|  |
| --- |
| RehaBrain |
| Documento di Analisi e Progettazione |
| Simone Cecire |
| Data: 7/4/2024 |

Sommario

[Scopo del progetto 3](#_Toc163395310)

[Specifica dei Requisiti 4](#_Toc163395311)

[Progettazione 5](#_Toc163395312)

[Elenco delle classi da realizzare 5](#_Toc163395313)

[Schema UML 5](#_Toc163395314)

[Descrizione delle classi, attributi e metodi 6](#_Toc163395315)

[Librerie da utilizzare 10](#_Toc163395316)

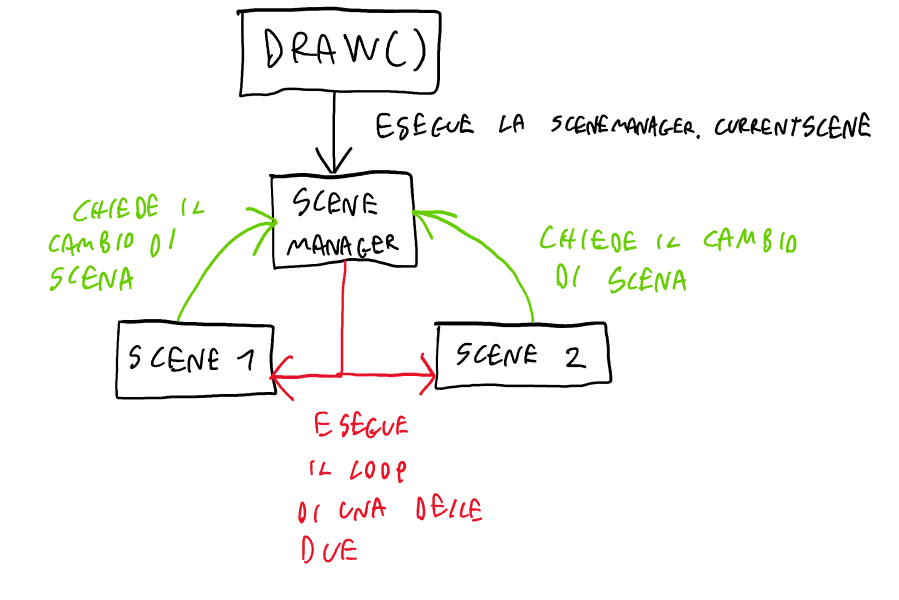
# Scopo del progetto

RehaBrain ha l’obiettivo di aiutare le persone affette da malattie neurodegenerative attraverso l’utilizzo della CST (Cognitive Stimulation Therapy), cercando di migliorare il loro attuale livello cognitivo.

# Specifica dei Requisiti

RehaBrain permette diverse attività di stimolazione cognitiva, tra cui:

1. Abbinamento di un oggetto alla parola del suo nome. Ad esempio: abbinare l’immagine di una forchetta alla scritta su schermo “forchetta”.
2. Selezionamento di determinate immagini appartenenti ad una categoria. Ad esempio: selezionare tutte le immagini (presenti in finestra) che appartengono ad una categoria di oggetti che può stare in cucina.
3. Scrivere un determinato numero di parole che facciano rima. Ad esempio: scrivere cinque parole che fanno rima con “mondo”.
4. Data la lettera di inizio e quella di fine di una parola, scriverla in modo completo (simile al gioco dell’impiccato).
5. Rimettere in ordine le immagini che rappresentano un’azione. Ad esempio: data l’azione “lavarsi i denti”, mettere in ordine quattro immagini che illustrano le azioni:
   1. Prendere lo spazzolino
   2. Mettere il dentifricio
   3. Spazzolare i denti
   4. Sciacquarsi



RehaBrain funziona con un meccanismo basato su scene (come si vede nell’immagine qui sopra) ogni attività è una scena ed avrà un metodo loop() che viene richiamato dal gestore delle scene (a sua volta richiamato nel draw() della classe principale). I cambi di attività (quindi di scena) sono gestiti dal gestore delle scene, in questo modo il cambio di attività non comporta la scrittura di codice nella classe principale.

