Digital Steps

Digital steps

Digital steps game program designed to teach Java programming in stages! Whether you're a beginner or an experienced programmer, this program has got you covered.

	Member Name	ID
1.	Renad Alkhtani	2220003572
2.	Miad Alosaimi	2220001911
3.	Nourah Alanzi	2220000572
4.	Nada alrashidi	2220000552
5.	Waad Alshammari	2220001372
6.	Wajood Al Jearah	2220001292
7.	Sarah Alhethily	230040060

Introduction

The purpose of this report is to present a program designed to facilitate the learning of the Java programming language in an interactive and engaging manner. The program is structured to cater to users with varying levels of programming experience, starting from beginners and progressing through intermediate and advanced stages. By covering the basics of programming concepts and the Java language, this program aims to provide a comprehensive learning experience for individuals interested in acquiring or enhancing their Java programming skills.

Objectives

- **Making Learning Easy for Beginners:** Digital Steps aims to make learning easier for novice programmers by providing step-by-step tutorials. In addition, exercises to consolidate concepts. The goal is to help beginners overcome the initial challenges of learning programming and gain confidence in their abilities.

- **Skill Development:** For intermediate learners, Digital Steps focuses on expanding their knowledge of the Java language and improving their problem-solving skills. The app introduces intermediate concepts. For example, the Loop, the Array, the Method, and so on.

- **Competency development:** Digital Steps meets the needs of experienced programmers by offering advanced topics such as explaining classes, objects, relationships, inheritance, polymorphism, etc. The goal is to enable experienced developers to confidently handle complex Java projects efficiently.

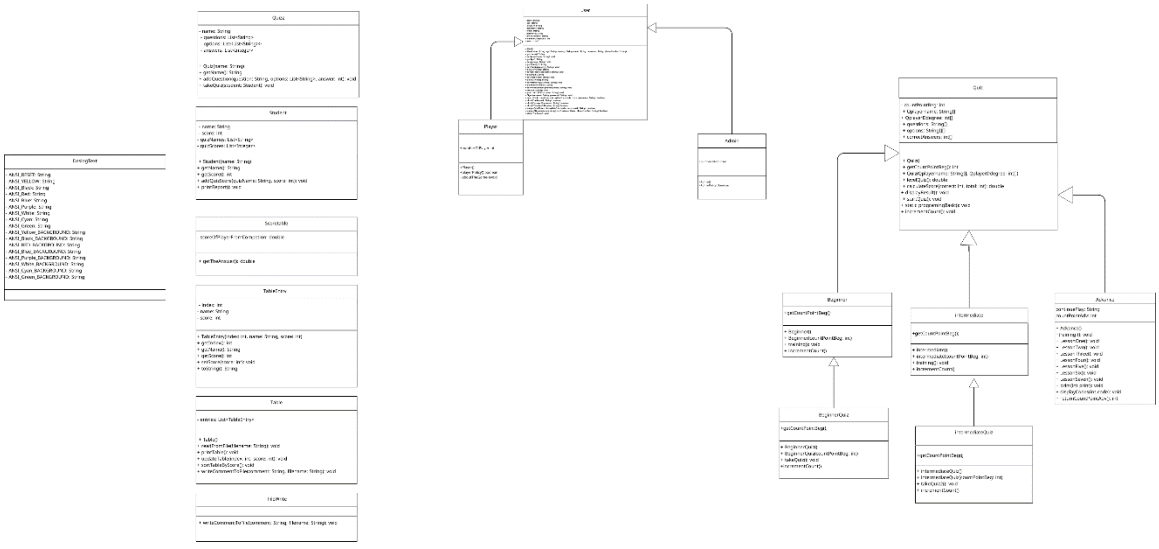
Targeted audience

Our main audience for DigitalSteps games are CS major students, programmers, and anyone interested in learning Java OOP. This encompasses beginner programmers seeking an interactive learning experience, self-learners looking for a gamified approach, experienced programmers transitioning to Java, and hobbyists exploring Java OOP for personal projects. The game aims to provide an engaging and immersive environment for individuals from various backgrounds who want to learn or enhance their Java OOP skills.

Reason for choosing the program

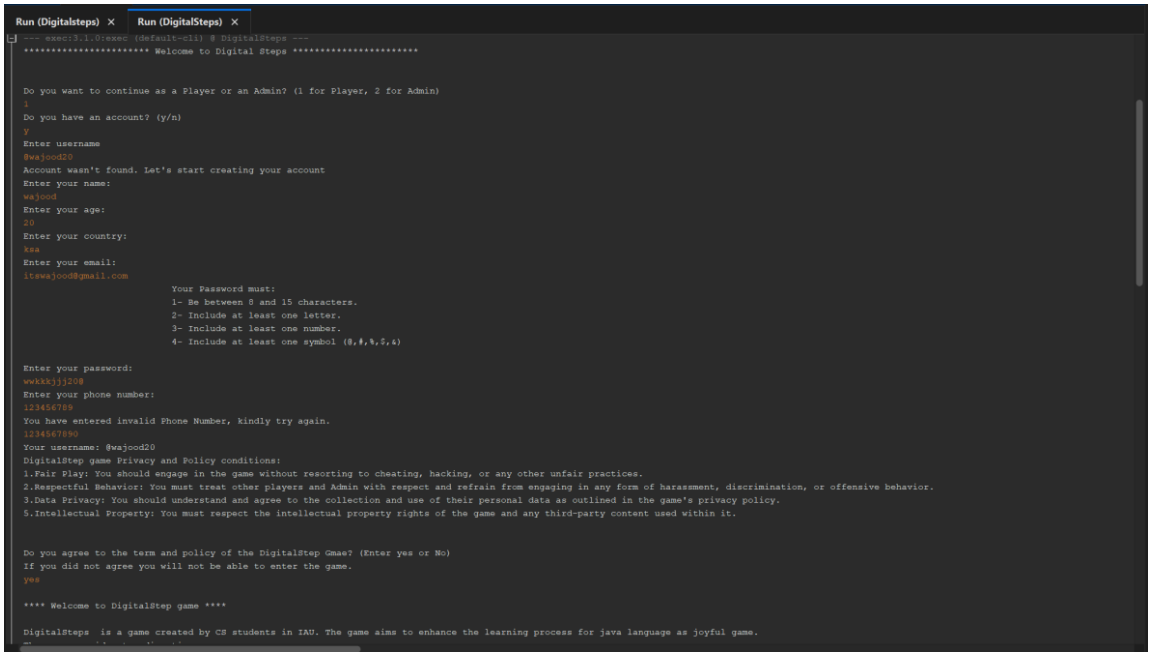
Digital Steps is a game that aims to teach the Java programming language interactively. The reason why we have chosen this program is that we aim to spread the concept of learning programming languages outside the framework of school and university education so that learning a programming language is fun, easy, and available to the public at all times. Also, we have chosen a game to learn programming languages in particular, as learning programming languages helps individuals acquire problem-solving skills, and programming languages with training are easy to understand, so our interactive game will make it easier for the player to learn Java language quickly.

