Grab&Run;

Software Design Document

By:

Marwa Hany
Menna Allah Medhat
Shereen Mohammed Zeinah
Toka Mohamed Naguib

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

- The purpose of this document is to outline the technical design of **Grab and run;** and provide an overview for implementation through describing software architecture, software subsystems and their interfaces.
- Describe the functionality which will be provided by each component or group of components and show how the various components interact in the design.
- This document will be used by software developers and testers to implement the software, so it must provide them with all the needed information and details for this task.

1.2 Scope

This documents focuses on the requirements mentioned in the SRS to enable developers and testers with enough details to implement the system.

Our system facilitates the process of education through an offline software loaded on our PC and smart phones without the need to have an internet connection. Users will easily download the desktop application and right away use it without the need to any license. It will have a very simple user interface to enable users to use it without the need to any guide. It focuses on logical thinking and programming, also users don`t necessary have a background of programming and algorithms. They have a collection of blocks which they can easily drag and drop and equivalent code will appear immediately. These blocks cover some outputs, inputs and control of flow of the code so user can easily customize his project according to his needs. Also user can easily save his/her code for later use.

1.3 Overview

This document briefly describes the system architecture of the web application, design diagrams as class diagram, data flow diagram, sequence diagram and user scenarios. The design patterns which will be used to implement the system, some simple user interface to

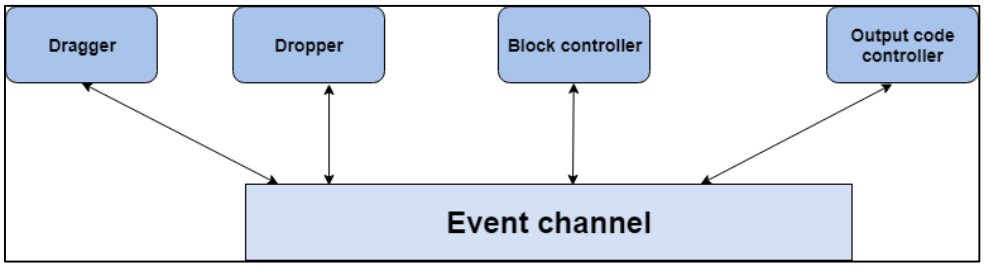
explain the expected output of the project. Also you will find a detailed explanation of each component in our system. How data is passed through our system and differ from each component to the other.

2. SYSTEM OVERVIEW

This system is intended to non-majored users who doesn't have a background in programming. You will notice a simple design and interface as it`s a measure functionality in this system to be easy to use and navigate through it without any complexity. System is sub divided into a set of components each component has a specific task to preform to increase coherency and decrease dependency between components. So you will find a component considered only with dragging object another component for dropping objects, controlling language for code to appear, saving our codes in external files and so on. This system is installed on the user`s local PC so it`s not necessary to have a network access to use the system or view his previous projects which is considered as a privilege.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design



System architecture

3.2 Decomposition Description

Subsystems:

- **Block controller:**

This component deals with all information considered with draggable blocks. Where it's called for each block to return block code and variable declaration depending on the type of the block. It sets if this block is copied or instantiated to decide whether to use variable declaration or not.

- **Dragger component:**

This component is responsible for dealing with any draggable game object in our game. So in order to make a game object draggable it must get respond from this component. While dragging our game object it keeps track of the position of this game object. At the end of dragging it checks if we are ending our drag in the right position, if it's the right position it customizes the block shape to the previously adjusted size and image.

- **Output code controller component:**

This component is called and executed each time a block is dragged and dropped in the drop panel. It`s responsible for getting and printing each block`s code and variable declaration depending on the type of each block so the output code of our algorithm appears to us on a separate panel in a synchronized way each time we a drop a block.

- **Dropper component:**

This component is called by the main drop panel when a block is about to be dropped to it , so it set it`s parent and position is set to the right position. Also it`s used to customize the preferred height of the drop panel to set it according to the number of blocks dropped on it.

3.3 Design Rationale

Using an event-driven architecture are as EDA is particularly well suited to the loosely coupled structure of complex engineered systems. We do not need to define a well bounded formal system of which components are either a part of or not. Instead, components can remain autonomous, being capable of coupling and decoupling into different systems in response to different events. Thus, components can be used and reused by many different systems. Secondly, versatility. Event-driven architecture allows systems to be constructed in a manner that facilitates greater responsiveness because event-driven systems are, by design, normalized to unpredictable, nonlinear, and asynchronous environments. They can be highly versatile and adaptable to different circumstance

4. DATA DESIGN

4.1 Data Description

Main function of this system is to enhance the user with the translated code corresponding to this own arrangement of blocks.

So our data is mainly considered with extracting the generated code and saving it externally on our disk for later use.

Code is saved in a .txt file with a name specified by the user.

5. COMPONENT DESIGN

5.1 Component description:

- **Dragger component:**

This component is responsible for dealing with any draggable game object in our game. So in order to make a game object draggable it must get respond from this component. This component enables us with some important functionalities, so on the beginning of the drag it does two functions the first one it checks the type of our block if it`s an expandable block as for loop block or an nonexpendable block as motor block, depending on the type of the block it changes the image of the block on dropping., the second function it calls **Block duplicator** component which will be mention in details later and send to it our game object and its order relative to other blocks. While dragging our game object it keeps track of the position of this game object. At the end of dragging it checks if we are ending our drag in the right position, if it's the right position it customizes the block shape to the previously adjusted size and image.

- **Output code controller component:**

This component is called and executed each time a block is dragged and dropped in the drop panel. It`s responsible for getting and printing each block`s code and variable declaration depending on the type of each block so the output code of our algorithm appears to us on a separate panel in a synchronized way each time we a drop a block.

- **Block duplicator component:**

As previously mentioned this component is called automatically from dragger component each time a game object is dragged and released in the drop panel. It instantiates a game object in our resources blocks in the same place and order of the dragged and released object so that the user can drag and use many game objects of the same type.

- **Code saver component:**

This component is called when save code button is clicked. It is responsible of extracting and saving our output code in an external file for later use.

- **Language controller component:**

A component called in the beginning of the game. It is used to determine the language of the output code to be translated from our blocks. A user can choose java language or python language, on clicking either of these languages this component is called to get the name of the language and path it to the blocks so they can translate their code in the right language chosen by the user. This component calls Output code controller component after getting the language name to start translating blocks in a synchronized way.

- **Block maximizar component:**

This component is called by the inside body of for block and while block and any block that can accept a body. It's responsible to customize the inside panel with the preferred size for the its height depending on the number of children that exists to fit in this block.