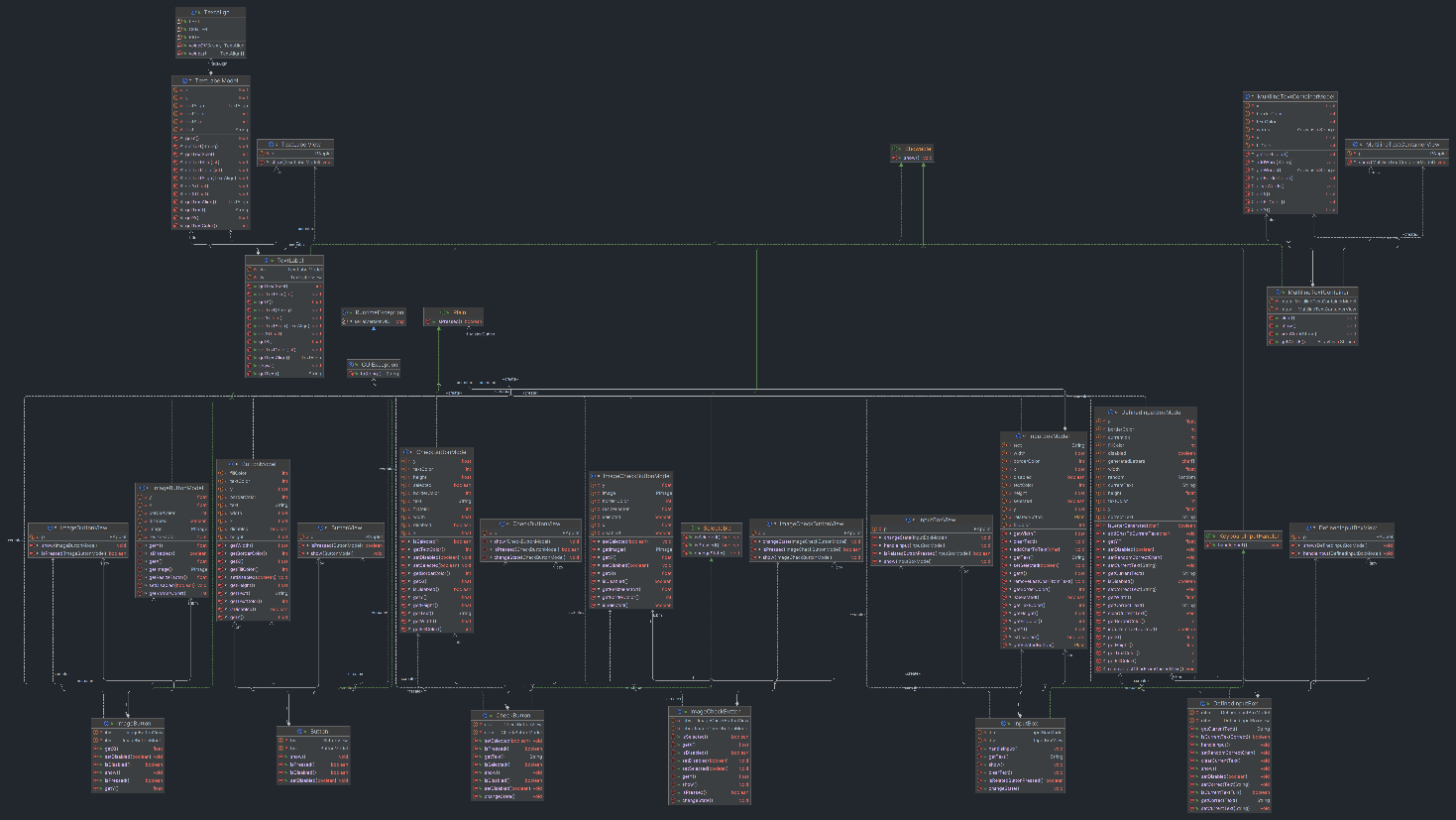
L’obiettivo è quello di creare un’applicazione che possa essere modulare ed adattabile alla persona, ad esempio con la creazione di profili diversi per avere “difficoltà” diverse nelle varie attività. Il profilo è un file di testo contenente le parole (ed il percorso ai file associati ad esse) personalizzato sulla persona. In futuro sarà possibile modificare i profili direttamente dall’applicazione senza dover modificare manualmente i file di configurazione.  
RehaBrain contiene vari file con parole ed immagini.