UML

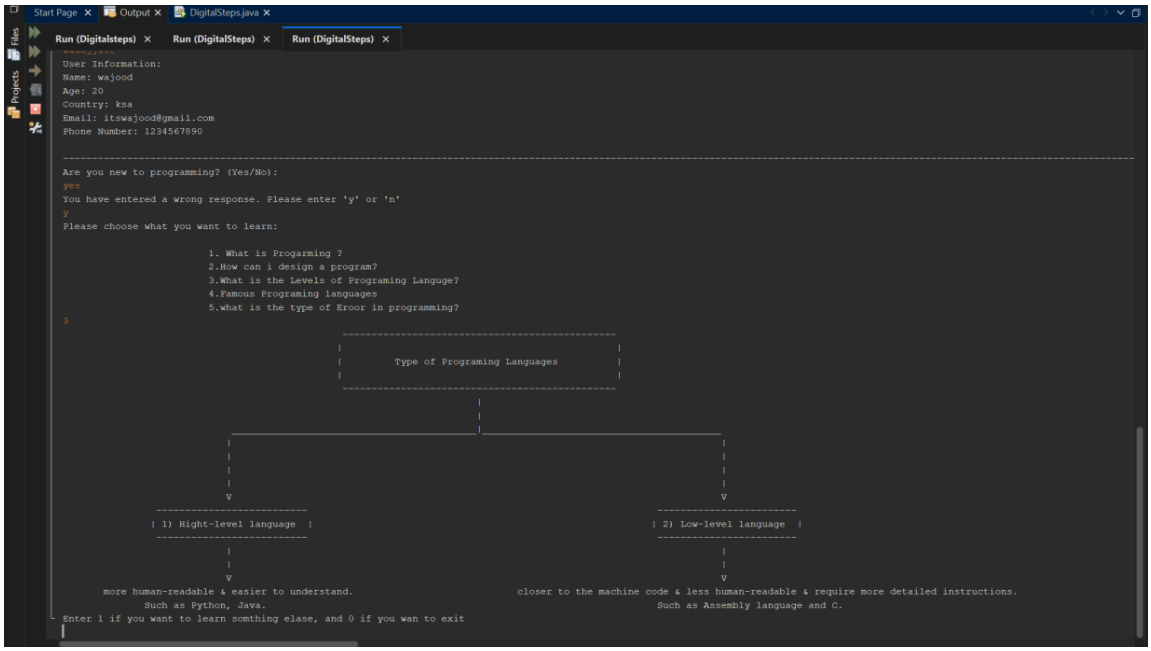


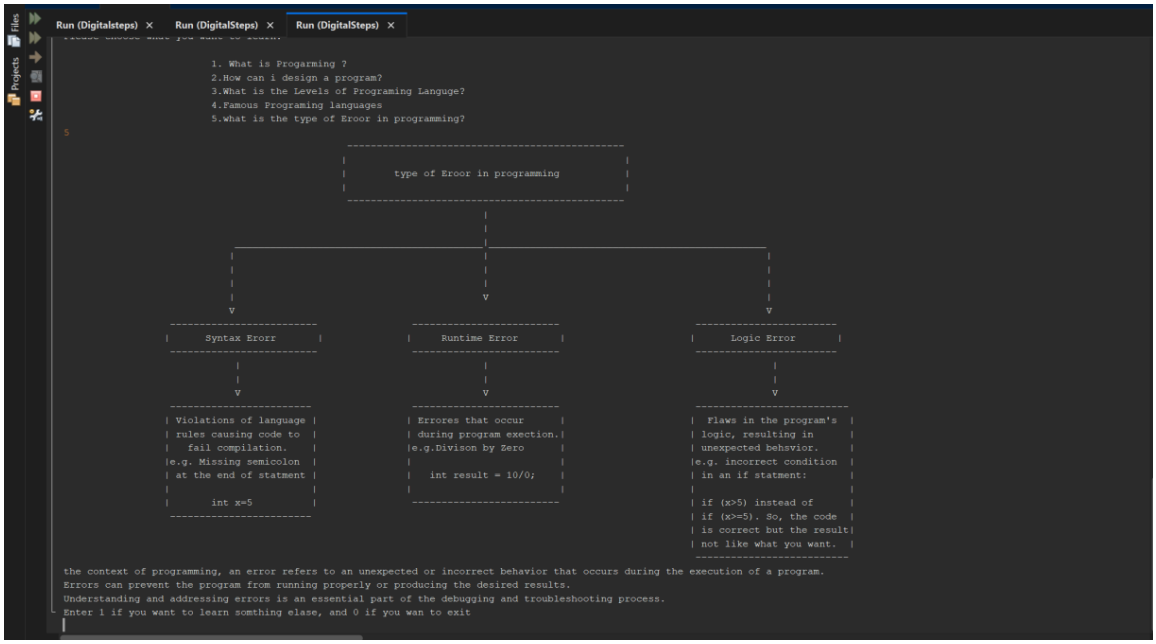
Interfaces

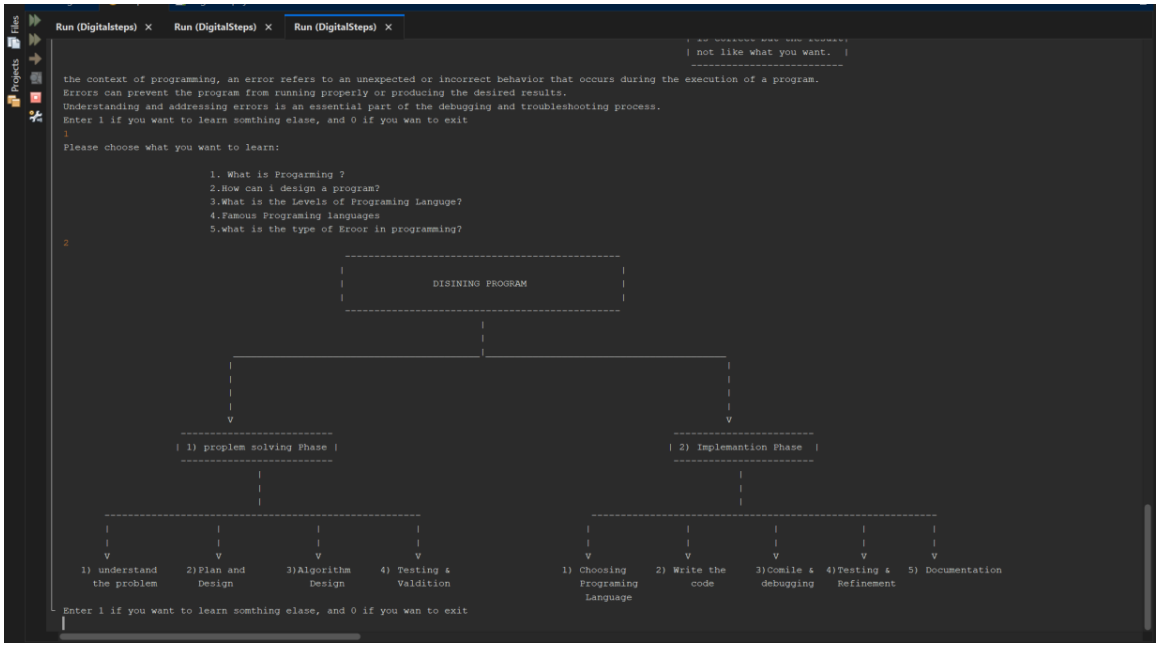
