Understanding program (Source Code) Structure

STUDENT PAYMENT SYSTEM

LP | Programming Language and Paradigms | March 26, 2023

Names of people who contributed in this assignment include;

Luyando Pongolani, Christopher Beza, Hassan Phiri, Moses Simataa, Abel Malusa and Paul Chanda.

I believe everyone contributed equally in this assignment and we hope in the next project everyone will have the necessary software's for GitHub commit track.

Contents

Chapter 1

Introduction

Student payment system software is a software vastly used among other software's in today's world, in the sense that, education is the core of civilization. Having an educated people in a country ensures that they can improve the country's welfare and economy in a variety of ways than dimmed before and maintaining student's details, data and the tracking of their payments would help reduce human error and speed up the process of registering of a student.

Chapter 2

Objective

The objective of the student payment system is to make it easy to manage student's payment data. The software provides objectives as follows;

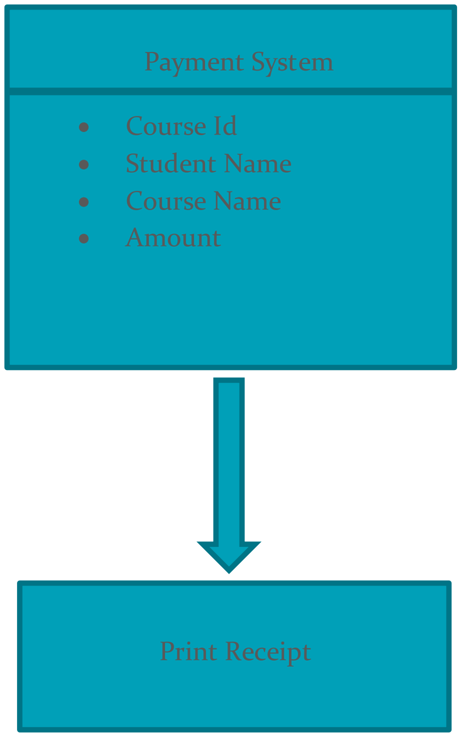
- Store student's data
- Add student's information
- Print out student's payment information
- Making payments
- Make payment system easier
- Help maintain account's easy

Chapter 3

System Design

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. Since this code used as source code to print out AST's is a portion of the actual source code of a student payment system, the system design that will be displayed below would be one of exactly where it starts.

Below is a UML diagram of the design.



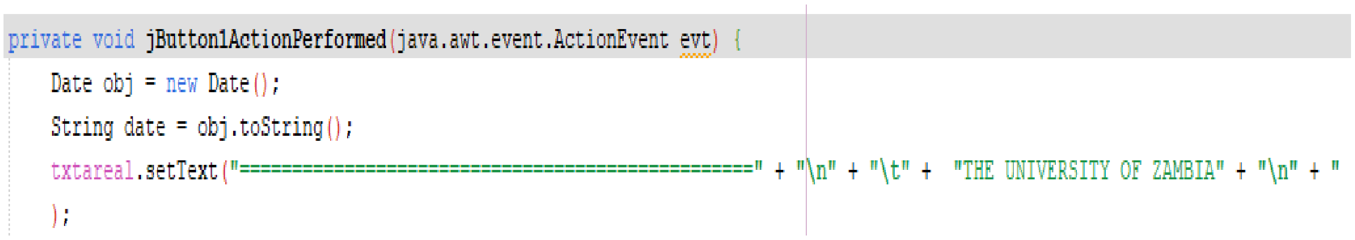
Chapter 4

Implementation

Here we look at how the software source code works in detail and also show how each segment of code is implemented.

