

Program Tips

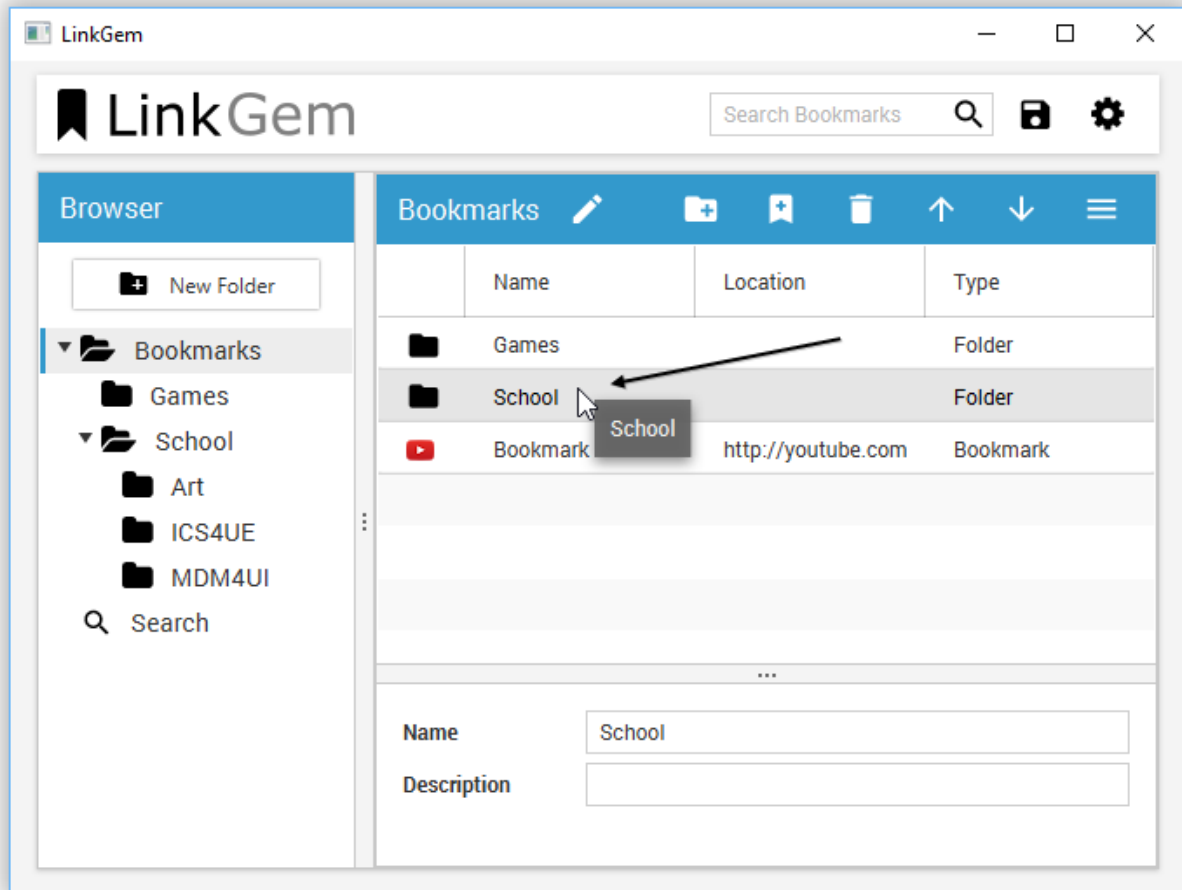
Tooltips are key

Tooltips are very useful. They provide information on what a control does.

Open Folder in Explorer

While in the Explorer view, you can double click on the bookmark item if it is a folder to reveal its contents.

A bookmark item is a folder when it has a folder icon or the type says “Folder”.



Program Requirements Outline

In this document, all 16 program requirements for the summative are outlined.