- **Block controller:**

This component is carried by each block of code in our game, and it`s called by the output code controller component so that we can print our code. Simple block controller can respond to several functions as, getting variable declaration of the block, determining whether it`s a copied block of an already dropped block or not, getting object code.

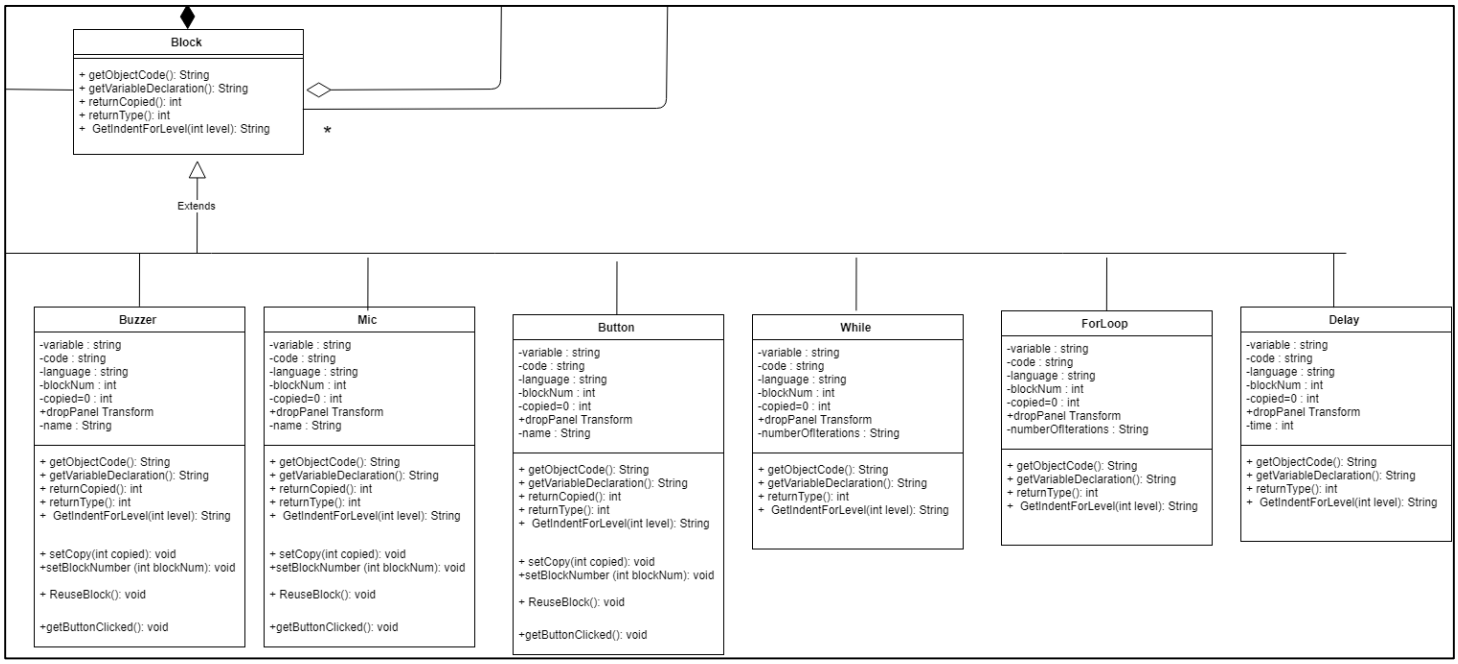
- **Dropper component:**

This component is called by the main drop panel when a block is about to be dropped to it , so it set it`s parent and position to the right position. Also it`s used to customize the preferred height of the drop panel to set it according to the number of blocks dropped on it.

5.2 Design patterns:

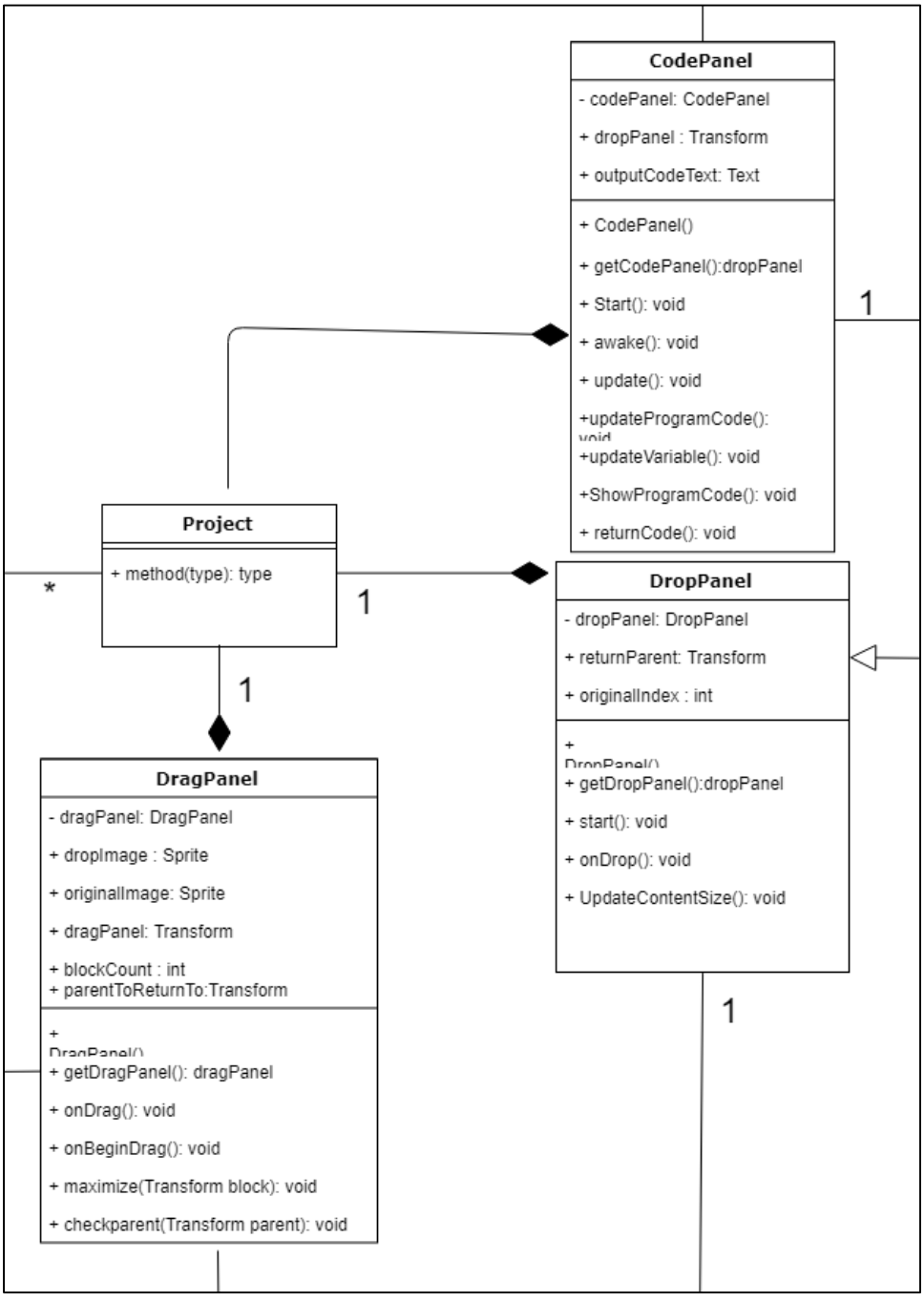
- **Abstract factory:**

Block class is an abstract factory for all blocks, where it`s an abstract class containing a set of methods. Class who inherits block class implements these methods. Each child class implement its method in their own way according to its function and need.