Header

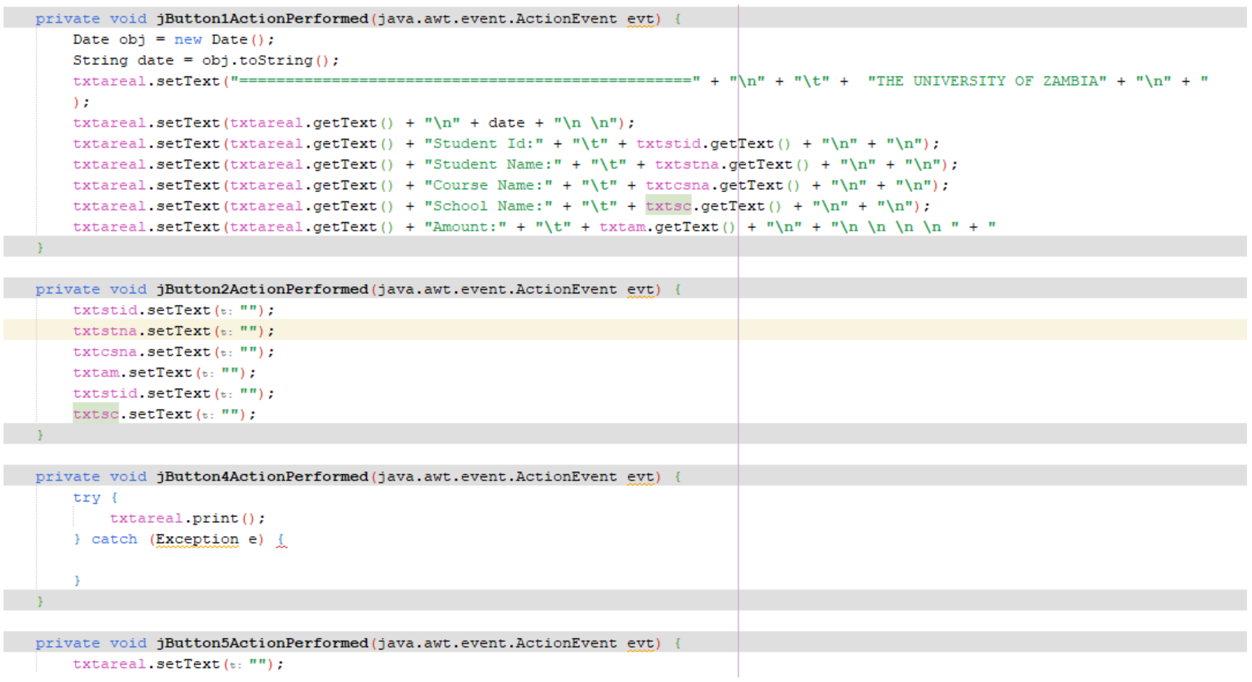
The header displayed below is embeded in the code of the add buttom action. Had to get a little creative as when the user is done inputing the information, they can always reset so as to input the next student information.



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    Date obj = new Date();
    String date = obj.toString();
    txtarea1.setText("=================================================" + "\n" + "\t" +  "THE UNIVERSITY OF ZAMBIA" + "\n" + "
    );
```

Buttons

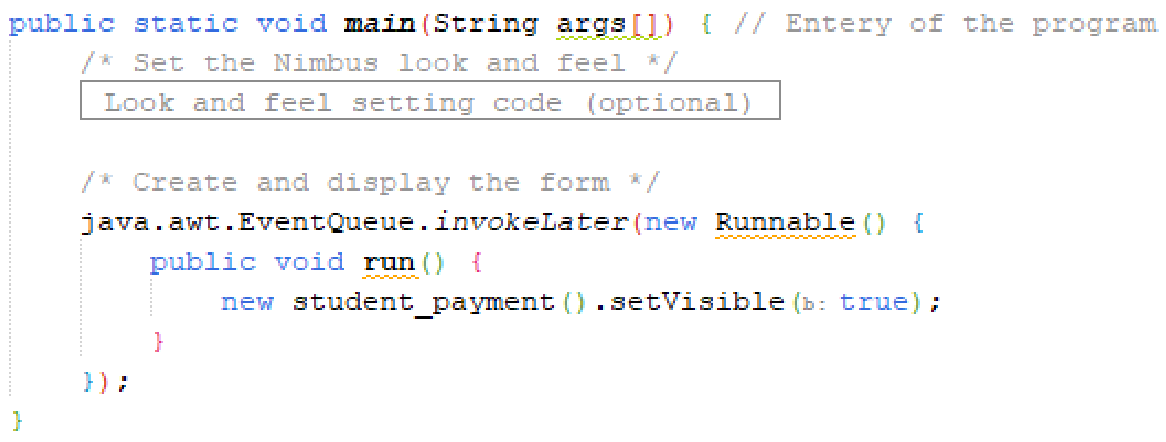
Below lies the buttons source code GUI that show’s how each button when clicked display’s its action assigned role.



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    Date obj = new Date();
    String date = obj.toString();
    txtareal.setText("==================================================" + "\n" + "\t" +  "THE UNIVERSITY OF ZAMBIA" + "\n" + "
    );
    txtareal.setText(txtareal.getText() + "\n" + date + "\n \n");
    txtareal.setText(txtareal.getText() + "Student Id:" + "\t" + txtstid.getText() + "\n" + "\n");
    txtareal.setText(txtareal.getText() + "Student Name:" + "\t" + txtstna.getText() + "\n" + "\n");
    txtareal.setText(txtareal.getText() + "Course Name:" + "\t" + txtcsna.getText() + "\n" + "\n");
    txtareal.setText(txtareal.getText() + "School Name:" + "\t" + txtsc.getText() + "\n" + "\n");
    txtareal.setText(txtareal.getText() + "Amount:" + "\t" + txtam.getText() + "\n" + "\n \n \n \n " + "
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    txtstid.setText(t: "");
    txtstna.setText(t: "");
    txtcsna.setText(t: "");
    txtam.setText(t: "");
    txtstid.setText(t: "");
    txtsc.setText(t: "");
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        txtareal.print();
    } catch (Exception e) {

    }
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    txtareal.setText(t: "");
```