To find evidence that a program requirement has been satisfied, the following will be provided:

1. Package
2. Class
3. Method

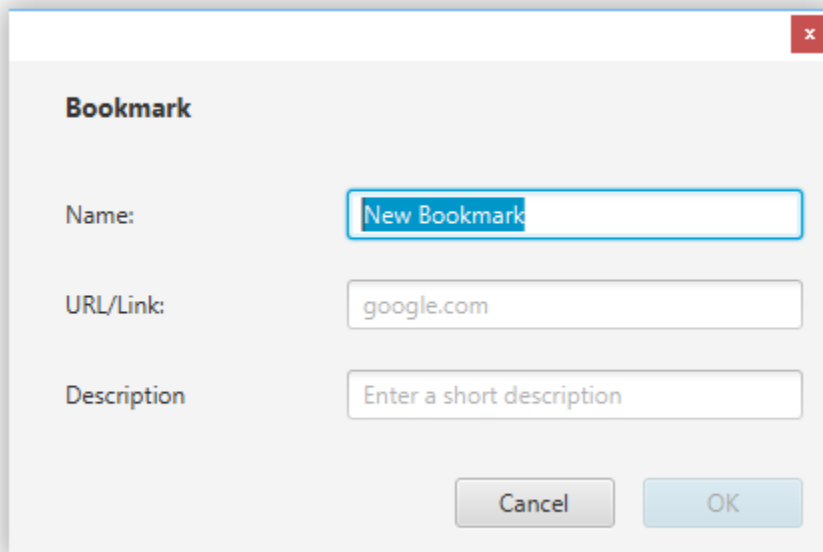
Note that sometimes methods have an ellipse (...) which means that some code has been cut off for the sake of readability for this document.

Produce a modular program divided among multiple files.

Multiple files are evident. (Over 50 classes)

Design user-friendly input-output form/forms.

Example: Bookmark dialog - It has validation support. It notifies if users enter something wrong!

A screenshot of a Java Swing dialog box titled "Bookmark". The dialog has a standard title bar with a red close button. Inside, there are three text input fields. The first field is labeled "Name:" and contains the text "New Bookmark". The second field is labeled "URL/Link:" and contains the text "google.com". The third field is labeled "Description" and has a placeholder text "Enter a short description". At the bottom right of the dialog are two buttons: "Cancel" and "OK".

```
package com.github.shaigem.linkgem.ui.dialog.bookmark;
```

Class:

```
public class BookmarkDialogPresenter implements Initializable, DialogBasedItemEditor
```

Ronnie Tran

Use data types (Int, Boolean, String).

For Int:

Used in loops.

```
package com.github.shaigem.linkgem.ui.main.explorer;
```

Class:

```
public class FolderExplorerPresenter implements Initializable
```

Method:

```
private void createViewSettingsMenu() {  
...  
    // Loop through all of the columns except for the first one which is the icon  
    column and create a  
    // menu item which allows users to toggle which column to display  
    for (int i = 1; i < itemTableView.getColumns().size(); i++) {  
        final TableColumn<Item, ?> itemTableColumn =  
itemTableView.getColumns().get(i);  
        final String columnName = itemTableColumn.getText();  
...  
    }  
}
```

For Boolean:

```
package com.github.shaigem.linkgem.model.item;
```

Class:

```
public class FolderItem extends Item
```

Constructor:

```
public FolderItem(String name, String description, boolean readOnly)
```

For String:

```
package com.github.shaigem.linkgem.model.item;
```

Class:

```
public class FolderItem extends Item
```

Constructor:

```
public FolderItem(String name, String description, boolean readOnly)
```

Use conditionals (if-statement, switch statement).

For if-statement:

```
package com.github.shaigem.linkgem.ui.main;
```

Class:

```
public class MainWindowPresenter implements Initializable
```

Method:

```
@EventListener
private void onSaveAll(SaveAllEvent event) {
    boolean success =
BookmarkSerialization.getInstance().serialize(folderRepository.getMasterFolder());
    if (success) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Save Success");
        alert.setHeaderText("Save Success!");
        alert.setContentText("All of your bookmarks have been successfully saved!");
        alert.show();
    } else {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Save Failure");
        alert.setHeaderText("Save Failure!");
        alert.setContentText("No bookmarks were saved.");
        alert.show();
    }
}
```

For switch statement:

```
package com.github.shaigem.linkgem.ui.main.explorer;
```

Class:

```
public class FolderExplorerPresenter implements Initializable
```

Method:

```
public void performAction(ExplorerAction action) {

    switch (action) {
        case ADD_BOOKMARK:
            onAddBookmarkAction();
            break;
        case ADD_FOLDER:
            onAddFolderAction();
            break;
        ...
    }
}
```

Ronnie Tran

Use loops (for, while).