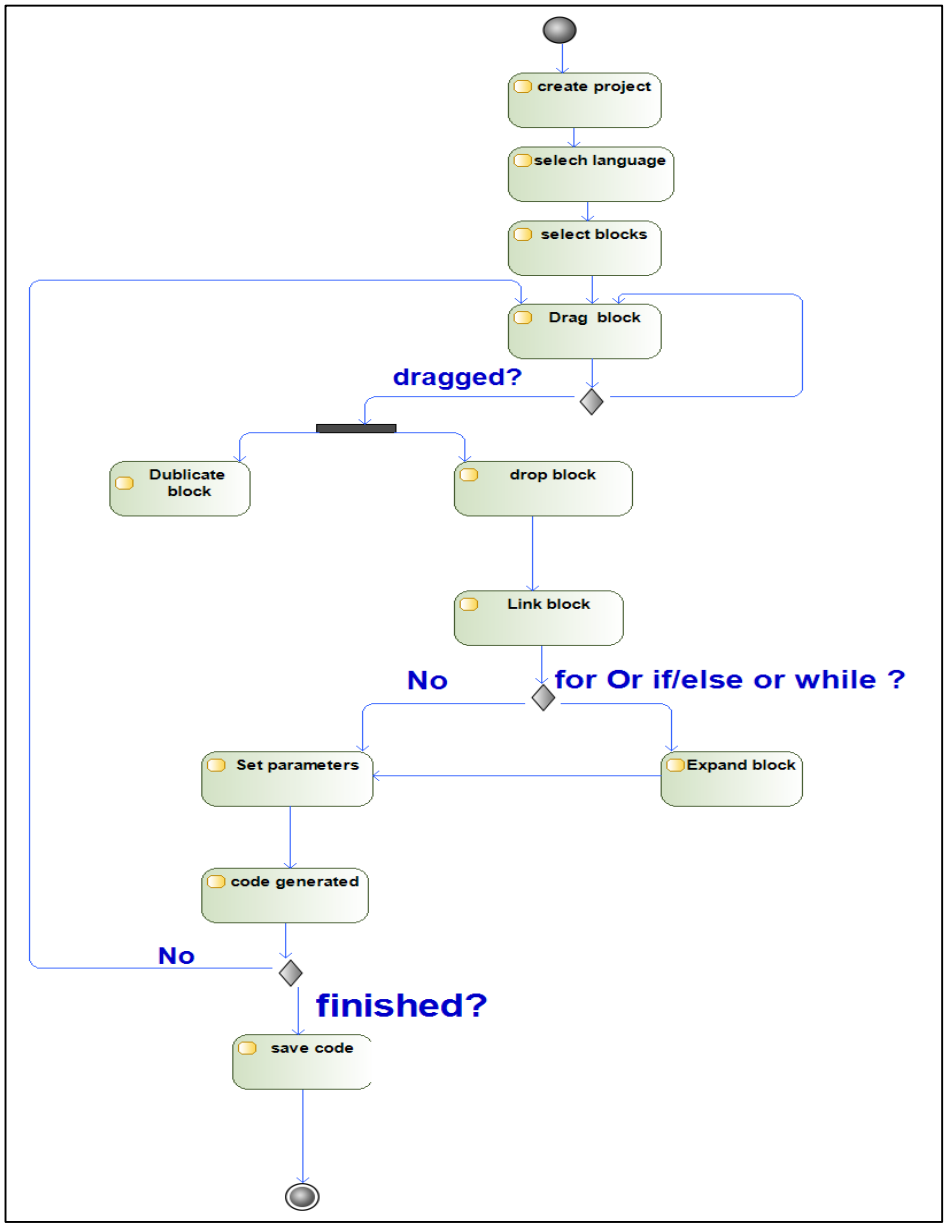


- **Singleton:**

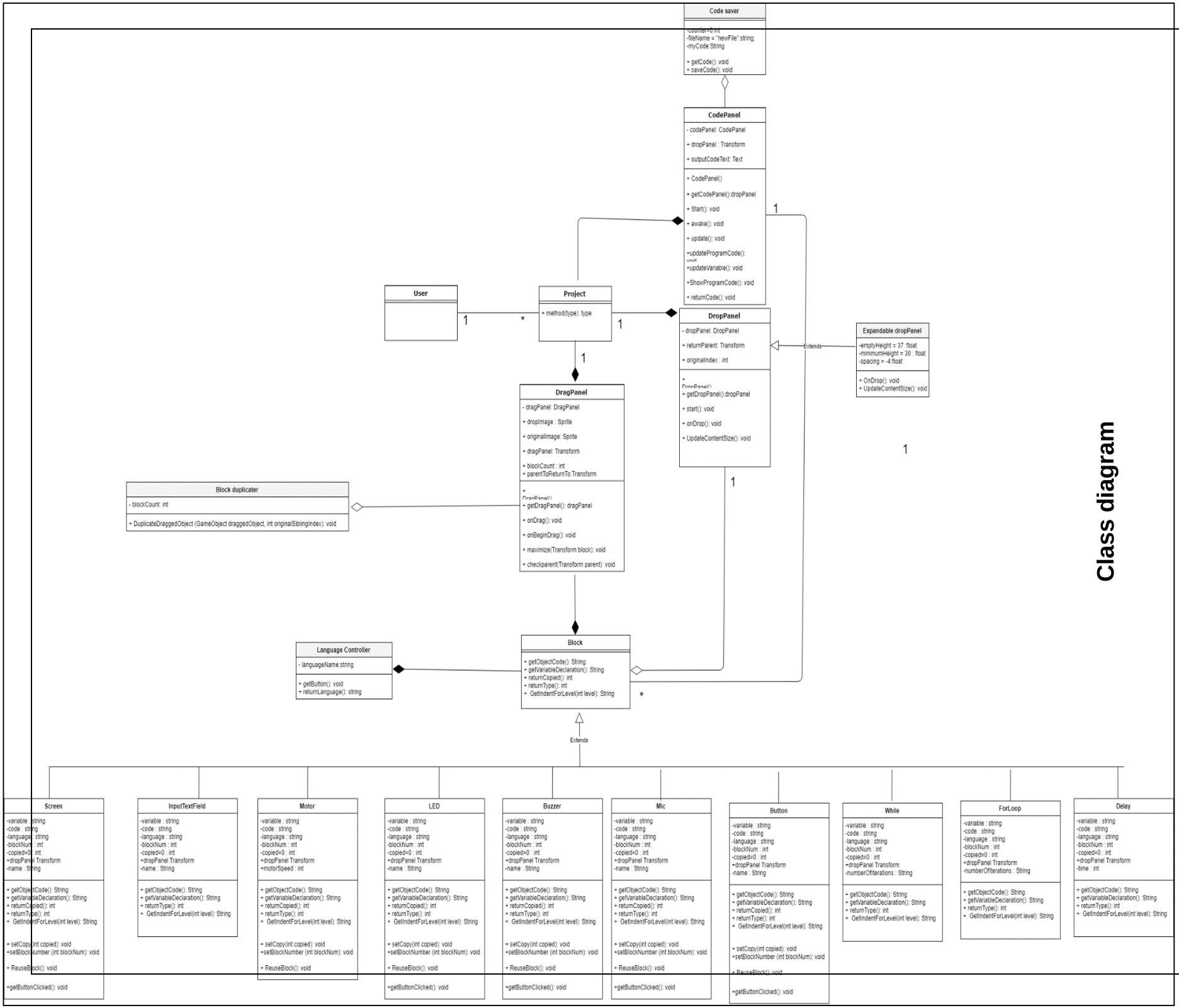
This design pattern is used with code panel, drop panel and drag panel. It's very useful because it ensures making only one instance of these panels through our game.