Entry of the Program

Attached below is the source code of the main class which is the entry of the program upon execution.

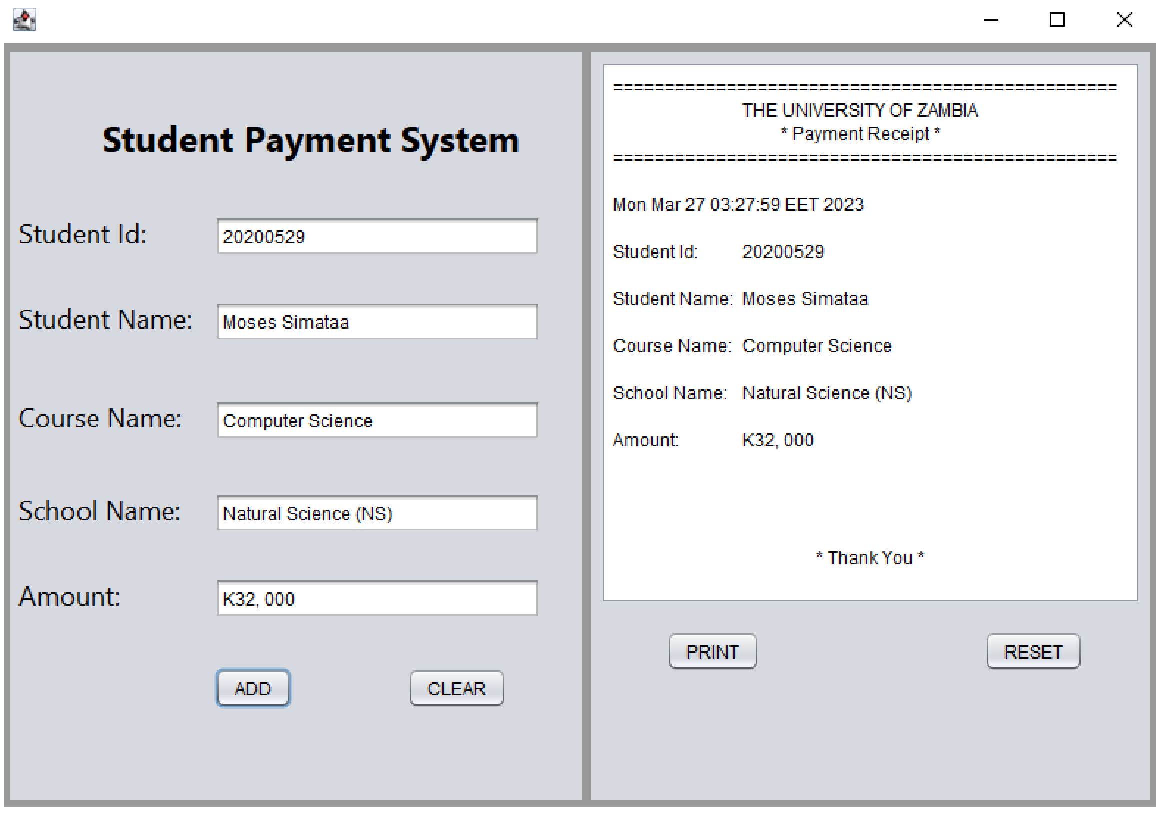


```
public static void main(String args[]) { // Entery of the program
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new student_payment().setVisible(b: true);
        }
    });
}
```