1- The beginning of the program interface:

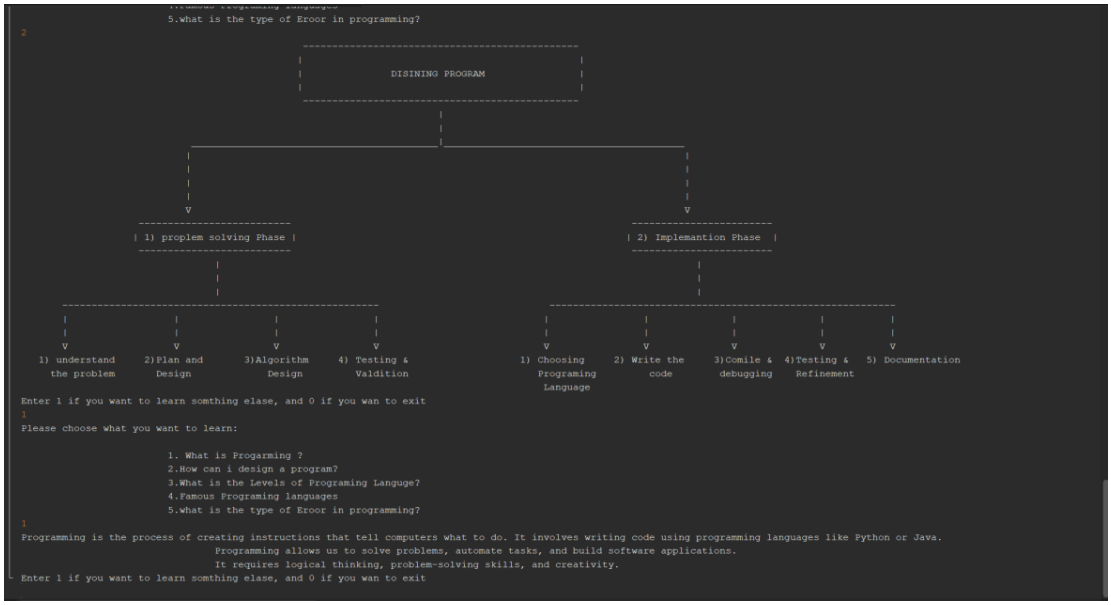


2- The Player part interface:

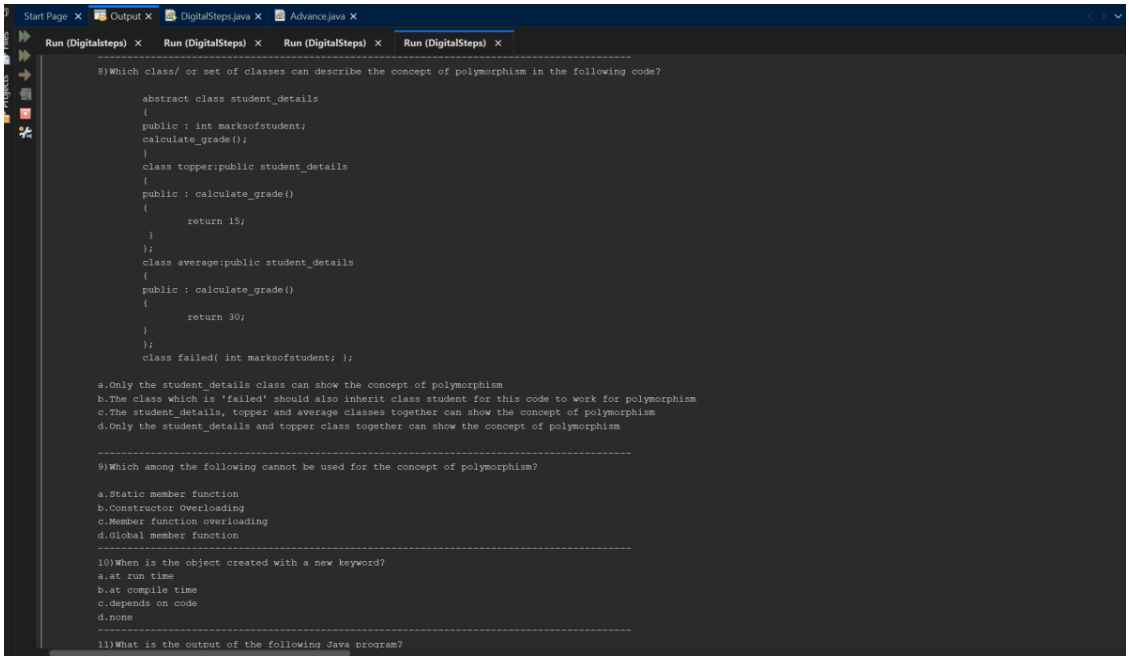



Run (Digitalsteps)
Run (DigitalSteps)
Run (DigitalSteps)
1. What is Progarming ?
2.How can i design a program?
3.What is the Levels of Programing Languge?
4.Famous Programing languages
5.what is the type of Eroor in programming?
5
type of Eroor in programming
Syntax Erorr
Violations of language rules causing code to fail compilation.
e.g. Missing semicolon at the end of statment
int x=5
Runtime Error
Errores that occur during program exection.
e.g.Divison by Zero
int result = 10/0;
Logic Error
Flaws in the program's logic, resulting in unexpected behsvior.
e.g. incorrect condition in an if statment:
if (x>5) instead of if (x>=5). So, the code is correct but the result not like what you want.
the context of programming, an error refers to an unexpected or incorrect behavior that occurs during the execution of a program.
Errors can prevent the program from running properly or producing the desired results.
Understanding and addressing errors is an essential part of the debugging and troubleshooting process.
Enter 1 if you want to learn somthing elase, and 0 if you wan to exit

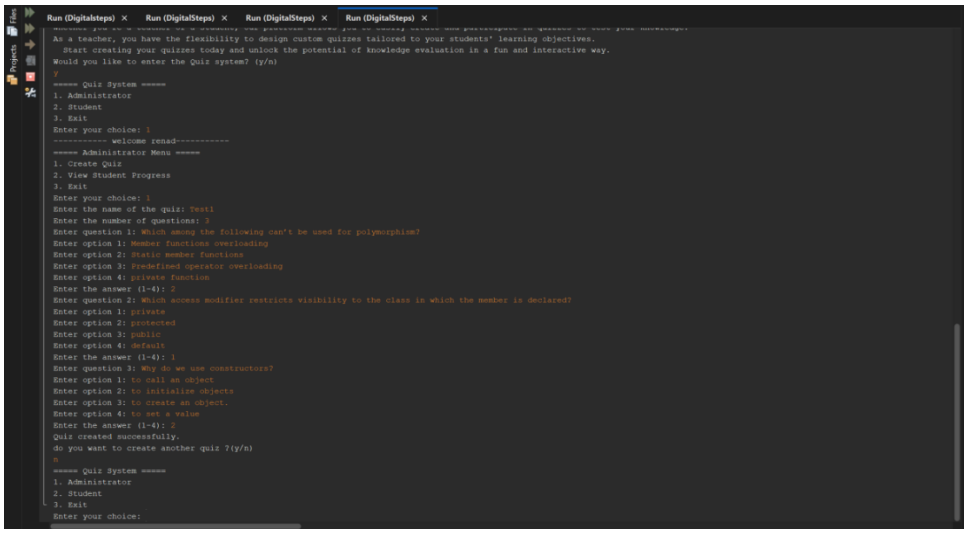

Run (Digitalsteps)
Run (DigitalSteps)
Run (DigitalSteps)
not like what you want.
the context of programming, an error refers to an unexpected or incorrect behavior that occurs during the execution of a program.
Errors can prevent the program from running properly or producing the desired results.
Understanding and addressing errors is an essential part of the debugging and troubleshooting process.
Enter 1 if you want to learn somthing elase, and 0 if you wan to exit
1
Please choose what you want to learn:
1. What is Progarming ?
2.How can i design a program?
3.What is the Levels of Programing Languge?
4.Famous Programing languages
5.what is the type of Eroor in programming?
2
DISINING PROGRAM
1) problem solving Phase
2) Implemantion Phase
1) understand the problem
2)Plan and Design
3)Algorithm Design
4) Testing & Valdition
1) Choosing Programing Language
2) Write the code
3)Comile & debugging
4)Testing & Refinement
5) Documentation
Enter 1 if you want to learn somthing elase, and 0 if you wan to exit

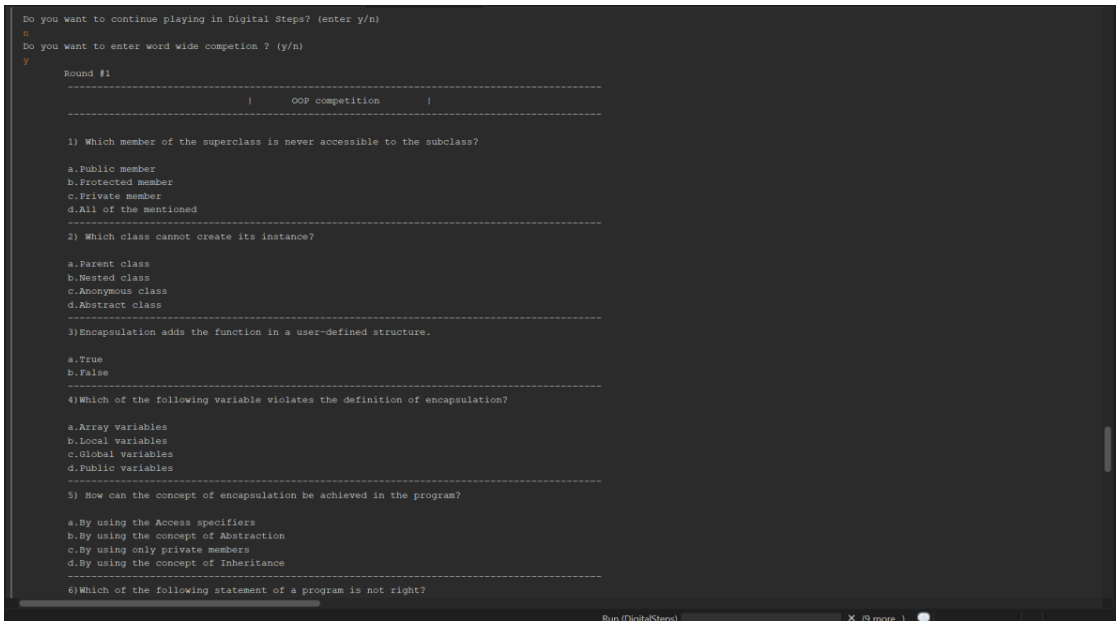

5.what is the type of Eroor in programming?
2
DISINING PROGRAM
1) proplem solving Phase
2) Implemantion Phase
1) understand the problem
2)Plan and Design
3)Algorithm Design
4) Testing & Valdition
1) Choosing Programing Language
2) Write the code
3)Comile & debugging
4)Testing & Refinement
5) Documentation
Enter 1 if you want to learn somthing elase, and 0 if you wan to exit
1
Please choose what you want to learn:
1. What is Progarming ?
2.How can i design a program?
3.What is the Levels of Programing Languge?
4.Famous Programing languages
5.what is the type of Eroor in programming?
1
Programming is the process of creating instructions that tell computers what to do. It involves writing code using programming languages like Python or Java.
Programming allows us to solve problems, automate tasks, and build software applications.
It requires logical thinking, problem-solving skills, and creativity.
Enter 1 if you want to learn somthing elase, and 0 if you wan to exit

