano il suo movimento nello spazi. Nelle prossime lezione parleremo di variabili di classe, le quali a differenza delle variabili istanza sono attributi che vengono condivisi da tutte le Istanze della Classe. Parleremo poi dei metodi di classe, metodi che permettono di passare come parametro la Classe invece dell'Istanza.

Riassunto in punti dei concetti trattati nella lezione di oggi:

- Le classi hanno variabili di istanza per memorizzare dati e definizioni di metodi che eseguono azioni.
- Un costruttore è un metodo che, invocato, crea ed inizializza un oggetto di una classe.
- Si possono creare diverse istanze di una stessa classe, con parametri uguali o diversi.
- La parola self, quando usata all'interno di una definizione di un metodo (incluso il costruttore), rappresenta l'oggetto che riceve l'invocazione di metodo.
- Le variabili istanza e i metodi istanza possono essere usati (modifica ed accesso) da qualunque posizione.

Di seguito il codice completo del videogioco creato fino ad ora:

```
import pygame
import random

#grandezze espresse in pixel
LARGHEZZA = 800
ALTEZZA = 600
#Codici rgb
BIANCO = (255, 255, 255)
BLU = (0, 0, 255)
ROSSO = (255, 0, 0)
#impostazioni di gioco (schermo e clock)
game_display = pygame.display.set_mode((LARGHEZZA,ALTEZZA))
pygame.display.set_caption('Il mondo di Bob')
clock = pygame.time.Clock()
```

```
15 class Giocatore:
16     #Costruttore
17     def __init__(self, nome, sesso):
18         self.x = random.randrange(0, LARGHEZZA)
19         self.y = random.randrange(0, ALTEZZA)
20         self.nome = nome
21         self.sesso = sesso
22 
23 
24     def muovi(self):
25         #nella variabile key_input memorizza il pulsante premuto
26         key_input = pygame.key.get_pressed()
27         #sposta l'istanza di 5 pixel
28         if key_input[pygame.K_LEFT]:
29             self.x -= 5
30         if key_input[pygame.K_UP]:
31             self.y -= 5
32         if key_input[pygame.K_RIGHT]:
33             self.x += 5
34         if key_input[pygame.K_DOWN]:
35             self.y += 5
36 
37     #metodo per mostrare sullo schermo l'immagine dell'istanza
38     def mostra(self, screen, immagine):
39         screen.blit(immagine, (self.x, self.y))
40 
41 class Ostacolo:
42     #Costruttore
43     def __init__(self, numero, pericolosit , movimento):
44         self.x = random.randrange(0, LARGHEZZA)
45         self.y = random.randrange(0, ALTEZZA)
46         self.numero = numero
47         self.pericolosit  = pericolosit
48         self.movimento = movimento
49 
50     def muovi(self):
51         if self.movimento:
52             #movimento random da -4 a 4 pixel
53             self.x += random.randrange(-4,5)
54             self.y += random.randrange(-4,5)
55 
56             #controllo per evitare che si muovano oltre i bordi
57             if self.x < 0: self.x = 0
58             elif self.x > LARGHEZZA: self.x = LARGHEZZA
59 
60             if self.y < 0: self.y = 0
61             elif self.y > ALTEZZA: self.y = ALTEZZA
62         else:
63             print("L'ostacolo non si pu  muovere")
```

```
64
65      #metodo per mostrare sullo schermo l'immagine dell'istanza
66      def mostra(self, screen, immagine):
67          screen.blit(immagine, (self.x, self.y))
68
69
70
71  def start():
72      #creo istanza bob ed istanze ostacolo appartenenti
73      #rispettivamente alle classi Giocatore ed Ostacolo
74      bob = Giocatore('Bob', 'M')
75      lista_ostacoli = []
76      for i in range(20):
77          lista_ostacoli.append(Ostacolo(i,10,True))
78
79      #Carico le immagini da attribuire alle istanze
80      bob_img = pygame.image.load('bob.png')
81      ostacolo1_img = pygame.image.load('ostacolo1.png')
82      #definisco il font per il testo
83      pygame.font.init()
84      myfont = pygame.font.SysFont('Comic Sans MS', 15, True)
85
86
87      while True:
88          for event in pygame.event.get():
89              if event.type == pygame.QUIT:
90                  pygame.quit()
91                  quit()
92          #definisco lo sfondo del videogioco del colore bianco
93          game_display.fill(BIANCO)
94          #mostro sullo schermo bob e l'ostacolo
95          bob.mostra(game_display, bob_img)
96          #mostro sulla mappa tutti gli ostacoli e li metto in
    movimento
97          for i in lista_ostacoli:
98              i.mostra(game_display, ostacolo1_img)
99              i.muovi()
100         #richiamo il metodo muovi della classe Giocatore per
    muovere bob
101         bob.muovi()
102         #variabili che memorizzano le coordinate di bob
103         posizioneBob_x = bob.x
104         posizioneBob_y = bob.y
105         #mostro sulla schermo le coordinate di Bob
106         textsurface = myfont.render(f'Bob si trova nelle
    coordinate: x: {str(posizioneBob_x)} y: {str(posizioneBob_y)}',
    False, (0, 0, 0))
107         #posiziono e mostro il testo nella mappa
108         game_display.blit(textsurface,(450,550))
```

```
109             pygame.display.flip()
110             pygame.display.update()
111             #definisco gli fps
112             clock.tick(60)
113 
114 start()
```

Listing 11: Codice videogioco "Il mondo di Bob" della seconda lezione

B Licenza del documento