For-Loop:

```
package com.github.shaigem.linkgem.ui.main.explorer;
```

Class:

```
public class FolderExplorerPresenter implements Initializable
```

Method:

```
private void createViewSettingsMenu() {  
...  
    // Loop through all of the columns except for the first one which is the icon  
    column and create a  
    // menu item which allows users to toggle which column to display  
    for (int i = 1; i < itemTableView.getColumns().size(); i++) {  
        final TableColumn<Item, ?> itemTableColumn =  
itemTableView.getColumns().get(i);  
        final String columnName = itemTableColumn.getText();  
...  
    }  
}
```

While-Loop:

The merge sorting implementation uses while loops.

```
package com.github.shaigem.linkgem.sort.impl;
```

Class:

```
public class BookmarkMergeSortingRoutine extends SortingRoutine
```

Method:

```
private Item[] merge(Item[] sortedLeftHalfArray, Item[] sortedRightHalfArray, boolean  
descending)
```

Use built-in methods and properties.

Used almost everywhere. Accessing JavaFX methods, using JSON, etc.

See IconManager for an example class that uses built-in methods.

```
package com.github.shaigem.linkgem.favicon;
```

Class:

```
public final class IconManager
```

Create and use custom functions.

Can be found almost everywhere.

One example:

```
package com.github.shaigem.linkgem.repository;
```

Class:

```
public class FolderRepository
```

Method:

```
private ObservableList<Item> collectItems(FolderItem folderItem) {  
    ObservableList<Item> items = FXCollections.observableArrayList();  
    for (Item item : folderItem.getChildren()) {  
        if (item instanceof FolderItem) {  
            items.add(item);  
            items.addAll(collectItems((FolderItem) item));  
        } else {  
            items.add(item);  
        }  
    }  
    return items;  
}
```

Use at least one form of array (one-dimensional, two-dimensional, data objects).

Class:

```
public class FolderRepository
```

Method:

```
private ObservableList<Item> collectItems(FolderItem folderItem) {  
    ObservableList<Item> items = FXCollections.observableArrayList();  
    for (Item item : folderItem.getChildren()) {  
        if (item instanceof FolderItem) {  
            items.add(item);  
            items.addAll(collectItems((FolderItem) item));  
        } else {  
            items.add(item);  
        }  
    }  
    return items;  
}
```

Note: An observable list uses an array list as a backing. It holds an array of Item objects.

One-Dimensional:

Use:

```
package com.github.shaigem.linkgem.ui.main.explorer;
```

Class:

```
public class FolderExplorerPresenter implements Initializable
```

```
private void performManualSorting(SortOrder order) {  
    if (getViewingFolder().getChildren().isEmpty()) {  
        return;  
    }  
    Item[] itemsToSort = new Item[getViewingFolder().getChildren().size()];  
    itemsToSort = getViewingFolder().getChildren().toArray(itemsToSort);  
    final BookmarkMergeSortingRoutine bookmarkMergeSortingRoutine = new  
BookmarkMergeSortingRoutine();  
    getViewingFolder().getChildren().setAll(bookmarkMergeSortingRoutine.sort(order,  
itemsToSort));  
}
```

Ronnie Tran

Perform searching as required.

```
package com.github.shaigem.linkgem.ui.main.explorer;
```

Class:

```
public class FolderExplorerPresenter implements Initializable
```

Method:

```
private void createViewSettingsMenu() {  
...  
    // loop through all of the columns except for the first one which is the icon  
    column and create a  
    menu item which allows users to toggle which column to display  
    for (int i = 1; i < itemTableView.getColumns().size(); i++) {  
        final TableColumn<Item, ?> itemTableColumn =  
itemTableView.getColumns().get(i);  
        final String columnName = itemTableColumn.getText();  
...  
        // we must always show the name column so don't allow users to hide it!  
        if (columnName.equals("Name")) { //SEARCHING FOR COLUMN WITH A NAME OF "Name"  
            checkMenuItem.setDisable(true);  
        }  
    }  
}
```


Ronnie Tran

Perform sorting as required.