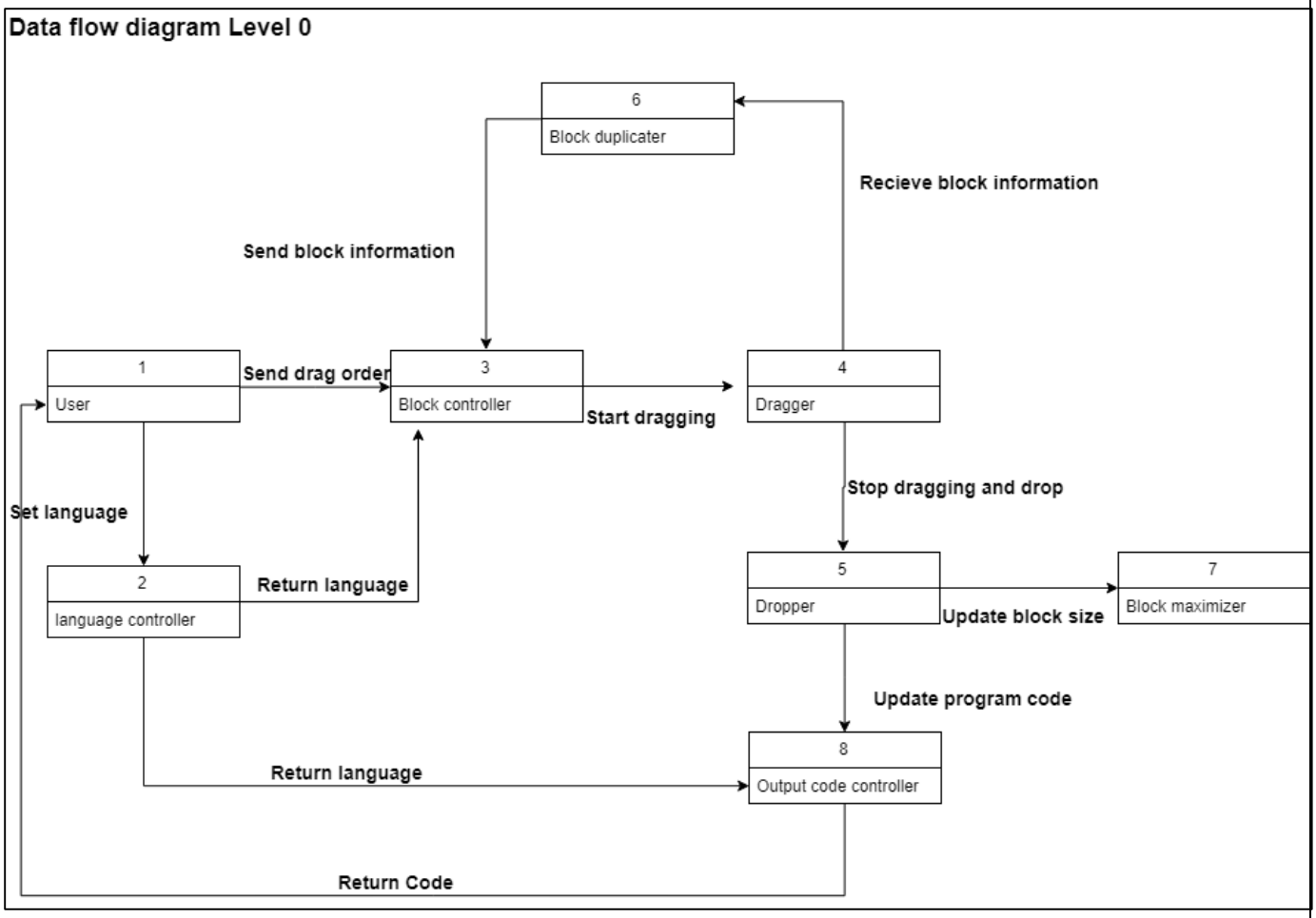
5.3 Design diagrams:

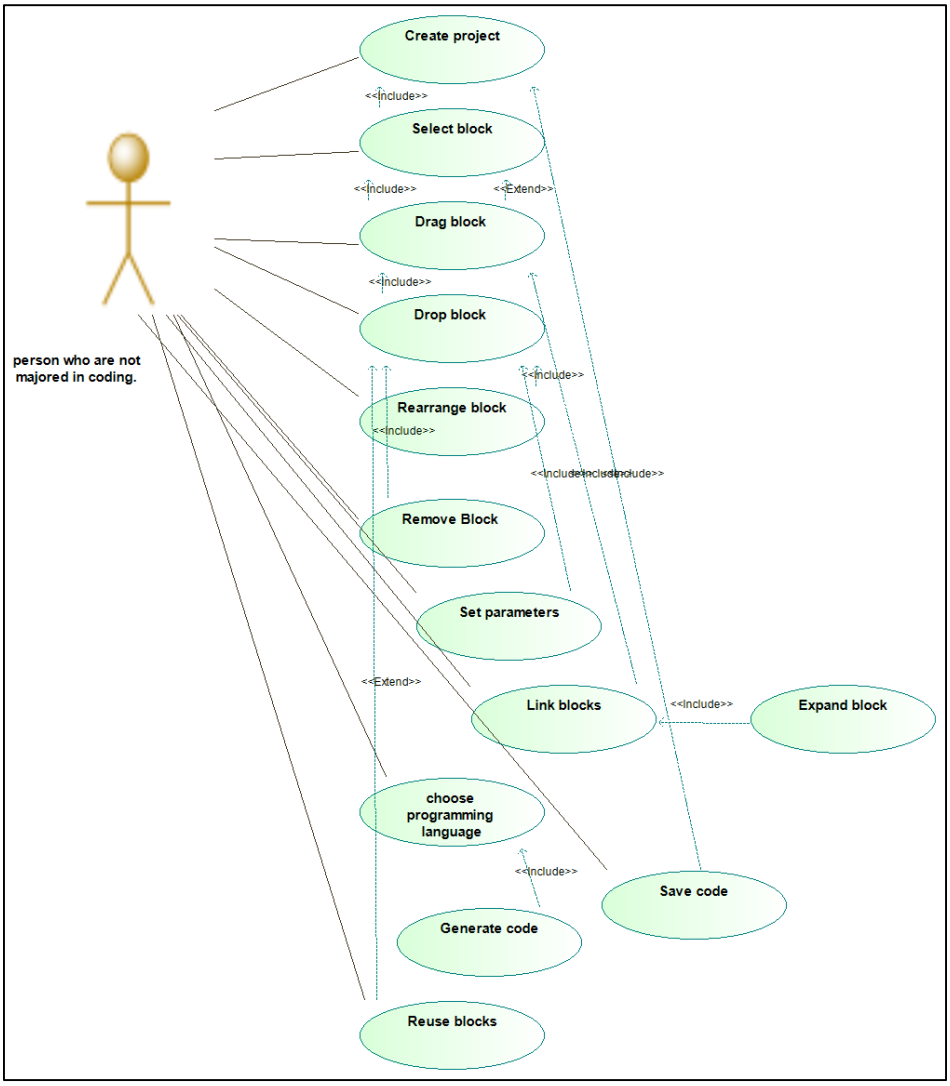


Activity diagram



Class diagram


Data flow diagram Level 0
6
Block duplicater
Recieve block information
Send block information
1
User
Send drag order
3
Block controller
Start dragging
4
Dragger
Stop dragging and drop
Set language
2
language controller
Return language
5
Dropper
Update block size
7
Block maximizer
Update program code
8
Output code controller
Return language
Return Code



Use case diagram

5.4 User scenarios and sequence diagram:

1st scenario:

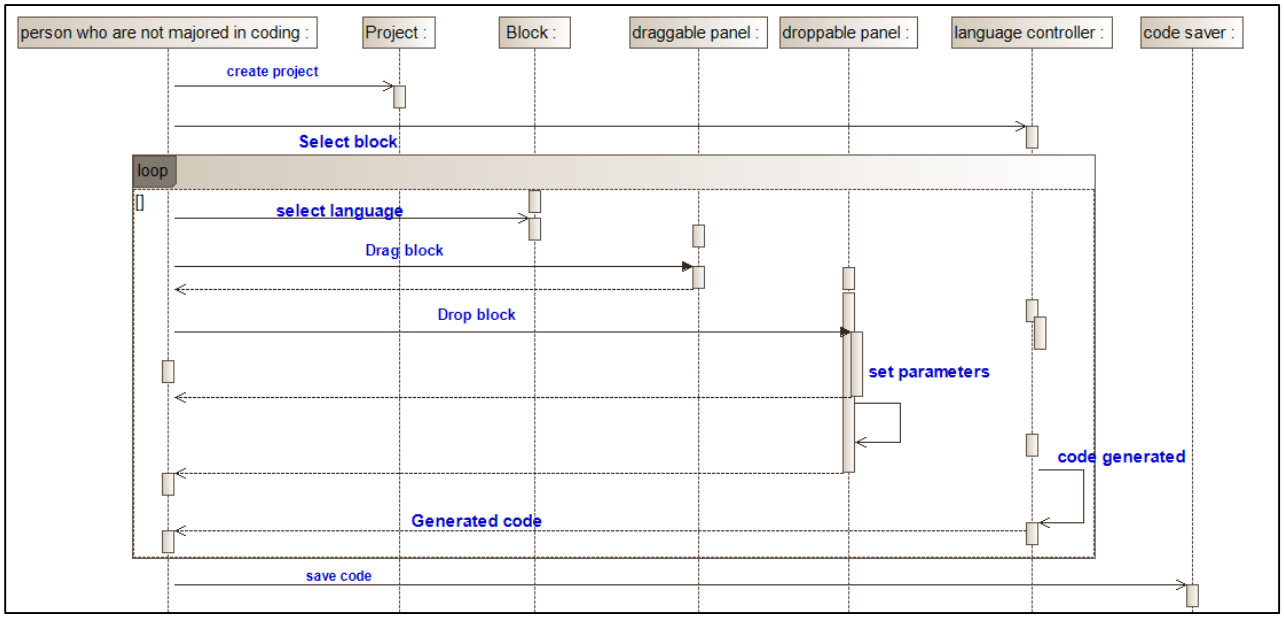
Participating Actors: User

Entry Condition: User Visit the application and use it to create a new project trying to learn new language.

Exit Condition: the user select close window

Flow of Events: the user create project by selecting the language he need to learn, then select the blocks he needs to drag from the

draggable panel then drop them in the droppable panel then set the required parameters then code generates automatically, finally if he had finished his code he can save it by pressing on save button.