# Progettazione

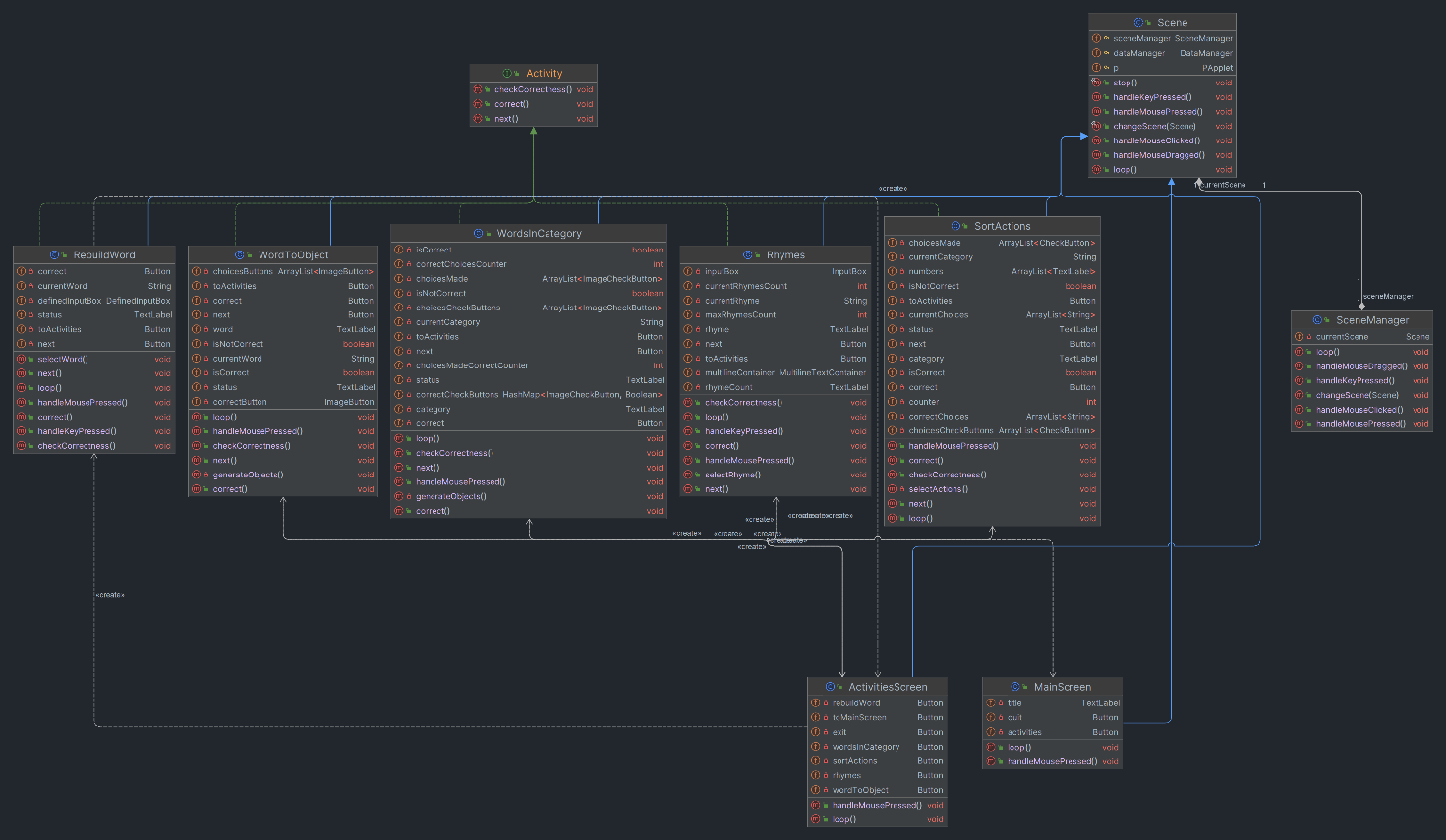
## Elenco delle classi da realizzare

### Schema UML

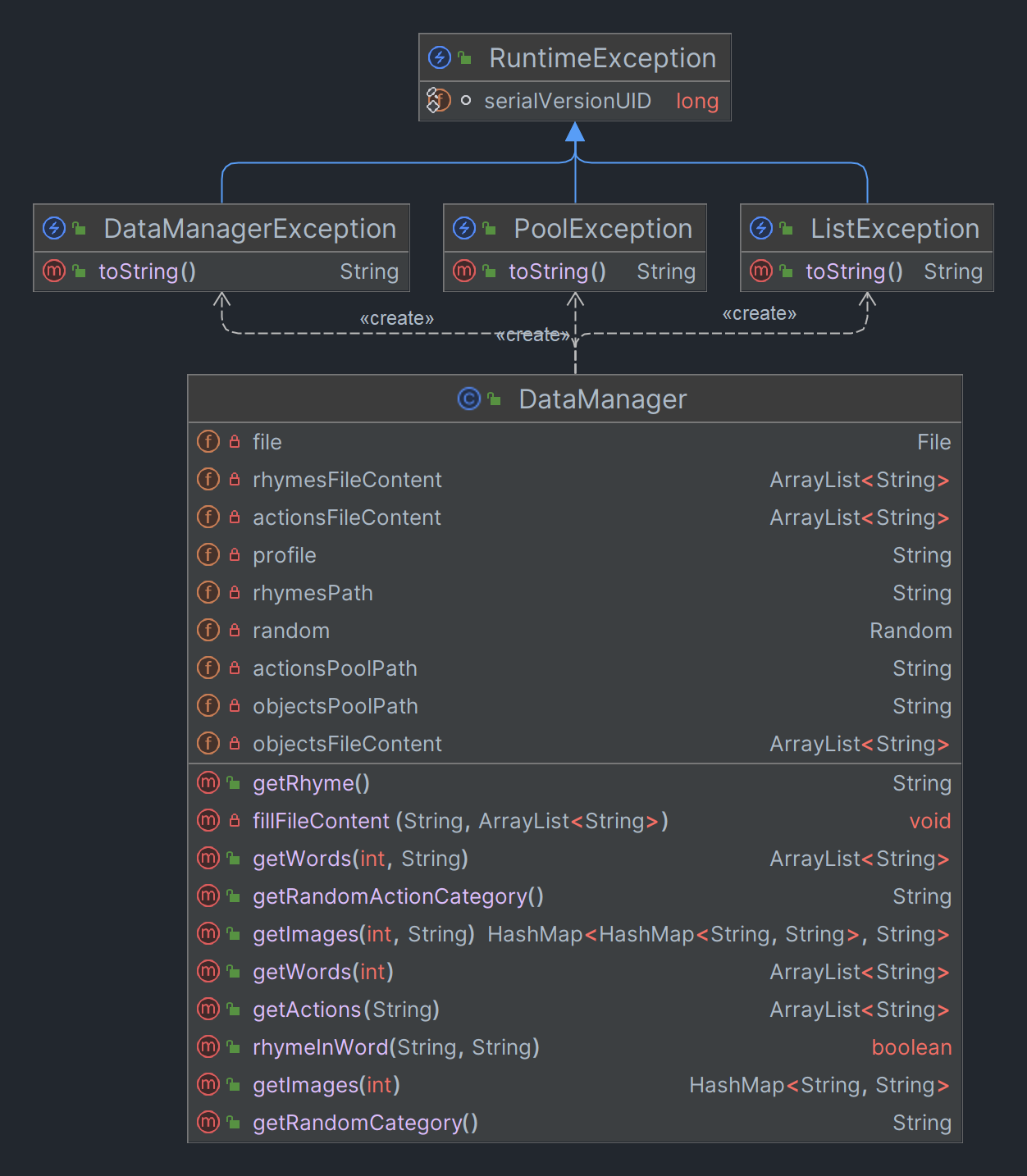
#### GUI Package



#### Scenes Package



#### DataManager Package



### Descrizione delle classi, attributi e metodi

#### GUI Package

Il package contiene tutte le classi necessarie a far funzionare l’applicazione. Tutto è sviluppato secondo il desgin pattern MVC.  
Sono presenti 4 interfacce ed 1 eccezione.

##### Showable (interfaccia)

Viene implementata in ogni classe mostrabile a schermo, contiene il metodo show().

##### Plain (Interfaccia)

Viene implementata in ogni tipologia di pulsante solo premibile, contiene il metodo isPressed().

##### Selectable (Interfaccia)

Viene implementata in ogni tipologia di pulsante selezionabile, contiene i metodi isPressed(), isSelected() e changeState().

##### KeyboardInputHandler (interfaccia)

Viene implementata in ogni tipologia di classe che permette l’input da tastiera, contiene il metodo handleInput() per gestire l’input da tastiera.