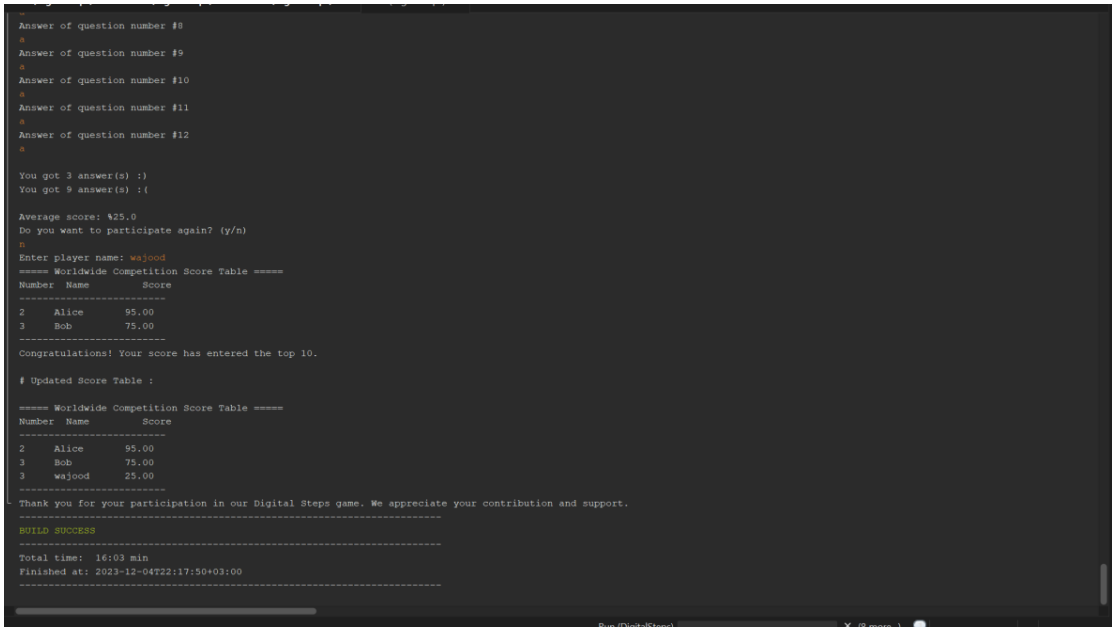

YOU ARE NOW IN PART ONE
Lesson 1
Syntax
Let's start with a simple program that prints "Hello World" to the screen.
Write this -> System.out.println("Hello World");
Note: Pay attention to uppercase and lowercase letters, as well as spaces
public void HelloWorld {
public static void main(String[] args){
write your code here: System.out.println("Hello World");
} }
Well done!
Go to next lesson? (y/n)
y



3- The Admin part interface:

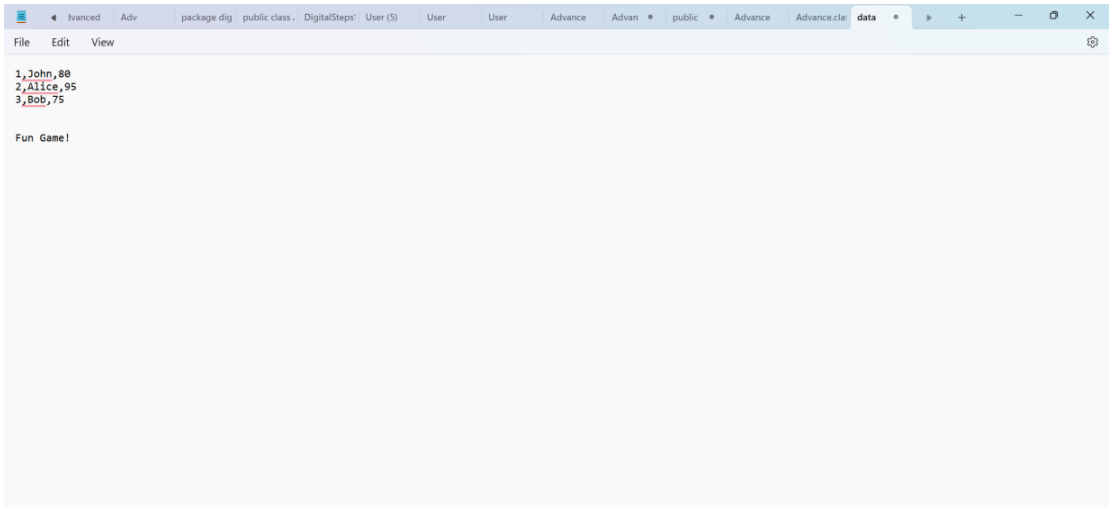



Do you want to continue playing in Digital Steps? (enter y/n)
n
Do you want to enter word wide competion ? (y/n)
y
Round #1
| OOP competition |
1) Which member of the superclass is never accessible to the subclass?
a.Public member
b.Protected member
c.Private member
d.All of the mentioned
2) Which class cannot create its instance?
a.Parent class
b.Nested class
c.Anonymous class
d.Abstract class
3)Encapsulation adds the function in a user-defined structure.
a.True
b.False
4)Which of the following variable violates the definition of encapsulation?
a.Array variables
b.Local variables
c.Global variables
d.Public variables
5) How can the concept of encapsulation be achieved in the program?
a.By using the Access specifiers
b.By using the concept of Abstraction
c.By using only private members
d.By using the concept of Inheritance
6)Which of the following statement of a program is not right?