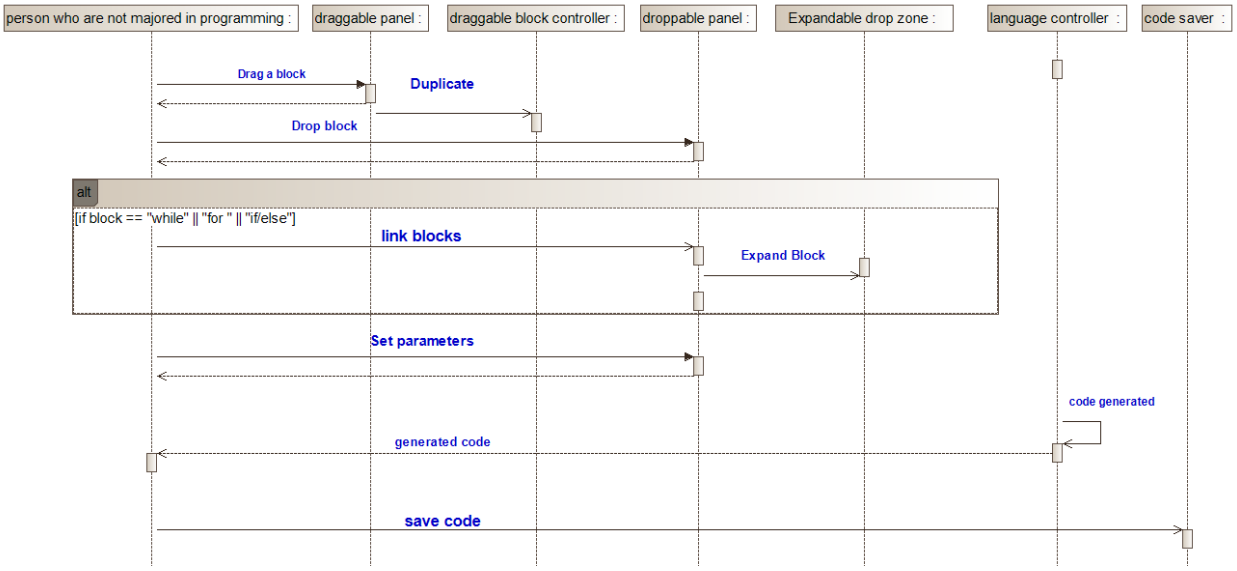
2nd scenario:

Participating Actors: User

Entry Condition: User drag a block from dragabble panel.

Exit Condition: the user select close window.

Flow of Events: the user drags a block from draggable panel when it is dragged, the dragabble controller will duplicate the dragged block in the same panel then the user drops it in droppable panel, if the block (while or for or if/else) then it will be expanded by expandable drop zone when user linking other blocks, finally if he had finished his code he can save it by pressing on save button.


person who are not majored in programming :
draggable panel :
draggable block controller :
droppable panel :
Expandable drop zone :
language controller :
code saver :
Drag a block
Duplicate
Drop block
alt
[if block == "while" || "for " || "if/else"]
link blocks
Expand Block
Set parameters
code generated
generated code
save code

6. HUMAN INTERFACE DESIGN

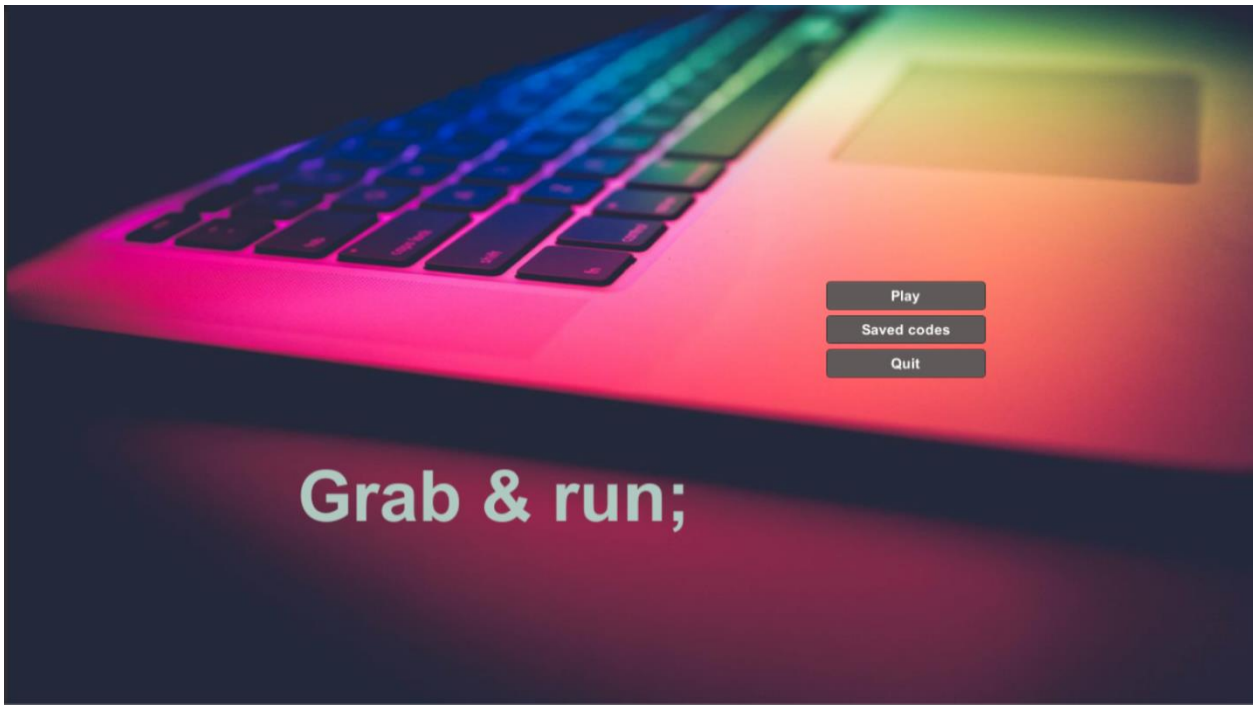
6.1 Overview of User Interface

To start the game, play button in the first scene will be pressed this will move us to the second scene in which user create his own project by enter a name of project then press on create project button then choose the programming language he want his program will be written with, after user selected his preferred language the main scene will appear which is consists of three main parts drag panel (left zone in scene) , drop panel (middle zone in the scene) , code panel (right zone in the scene) , the first part "drag panel" contain ten blocks motor block ,led block ,buzzer block ,mice block , for block ,while block ,input field block ,print block ,button block ,and delay block, user can drag these blocks from the drag panel a then drop them in the drop panel which is the second part in this scene, different blocks have a different parameters that the user must enter them according to the program he wants to write, also he can reuse some of the blocks (led block , buzzer block , mice block , button block) from the plus button appear on block in order to make a logical sequence of program code, the code of each block will appear in the last part of the scene "code panel" at the same time the block was dropped in the drop panel.