Use:

```
package com.github.shaigem.linkgem.ui.main.explorer;
```

Class:

```
public class FolderExplorerPresenter implements Initializable
```

Method:

```
private void performManualSorting(SortOrder order) {  
    if (getViewingFolder().getChildren().isEmpty()) {  
        return;  
    }  
    Item[] itemsToSort = new Item[getViewingFolder().getChildren().size()];  
    itemsToSort = getViewingFolder().getChildren().toArray(itemsToSort);  
    final BookmarkMergeSortingRoutine bookmarkMergeSortingRoutine = new  
BookmarkMergeSortingRoutine();  
    getViewingFolder().getChildren().setAll(bookmarkMergeSortingRoutine.sort(order,  
itemsToSort));  
}
```

Implementation:

```
package com.github.shaigem.linkgem.sort.impl;
```

Class:

```
public class BookmarkMergeSortingRoutine extends SortingRoutine
```

Method:

```
private Item[] merge(Item[] sortedLeftHalfArray, Item[] sortedRightHalfArray, boolean  
descending)
```

Ronnie Tran

Use recursion (if necessary).

In the example provided below, the *collectItems* method collects all items in the given folder. If the given folder contains more folders, recursion will be used to collect items in all sub-folders as well.

```
package com.github.shaigem.linkgem.repository;
```

Class:

```
public class FolderRepository
```

Method:

```
private ObservableList<Item> collectItems(FolderItem folderItem) {  
    ObservableList<Item> items = FXCollections.observableArrayList();  
    for (Item item : folderItem.getChildren()) {  
        if (item instanceof FolderItem) {  
            items.add(item);  
            items.addAll(collectItems((FolderItem) item));  
        } else {  
            items.add(item);  
        }  
    }  
    return items;  
}
```

Read from and write to an external file (e.g., input and store high scores or other needed data).

JSON was used to store the user's bookmarks and folder for later use.

See: data/items.json

```
package com.github.shaigem.linkgem.serialization;
```

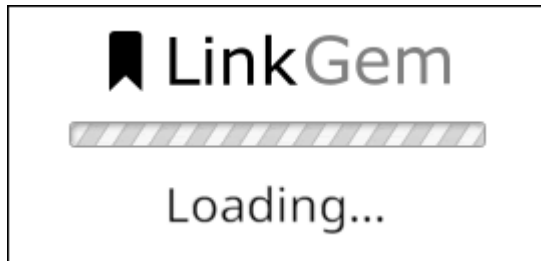
Class:

```
public final class BookmarkSerialization
```

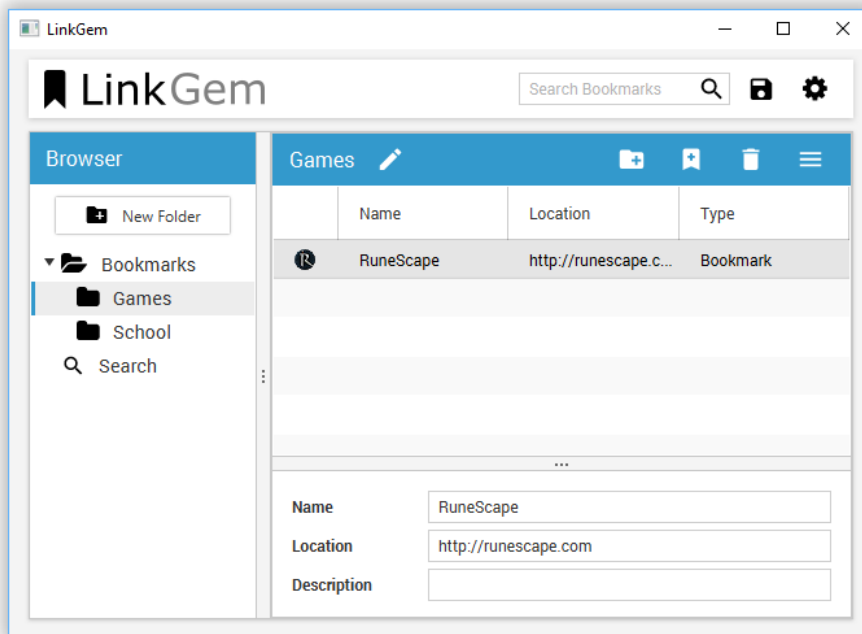
Ronnie Tran

Encapsulate the final program to include intro, game, and game overscreens.