Answer of question number #8
a
Answer of question number #9
a
Answer of question number #10
a
Answer of question number #11
a
Answer of question number #12
a
You got 3 answer(s) :)
You got 9 answer(s) :(
Average score: %25.0
Do you want to participate again? (y/n)
n
Enter player name: wajood
===== Worldwide Competition Score Table =====
Number Name Score
2 Alice 95.00
3 Bob 75.00
Congratulations! Your score has entered the top 10.
Updated Score Table :
===== Worldwide Competition Score Table =====
Number Name Score
2 Alice 95.00
3 Bob 75.00
3 wajood 25.00
Thank you for your participation in our Digital Steps game. We appreciate your contribution and support.
BUILD SUCCESS
Total time: 16:03 min
Finished at: 2023-12-04T22:17:50+03:00

```
Enter your answer (1-4), or 's' to skip, or 'b' to go back: 2
Correct answer!

Question 2:
Which access modifier restricts visibility to the class in which the member is declared?
1) private
2) protected
3) public
4) default
Enter your answer (1-4), or 's' to skip, or 'b' to go back: s
Question skipped.

Question 3:
Why do we use constructors?
1) to call an object
2) to initialize objects
3) to create an object.
4) to set a value
Enter your answer (1-4), or 's' to skip, or 'b' to go back: 2
Correct answer!

You have not answered all the questions. Do you want to submit the quiz? (y/n): n
Quiz not submitted.
===== Quiz System =====
1. Administrator
2. Student
3. Exit
Enter your choice: 2
Enter your name: wajood
Welcome back, wajood!
Your current score: 66
You can only answer quizzes and view your own score.
--------------------
Available Quizzes:
- Test1
Enter the name of the quiz you want to take: Test1
----- Test1 Quiz -----
You have already taken this quiz.
===== Quiz System =====
1. Administrator
2. Student
3. Exit
Enter your choice:
```

```
Run (DigitalSteps) ×  Run (DigitalSteps) ×  Run (DigitalSteps) ×  Run (DigitalSteps) ×

a
Answer of question number #9
a
Answer of question number #10
c
Answer of question number #11
a
Answer of question number #12
b

You got 4 answer(s) :)
You got 8 answer(s) :(

Average score: %33.33333333333333
Do you want to participate again? (y/n)
n
Enter player name: wajood
===== Worldwide Competition Score Table =====
Number  Name          Score
-------------------------
2     Alice       95.00
3     Bob         75.00
-------------------------
Congratulations! Your score has entered the top 10.

# Updated Score Table :

===== Worldwide Competition Score Table =====
Number  Name          Score
-------------------------
2     Alice       95.00
3     Bob         75.00
3     wajood      33.33
-------------------------
Enter a comment: Fun Game!
Thank you for your participation in our Digital Steps game. We appreciate your contribution and support.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  06:00 min
Finished at: 2023-12-04T22:25:40+03:00
------------------------------------------------------------------------
```



Challenges

We have faced many challenges working as a team in programming a Java project. Although the Java class was helpful while working spaertlty, we faced problems regarding the IDE versions, time limitations, code integration, and self-learning part problems. To elaborate, textFild object exists in the latest IDE version, which causes our program to show errors in the output statements when we collect the program in one device to test it; the reason was that the textFild object exists in some of our team member IDEs, and some of us do not. In addition, the self-learning part has caused errors that take a lot of time to fix and make it function correctly. However, our cooperation as a team in writing the program enabled us to overcome these challenges and produce highly efficient outcomes.

Class Descriptions

User Class

The User class is a super class for two subclasses Admin class, and Player class. It includes the main attributes and methods required to set up and manage user accounts, implementing various checks to ensure data validity and uniqueness. When a user interacts with this class, they engage in a process to create an account within the game. The class asks users for essential information like name, age, country, email, password, and phone number. While creating an account, each piece of data undergoes validation checks. For example, the email must match a specific format "checkEmail()", the password must meet specific conditions "checkPassword()", and the phone number must consist of ten digits "checkPNumber()"; Furthermore, it provides methods to print user information "printUserInfo()" and sign-in method that enter the users to their accounts based on their provided username and password "SignIn()". There's also an "aboutTheGame()" method, which informs users about the purpose and essence of DigitalSteps as an educational game centered around Object-Oriented Programming (OOP) principles in Java. Moreover, the number of created accounts is tracked through a static variable "numberOfAccounts". In general, the User class encapsulates the creation of user account management within the program, ensuring that accounts are created securely, following specified guidelines, and contributing to the program's functionality and integrity.

Attributes	**Description**
private String name	Stores the user's name.
private String age	Stores the user's age.
private String country	Stores the user's country.
private String username	Holds the user's generated username.
private String email	Stores the user's email address.
private String password	Stores the user's password.
private String phoneNumber	Stores the user's phone number.
public static int numberOfAccounts	Counts the total number of user accounts.
public User[] users	An array of User objects.
private ArrayList<User> userList	An ArrayList of User objects
Constructors	Description
public User()	Default constructor initializing the numberOfAccounts and users array.
public User(String name, String age, String country, String email, String password, String phoneNumber)	Parameterized constructor that sets user attributes (name, age, country, email, password, phoneNumber) and increments numberOfAccounts.
Methods	Description

public String getName()	Returns the user's name.
public void setName(String name)	Sets the user's name.
public String getAge()	Returns the user's age.
public void setAge(String age)	Sets the user's age.
public String getCountry()	Returns the user's country.
public void setCountry(String country)	Sets the user's country.
public String getUsername()	Generates and returns the user's username based on name and age.
public void setUsername(String username)	Sets the user's username.
public String getEmail()	Returns the user's email.
public void setEmail(String email)	Sets the user's email after validating uniqueness and validity.
public String getPassword()	Returns the user's password.
public void setPassword(String password)	**Sets the user's password after checking if the password meets the conditions.**
public String getPhoneNumber()	**Returns the user's phone number.**
public void setPhoneNumber(String phoneNumber)	**Sets the user's phone number after validation.**
public void createAccount()	**Make the user enter details to create an account, validating each input.**
public void printUserInfo(String username)	**Prints user information if the provided username matches.**
public void Sign(String username, String password)	**This method attempts to sign in a user by verifying the provided username and password.**
public static boolean searchForAccount(ArrayList<? extends User> users, String username)	**Searches for a user account by username within a list of users.**
public static boolean checkEmail(String email)	**Validates the email format.**
public static boolean checkPassword(String password)	**Validates the password format against certain criteria.**
public static boolean checkPNumber(String pNumber)	**Returns true only if the phone number is specifically 10 digits.**
public static boolean uniqueEmail(ArrayList<? extends User> users, String email)	**Check if an email is unique within a list of users.**
public static boolean uniquePNumber(ArrayList<? extends User> users, String phoneNumber)	**Check if a phone number is unique within a list of users.**
void aboutTheGame()	**Prints a textbox about the game.**

Player Class

The Player class is a subclass inherent from the super class called User. This class is used to store, display, and manage all information regarding the players. The class methods have high input validation since it handles any user input mistake successfully.