##### GUIException

Gestisce i casi di errore dell’interfaccia grafica, estende RuntimeException.

##### Button (package)

Il package per gestire un semplice pulsante, contiene anche le sue classi controller, view e model.

##### IamgeButton (package)

Il package per gestire un pulsante con un’immagine, contiene anche le sue classi controller, view e model.

##### CheckButton (package)

Il package per gestire un pulsante che viene selezionato al click, contiene anche le sue classi controller, view e model.

##### ImageCheckButton (package)

Il package per gestire un pulsante (con immagine) che viene selezionato al click, contiene anche le sue classi controller, view e model.

##### InputBox (package)

Package per la gestione degli input da tastiera. Viene rappresentata come una casella per l’inserimento di testo.

##### DefinedInputBox (package)

Package per la gestione degli input da tastiera ma con una parola definita in partenza. Viene rappresentata come una casella per l’inserimento di testo ma con la prima e l’ultima lettera visibili ed una serie di trattini (\_) per le altre lettere.

##### MultilineTextContainer (package)

Il package per gestire un insieme di testo su più righe, contiene anche le sue classi controller, view e model.

##### TextLabel (package)

Il package per gestire del testo su schermo, contiene anche le sue classi controller, view e model

#### Scenes Package

Il package contiene tutte le classi utili a far funzionare il meccanismo delle scene. È presente un’interfaccia.

##### Scene

Classe astratta che contiene un metodo astratto loop() che viene richiamato dallo SceneManager e ridefinito in ogni classe figlia di Scene. Contiene vari metodi per la gestione degli input che possono essere ridefiniti da ogni classe figlia e che vengono richiamati dallo SceneManager. Contiene un metodo final changeScene() che richiama lo SceneManager per fare un cambio di scena ed un altro metodo final stop() che chiude il programma attraverso Processing.

##### SceneManager

Gestore delle scene. Contiene il metodo loop(), che viene richiamato nel draw() e gli altri metodi per la gestione degli input che vengono richiamati nel programma principale. Serve per gestire il cambio di scene e tenere la classe principale più pulita.

##### Activity (interfaccia)

Interfaccia che rappresenta un’attività, contiene i metodi: checkCorrectness() per verificare la correttezza delle risposte, correct() per correggere le risposte e next() per avanzare alla successiva attività.

##### WordToObject

Questa classe estende Scene, implementa Activity e gestisce l’attività 1.  
Contiene il metodo loop() che gestisce la visualizzazione di tutti gli elementi grafici presenti nell’attività ed il metodo handleMousePressed() gestisce tutti gli input dei pulsanti.  
Contiene il metodo generateObjects() che:

1. Crea un ArrayList di 4 pulsanti che rappresentano tutti gli oggetti dell’esercizio che si sta svolgendo al momento
2. Prende un HashMap (dal DataManager) che contiene il nome abbinato al percorso delle immagini corrispondenti agli oggetti
3. Itera sulle chiavi del HashMap (i nomi delle immagini) e ne seleziona una casualmente che setta come parola corrente
4. Genera tutti i pulsanti e setta quello corrispondente alla parola da indovinare in una variabile

##### WordInCategory

Questa classe estende Scene, implementa Activity e gestisce l’attività 2.  
Contiene il metodo loop() che gestisce la visualizzazione di tutti gli elementi grafici presenti nell’attività ed il metodo handleMousePressed() gestisce tutti gli input dei pulsanti.  
Contiene il metodo generateObjects() che:

1. Crea un ArrayList di 5 pulsanti che rappresentano tutti gli oggetti dell’esercizio che si sta svolgendo al momento
2. Crea un HashMap (fatta di CheckButton come chiavi e Boolean come valori) che rappresenta la correttezza dei singoli pulsanti (se il pulsante è della categoria precedentemente scelta avrà come valore true)
3. Prende un HashMap (dal DataManager) che contiene immagini (fatte a loro volta da un HashMap con chiave il nome dell’immagine e valore il percorso di essa) come chiavi e la loro categoria come valori
4. Genera tutti i pulsanti e setta quelli corrispondenti alla categoria scelta nel HashMap fatta di CheckButton e Boolean

##### Rhymes

Questa classe estende Scene, implementa Activity e gestisce l’attività 3.  
Contiene il metodo loop() che gestisce la visualizzazione di tutti gli elementi grafici presenti nell’attività, il metodo handleMousePressed() gestisce tutti gli input dei pulsanti ed il metodo handleKeyPressed() che gestisce tutti gli input da tastiera.  
Contiene il metodo selectRhyme() che seleziona una nuova rima casuale dal DataManager e elimina quelle scritte precedentemente.