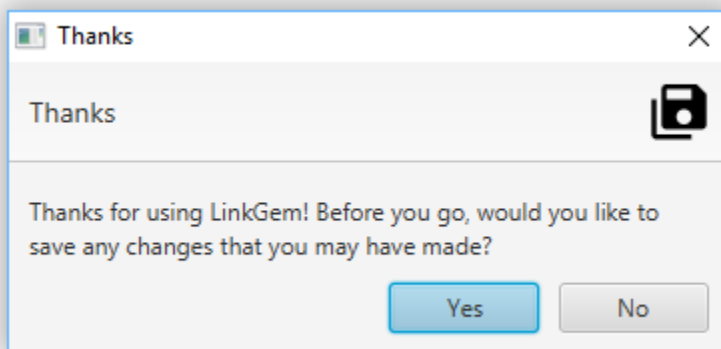
Intro Splash:



Main Program:



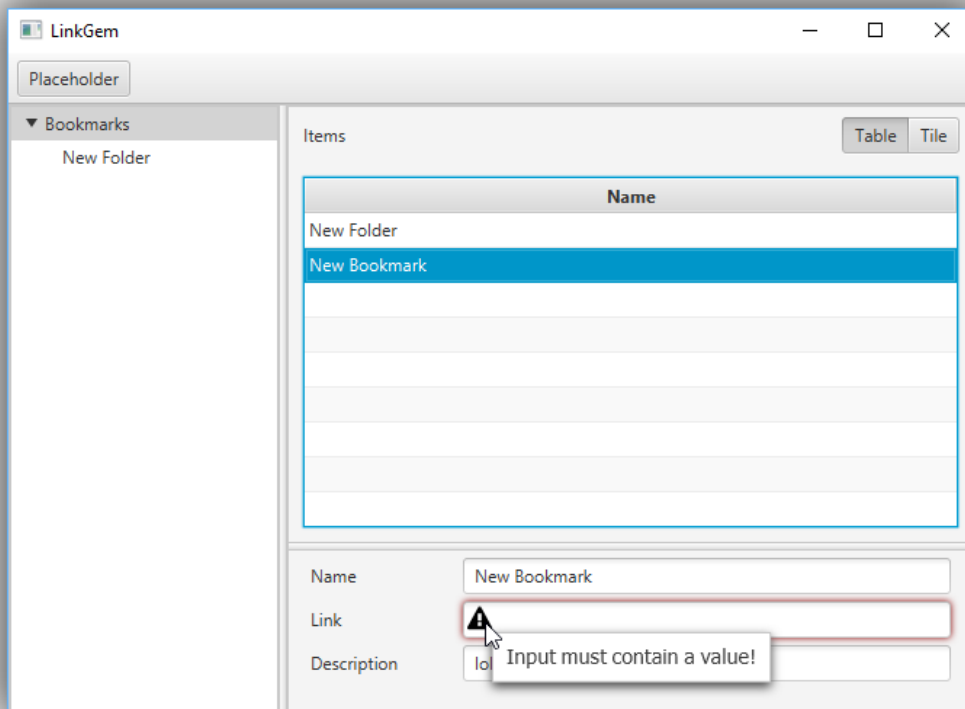
Outro is just a dialog saying thanks and it requests the user to save just in case



Add an enhancement (scorekeeping, timer, graphics, and animation).

1. Instead of creating only one folder for bookmarks, users can create as many folders as they can to organise bookmarks.
2. Search for items as you type
3. CSS for user interface styling

Before CSS was used:



Almost all JavaFX controls were styled manually using CSS.

CSS files can be found in the resources folder.

Use white space and indenting to improve readability of code.

Can be seen in any class file.

Ronnie Tran

Use internal comments to explain clearly the program code.

Found in almost all classes and methods. Internal comments are only made to methods which may be hard to understand without comments.