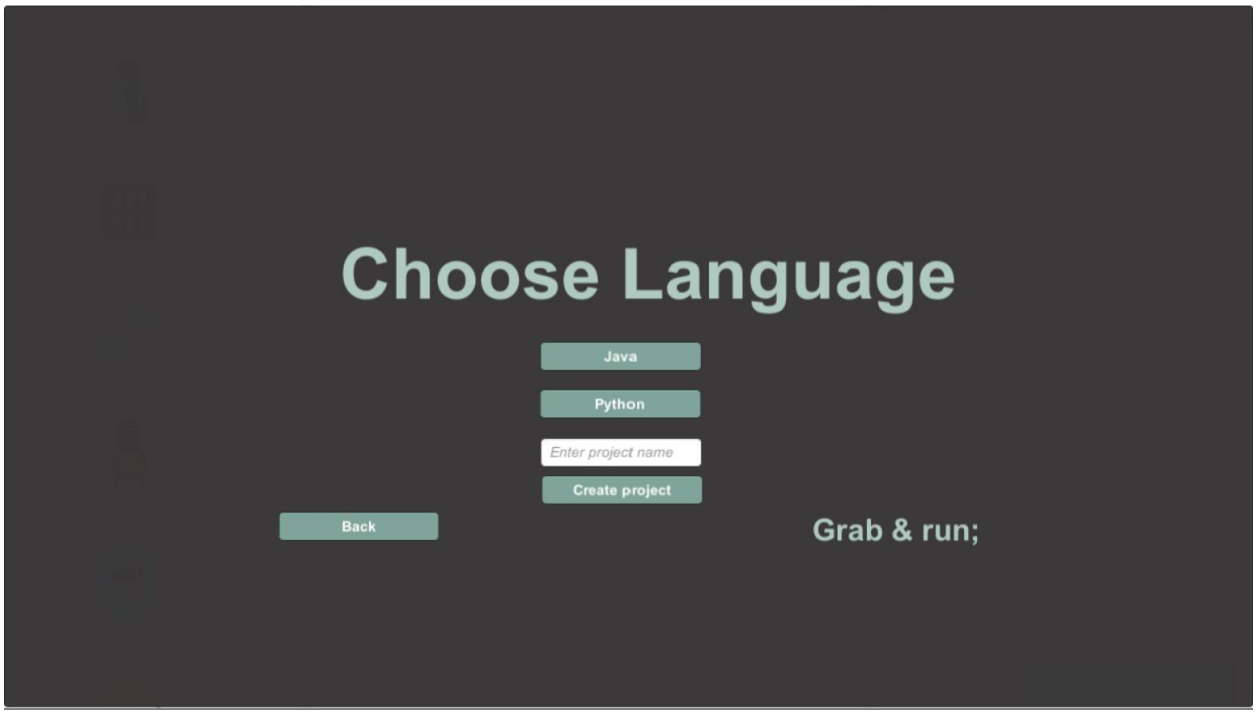
After user finishing his program he can save his code from the save button in the bottom of code panel so he can run his code on different platforms.

.