Attributes	Description
public static int numberOfPlayer	This attribute is counter for the number of players.
Methods	**Description**
public Player ()	Default constructor with numberOfPlayer counter increment.
public boolean playerPolicy ()	This method verifies the acceptance of the user for game term and policy, and it contains input validation on user answer.
void aboutTheGame ()	This method displays basic information about the game for the player.

Advance Class

Advance class, is a class that is dedicated to the advanced level of programming. At this level, the user is expected to learn the main concepts of Object-Oriented Programming (OOP). The main concepts such as classes and objects, inheritance, and polymorphism are covered in seven lessons. Each lesson provides a full explanation of the topic, and during the lesson the player is expected to be asked some various questions to make sure that player is understanding the subject. Moreover, after each lesson the program asks the player if they want to continue to the next lesson or exit the game. Overall, the Advance class provides a comprehensive exploration of advanced programming, emphasizing essential Object-Oriented Programming concepts in seven lessons. The class, features interactive questions and the option to progress or exit after each lesson, enhances the overall learning experience.

Attributes	Description
Scanner input	This attribute represents a Scanner object used for input operations within the Advance class.
String continuePlay	String variable used to store user input for continuing or ending the lessons.
static int countPointAdv	Static integer variable used to track and store the count of points achieved during the Advance level training.
Method	Description
public void training()	This void method is for the Advance Level training, it has the calling for the seven OOP lessons. After each lesson, the user gets asked if he wants to continue the learning game or exit the game.
public void LessonOne()	Teaches about classes and objects, including creating classes, and objects, and verifying user understanding.
public void LessonTwo()	Focuses on constructors, their types, and their usage within Java classes.
public void LessonThree()	Covers the access modifiers.
public void LessonFour()	This method teaches about getter and setter and how to access private fields in a class.
public void LessonFive()	Covers class relationships including abstraction, encapsulation, associations, aggregations, and compositions.
public void LessonSix()	Explains inheritance, its benefits, how to inherit from a class, and cautions related to multiple inheritance in Java.
public void LessonSeven()	Discusses polymorphism, how it uses inheritance, and presents an example of its usage.
public void print(int print)	Prints lines for designing the output, based on the parameter provided.
public void innCorrectAnsw()	Displays an incorrect answer message and asks the user to try again or exit.
public void displayCodes(int code)	Prints specific code snippets used during the lessons based on the provided parameter.
public int returnCountPointAdv()	Returns the earned points during the Advance level.

Admin Class

The admin class is a subclass inherent from the super class called User. This class is used to store, display, and manage all information regarding the admins. The class methods have high input validation since it handles any user input mistake successfully.

.

Attribute	Description
public static int numberOfAdmin	This attribute is counter for the number of Admin.
Method	**Description**
Public Admin()	Default constructor
Public boolean AdminPolicy()	This method verifies the acceptance of the user for game term and policy and it contain input validation on user answer

Student Class

The Student class represents a student and provides methods to retrieve the student's name, score, and add quiz scores. It also has a method to print a report showing the student's quiz names, scores, and the total score.

Attributes	Description
Private String name	**Stores the name of the student.**
Private int score	**Represents the total score of the student.**
Private quizNames: List<String>	Stores the names of quizzes taken by the student.
Private quizScores: List<Integer>	Stores the scores obtained by the student in each quiz.
Methods	**Description**
Public Student(String name)	Constructor: Initializes the student object with a name and sets the score to 0. It also initializes the quizNames and quizScores lists.
Public String getName()	Returns the name of the student.
Public int getScore()	Returns the total score of the student.
Public void addQuizScore(String quizName, int score)	Adds the quiz name and score to the respective lists. It also updates the total score of the student.
Public void printReport()	Prints a report for the student, displaying the quiz names and scores.

Beginner Class

In our beginner level programming class, we cover ten essential lessons that encompass the fundamental principles of Java programming as well as numerous other programming languages. These lessons focus on crucial concepts such as variables, conditionals and switch statements. Each lesson is designed to provide a clear explanation of the topic at hand, followed by a series of practice questions to reinforce understanding. Furthermore, we have taken measures to ensure that user errors in input are promptly identified and addressed in order to enhance the learning experience. we aim to help students avoid common mistakes and solidify their grasp of the material. Additionally, our program incorporates a points system that rewards users for correctly completing the training exercises. Each time a user provides the correct solution to a training question, they earn points based on their accuracy. This feature not only motivates users to actively engage with the material but also allows them to track their progress and measure their proficiency in Java programming. Overall, our program offers a comprehensive introduction to Java programming and serves as a solid foundation for further exploration in the world of programming. By gamifying the learning experience through the point system, we aim to create a more interactive and enjoyable environment for our students. It encourages them to strive for accuracy and provides a sense of achievement as they accumulate points throughout the course.

Intermediate Class

In our Intermediate level programming class, we did the same as beginner class.

Quizz Class

The Quiz class represents a quiz and allows students to take the quiz. It has methods to add questions, options, and answers to the quiz. The takeQuiz method displays each question, presents options, and records the student's answers. It calculates the score based on the answers and allows for skipping or going back to previous questions. It also handles quiz submission and stores the score in the student object.

Attributes	Description
Private String name	Stores the name of the quiz.
Private questions: List<String>	Stores the questions in the quiz.
Private options: List<List<String>>	Stores the options for each question.
Private answers: List<Integer>	Stores the correct answers for each question.
Methods	Description
Public Quiz(String name)	Constructor: Initializes the quiz object with a name. It also initializes the questions, options, and answers lists.
Public String getName()	Returns the name of the quiz.
Public void addQuestion(String question, options: List<String>, int answer)	Adds a question, options, and the correct answer to the respective lists.
Public void takeQuiz(Student student)	Allows a student to take the quiz. It displays each question, prompts for an answer, and calculates the score. It also handles skipping questions and going back to the previous question. The score is stored in the student object.

Quiz Class

This class represents a quiz with methods to start the quiz, calculate scores, display results, and provide learning materials for programming topics.