##### RebuildWord

Questa classe estende Scene, implementa Activity e gestisce l’attività 4.  
Contiene il metodo loop() che gestisce la visualizzazione di tutti gli elementi grafici presenti nell’attività, il metodo handleMousePressed() gestisce tutti gli input dei pulsanti ed il metodo handleKeyPressed() che gestisce tutti gli input da tastiera.  
Contiene il metodo selectWord() che seleziona una nuova parola casuale dal DataManager e la setta come parola corretta del DefinedInputBox dove l’utente andrà a scrivere.

##### SortActions

Questa classe estende Scene, implementa Activity e gestisce l’attività 5.  
Contiene il metodo loop() che gestisce la visualizzazione di tutti gli elementi grafici presenti nell’attività, il metodo handleMousePressed() gestisce tutti gli input dei pulsanti.  
Contiene il metodo selectActions() che:

1. Crea un ArrayList di 4 pulsanti che rappresenta tutte le azioni dell’esercizio corrente
2. Crea un ArrayList di CheckButton che rappresenta l’ordine di selezionamento dei pulsanti (e quindi delle azioni).
3. Prende un ArrayList di String (dal DataManager) che rappresenta tutte le azioni ordinate nel modo corretto.
4. Crea tutti i CheckButton (corrispondenti alle singole azioni ma con le posizioni mischiate).

##### ActivitiesScreen

Questa classe estende Scene e gestisce il menù delle attività. Contiene vari pulsanti per selezionare le singole attività, per tornare alla schermata principale e per chiudere il programma.

##### MainScreen

Questa classe estende Scene e gestisce il menù principale. Contiene un pulsante per andare alla schermata di selezione delle attività e per uscire dal programma.

#### DataManager Package

Il package che contiene il DataManager, ossia la classe che permette di acquisire i dati (parole ed immagini) dai vari file.

##### DataManager

La classe che permette di gestire l’utilizzo delle parole ed immagini nell’applicazione.  
Alla creazione dell’oggetto cerca nel file current.configuration il percorso della cartella da utilizzare come profilo (presente dopo la parola chiave “using”) e riempie degli ArrayList di String con il contenuto di tutte le righe dei file (memorizzati nel profilo):

* actions.pool che contiene tutti i percorsi dei file (.list) contenenti le azioni
* objects.pool che contiene il nome, il percorso e la categoria delle immagini corrispondenti agli oggetti
* rhymes.pool che contiene un elenco di parola con cui cercare le rime

Contiene il metodo fillFileContent() che viene richiamato nel costruttore e riempie un ArrayList in diversi modi rispetto alla tipologia del file (contrassegnata nella prima riga come !LIST per file solo di elenco, !WORDS-POOL per file con parole da estrarre in modo casuale e !POOL per file con immagini da estrarre in modo casuale).  
Contiene i metodi getWords(int n) e getImages(int n) che restituiscono n parole o immagini ed i loro rispettivi overload con l’aggiunta di una categoria specifica come parametro, il metodo getActions(String category) che permette di restituire un ArrayList contenente le azioni di uno specifico file, i metodi getRandomCategory() e getRandomActionCategory() che restituiscono una categoria casuale per immagini e azioni, il metodo getRhyme() che restituisce una rima casuale dal file contenente le rime e il metodo rhymeInWord(String rhyme, String word) che controlla che una parola faccia rima con un’altra.

##### DataManagerException

Eccezione per errori del DataManager. Estende RuntimeException.

##### ListException

Eccezione per errori sui file di tipo lista (contenenti solo elenchi di parole). Estende RuntimeException.

##### PoolException

Eccezione per errori sui file di tipo pool (contenenti elenchi di parole abbinate a valori, come immagini). Estende RuntimeException.

## Librerie da utilizzare

Per realizzare il progetto è stato utilizzato Processing ed il Collections framework, di questo nello specifico vengono utilizzati ArrayList e HashMap, le seconde per la gestione della corrispondenza testo-immagine.