6.2 Screen Images

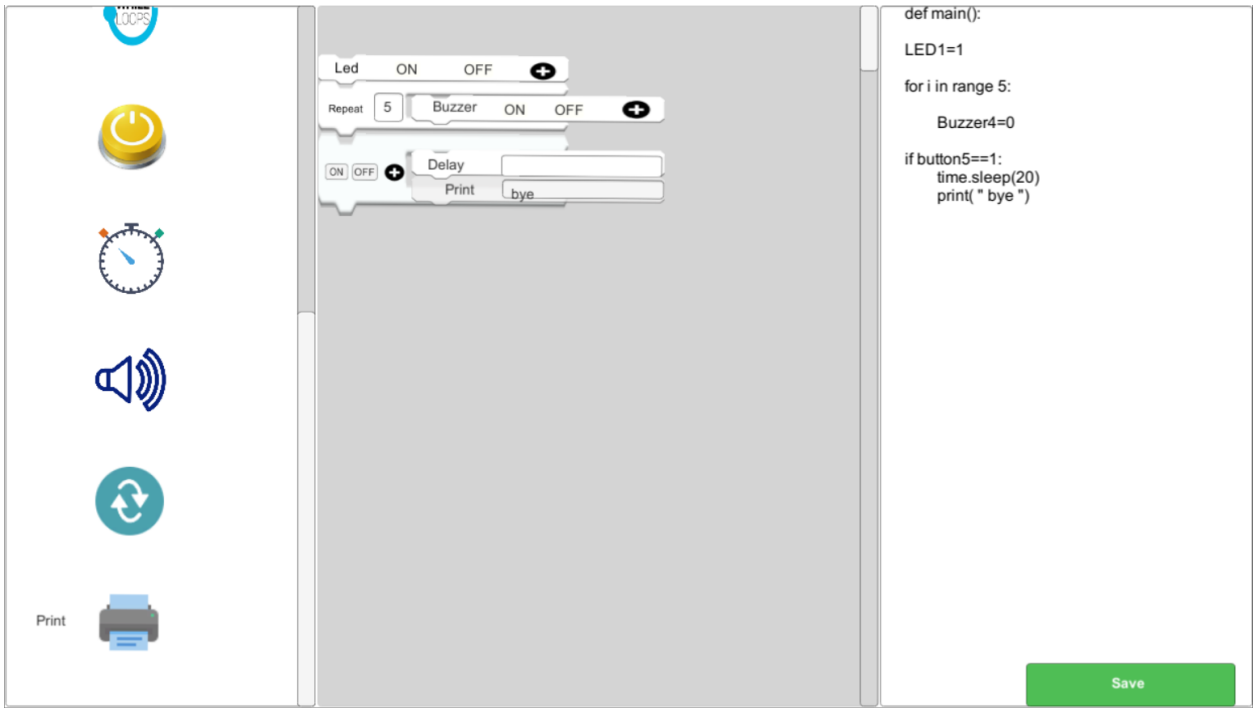


First scene

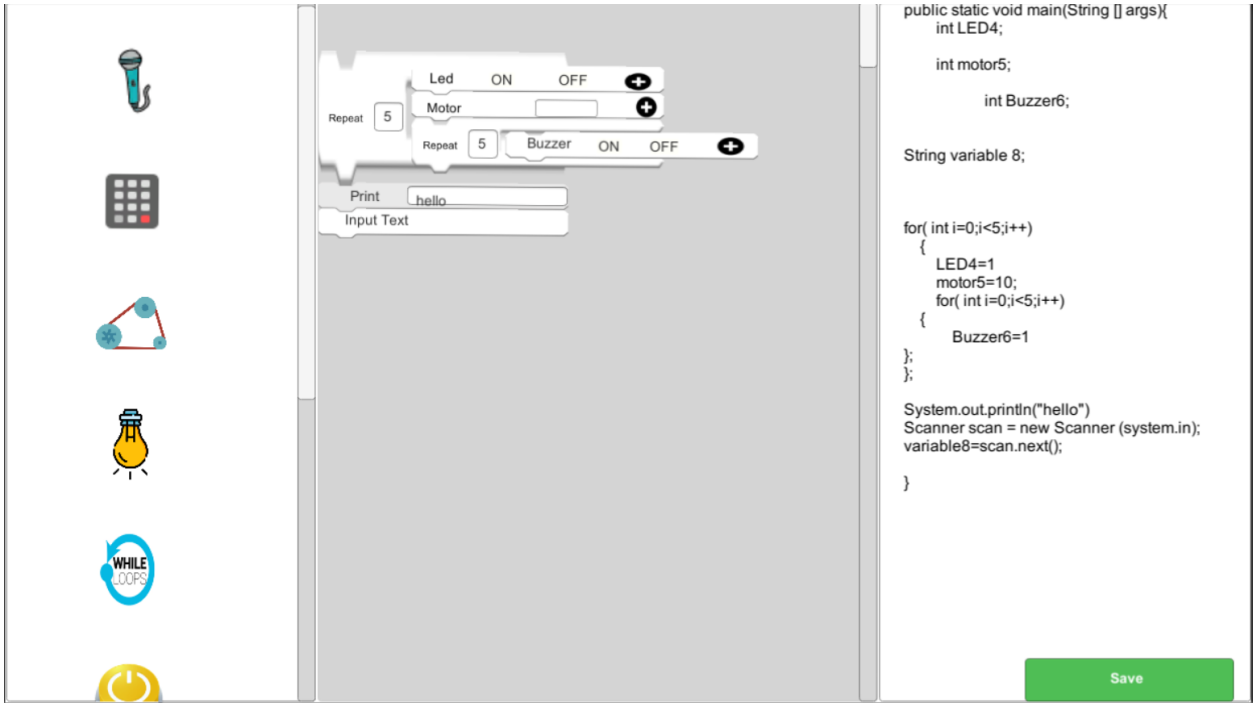


Second scene

Saved codes scene



Main scene python



Main scene java

6.3 Screen Objects and Actions

- play button object when pressed the first scene is changed to the second scene by disabling the first scene "play panel ".
- quit button object when pressed the game will be closed.
- select language object user can choose from two programming languages (java and python) and once he chooses his preferred language, language name will be saved by language controller in block and output code controller, so block or output code controller can use language name to get each block code with this language and make code appeared in the code panel in the correct language, after user selects his preferred language this will move us to main scene by disabling the second scene "select language panel" .
- create a project button object get the project name that will be entered by the user and give it to save coder component.
- back button object when pressed it will return the first scene back to the user by enabling the first scene "play panel".

- drag panel object which act as a container of all blocks user can use it.
- drop panel object which act as a container to blocks that user decide to use them in his program and select them from drop panel.
- code panel object which is responsible for the appearance of code of each block as a text once it is dropped in the droppable panel.
- block object which can be dragged by the user from drag panel and then dropped to drop panel also each block carry his own code that will appear in the code panel.
- save button object when pressed text(code) that exist in the code panel will be saved in a external file out of the game, as code saver component get the name of the project that is already saved from second scene and code from the code panel and save the code as a text in a file its name is name of the project.

7. REQUIREMENTS MATRIX

The following table shows each functional requirement and the corresponding component which implements the following component.

Functional requirement #	Functional requirement	Component
1.	Get object code of block	Block controller
2.	Get variable declaration	Block controller
3.	Instantiate block	Dragger
4.	Save code	Output code controller
5.	Generate code	Output code controller
6.	Maximize block	Dropper
7.	Choose programming language	Language controller
8.	Block duplication	Dragger
9.	Set parameters	Block controller

- The following table shows each test case and the corresponding functional requirement it`s testing

TestCase#	Technical requirement	Test case	Test steps	Test data	Expected output	Test result
1.	Get object code of LED	Test Object Code	• Drag a block • Drop in drop panel • Press on or Off • Press enter • Check code generated in code panel	1,0	LED=1 or 0	Pass
2.	Get object code of motor	Test Object Code with string/symbol input	• Drag a block • Drop in drop panel • Enter symbol/string • Press enter	X,/,%	No code generated	pass

			• Check code generated in code panel			
3.	Get object code of for/while	Test Object Code of for/while	• Drag a block • Drop in drop panel • Enter number of iterations • Drag blocks for body • Check code generated in code panel		Code generated with body of for/while	pass
4.	Get object code of for/while	Test Object Code of for/while	• Drag a block • Drop in drop panel • Enter number of iterations • Check code generated in code panel		Code generated without body of for/while	pass
5.	Get variable declaration of LED/Buzzer/motor/Mic in python	Test Object Variable Declaration In python	• Drag a block • Drop in drop panel • Check code generated in code panel		No variable declaration	Pass
6.	Get variable declaration of LED/Buzzer/motor/Mic in java	Test Object Variable Declaration In java	• Drag a block • Drop in drop panel • Check code generated in code panel		Int LED	Pass
7.	Instantiate block	Test Block Instantiation	• Drag a block • Drop in drop panel		Gameobject block	Pass
8.	Instantiate block	Test Block Instantiation	• Drag a block • Drop anywhere except dropPanel		No object instantiated	pass
9.	Save code	Test Save Code	• Drag a block • Drop in drop panel • Click save code button		.txt file on the local pc containing the generated code	pass
10.	Save code	Test Save	• Drag a block		.txt file on	pass

		Code With Empty Blocks	• Drop in drop panel • Click save code button		the local pc containing the generated code	
11.	Maximize block	Test block maximization of an expandable block	• Drag an expanded block • Drop in drop panel • Drag another block • Drop in the expanded block		Expanded block maximized to the preferred height of its child	pass
12.	Maximize block	Test block maximization of an unexpanded block	• Drag an unexpanded block • Drop in drop panel • Drag another block • Drop in the unexpanded block		Nothing change	pass
13.	Language controller	Test python language controller	• Choose python language from language page • Click create project • Drag a block • Drop in drop panel • Check code generated in code panel		Code generated in python	pass
14.	Language controller	Test java language controller	• Choose java language from language page • Click create project • Drag a block • Drop in drop panel		Code generated in java	pass

			• Check code generated in code panel			
15.	Language controller	Test no language controller	• Don`t choose any language from language page • Click create project • Drag a block • Drop in drop panel • Check code generated in code panel		No code generated	pass
16.	Block duplication	Test block duplication	• Drag a motor/LED/Buzzer • Drop in drop panel • Click on copy button		Block is copied in drop panel	pass
17.	Project creation	Test with correct name	• Enter play mode • Select programming language • Enter a string file name	File name	File is created with this name when clicking save in code panel	pass
18.	Project creation	Test with incorrect name	• Enter play mode • Select programming language • Enter file name that starts with a symbol or number		File not created and asks you to enter another name	fail
19.	Project creation	Test with empty name	• Enter play mode • Select programming language • Leave name field empty		File not created and asks you to enter name of your file	fail