Attributes	Description
private int countPointBeg:	this is a private integer attribute that represents the count of points at the beginning of the quiz. It is accessible only within the class.
public String[] Qplayername	This is a public array of strings that represents the names of the players in the quiz.
public int[] QplayerDdegree	This is a public array of integers that represents the degree or score of the players in the quiz.
public String[] questions	This is a public array of strings that represents the questions in the quiz.
public String[][] options	This is a public two-dimensional array of strings that represents the options for each question in the quiz.
public String[][] options	This is a public two-dimensional array of strings that represents the options for each question in the quiz.
Methods	**Description**
public Quiz():	This is a constructor method that initializes the countPointBeg attribute to 0.
public int getCountPointBeg():	This method returns the value of the countPointBeg attribute.
public Quiz(String[] Qplayername, int[] QplayerDdegree)	This is a constructor method that takes an array of player names and an array of player degrees as parameters. It initializes the Qplayername and QplayerDdegree attributes with the provided values.
public double calculateScore(int correct ,int total)	This method takes the number of correct answers and the total number of questions as parameters and calculates the score by multiplying them .It returns a double value representing the score

public void displayResult()	This method displays the quiz result . It prints the player name ,score ,and the total number of questions answered .It uses the Qplayername and QplayerDdegree attributes.
public void startQuiz()	-This method starts the quiz .It initializes the questions ,options ,and correctAnswers attributes with predefined values .It prompts the user to answer each question ,checks if the answer is correct , and calculates the score .It updates the Qplayername and QplayerDdegree attributes with the player's name and score .It then calls the displayResult() method to display the result
public static void programingBasic()	This method provides a menu for It prompts .learning programming topics the user to choose a topic and displays It includes .the corresponding content ,options for learning about programming levels of programming ,program design famous programming ,languages and types of errors in ,languages programming.
public void incrementCount()	This method increments the value of the countPointBeg attribute.

Scoretable Class

This class represents a score table. It provides a static method getTheAnswer() which prompts the user to answer a series of questions, calculates the number of correct and incorrect answers, and returns the player's score as a percentage.

Attributes	Description
Private double scoreOfPlayerFromCompetion	
Methods	**Description**
Public double getTheAnswer()	

TableEntry Class

This class represents an entry in the score table. It stores the index, name, and score of a player. It provides methods to get and set these values and overrides the toString() method to format the entry as a string.

Attributes	Description
Private int index	An integer representing the index of the entry.
Private String name	A string representing the name associated with the entry.
Private int score	A double representing the score associated with the entry.
Methods	**Description**
Public TableEntry(int index, String name, int score)	A constructor that initializes the index, name, and score attributes.
Public int getIndex()	Returns the index of the entry.
Public String getName()	Returns the name associated with the entry.
Public int getScore()	Returns the score associated with the entry.
Public void setScore(int score)	Updates the score of the entry.
Public String toString()	Overrides the toString() method to return a formatted string representation of the entry.

Table Class

This class represents the score table itself. It maintains a list of TableEntry objects and provides methods to read entries from a file, print the table, update an entry's score, sort the table by score, and write comments to a file.

Attributes	Description
Private entries: List<TableEntry>	A list of TableEntry objects representing the entries in the table.
Methods	**Description**
Public Table()	A constructor that initializes the entries attribute as an empty list.
Public void readFromFile(String filename)	Reads entries from a file and populates the entries list. Each line in the file represents an entry with the format index, name, score.
Public void printTable()	Prints the table with the headers and all the entries.
Public void updateTable(int index, int score)	Updates the score of the entry with the given index.
Public void sortTableByScore()	Sorts the entries in the table in descending order based on their scores.
checkAndUpdateTopScores(score: double, playerName: String): void:	Checks if the given score qualifies for the top 10 scores in the table. If so, it adds the entry to the table, updates the table, and displays a congratulatory message.
Public void writeCommentToFile(String comment , String filename)	Writes a comment to a file.

FileWrite Class

This class provides a method to write a comment to a file. It is used by the Table class to write comments to the score table file.

Methods	Description
Public void writeCommentToFile(String comment, String filename)	This method takes a comment and a filename as parameters and writes the comment to the specified file using a PrintWriter and FileWriter. If an IOException occurs, an error message is printed.

DesingText Class

a class named DesingText that declares and initializes ANSI escape codes for various colors and background colors. It includes constants for resetting the color, as well as for different color and background combinations, such as yellow, black, red, blue, purple, white, cyan, and green. The class does not contain any methods or behavior; it solely serves as a collection of constants representing ANSI escape codes for text and background color styling in the console.

Attribute	description
ANSI_RESET	Stores the ANSI escape code for resetting the color.
ANSI_YELLOW	Stores the ANSI escape code for yellow text color.
ANSI_Black	Stores the ANSI escape code for black text color.
ANSI_Red	Stores the ANSI escape code for red text color.
ANSI_Blue	Stores the ANSI escape code for blue text color.
ANSI_Purple	Stores the ANSI escape code for purple text color.
ANSI_White	Stores the ANSI escape code for white text color.
ANSI_Cyan	Stores the ANSI escape code for cyan text color.
ANSI_Green	Stores the ANSI escape code for green text color.
ANSI_Yellow_BACKGROUND	Stores the ANSI escape code for yellow background color.
ANSI_Black_BACKGROUND	Stores the ANSI escape code for black background color.
ANSI_RED_BACKGROUND	Stores the ANSI escape code for red background color.
ANSI_Blue_BACKGROUND	Stores the ANSI escape code for blue background color.
ANSI_Purple_BACKGROUND	Stores the ANSI escape code for purple background color.

ANSI_White_BACKGROUND	Stores the ANSI escape code for white background color.
ANSI_Cyan_BACKGROUND	Stores the ANSI escape code for cyan background color.

Future work

Our project aspires to develop in many aspects in the future. For instance, we aim to expand the project to include more than one programming language, also we aspire to include university courses to make the learning process joyful and more creative rather than the ordinary learning technique.

Conclusion

In brief, Digital Steps is an exceptional educational application suitable for programmers at all levels. It provides a comprehensive curriculum, adaptive learning features, and interactive content. By facilitating beginner-friendly learning, enhancing skill development, and promoting advanced proficiency, Digital Steps empowers individuals to effectively master the Java language. With Digital Steps, programmers can unleash their full potential in the world of Java programming.

Github link :

https://github.com/RenadQ/Digital-Steps-Game.git