Chapter 5

Result

The result is the Graphical interface part of the assignment. Attached below is the GUI of the student payment system.



Student Payment System

Student Id:	20200529
Student Name:	Moses Simataa
Course Name:	Computer Science
School Name:	Natural Science (NS)
Amount:	K32, 000

```
==================================================
            THE UNIVERSITY OF ZAMBIA
               * Payment Receipt *
==================================================

Mon Mar 27 03:27:59 EET 2023

Student Id:      20200529

Student Name:  Moses Simataa

Course Name:   Computer Science

School Name:   Natural Science (NS)

Amount:          K32, 000



                     * Thank You *
```

Chapter 6

JavaParser

Abstract Syntax Tress (AST)

JavaParser is an open-source library that allows interactions with Java source code through a Java abstract syntax tree (AST) [62]. The library helps parsing source code and provides aid to navigate around the AST, giving programmers the ability to traverse the code without having to write the tree traversal code from scratch. JavaParser can also unparse—that is, pretty print—an AST back to Java source code. The fundamental feature of the library is to provide programmers the capability of building their own code by manipulating the structure of the source code. The library is a strong tool to analyse, transform and generate code base. Below is the attached Google drive link containing the files of the AST and the others for your perusal.
https://drive.google.com/drive/u/2/folders/1ayViP6LZLIQNVIvWJwXhAVMoCU8HjSny
Or
drive.google.com/drive/u/2/folders/1ayViP6LZLIQNVIvWJwXhAVMoCU8HjSny

The Total Number of Tokens in the Source Code

At the lexical structure, computer programs are viewed as simple sequences of lexical items called tokens. Tokens can be classified as being either;

- Identifiers
- Keywords
- Operators
- Separators
- Literals
- Comments

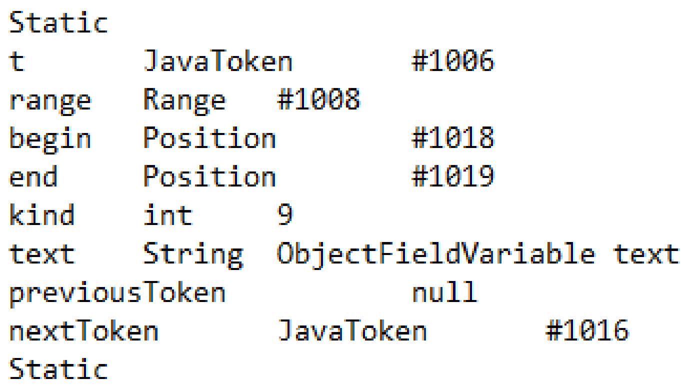
Every byte is regarded as token unless in situations like whitespaces.

I used toggle line breakpoint to see the total number of tokens in terms of range begin and end and also used the getTokenRange() to obtain the names of the tokens present. Please find attached the Google drive link as well as the print taken from the IDE.

https://drive.google.com/drive/u/2/folders/1ayViP6LZLlQNVIvWJwXhAVMoCU8HjSny

Or

drive.google.com/drive/u/2/folders/1ayViP6LZLlQNVIvWJwXhAVMoCU8HjSny



```
Static
t       JavaToken       #1006
range   Range   #1008
begin   Position        #1018
end     Position        #1019
kind    int     9
text    String  ObjectFieldVariable text
previousToken           null
nextToken       JavaToken       #1016
Static
```

Types of Tokens and how many they are of each type

As stated above, tokens are classified into six categories;

- Identifiers
- Keywords
- Operators
- Separators
- Literals
- Comments

Now the challenge lies in counting and classifying each according to its type. A work around I did was to use the AST as it shows 3 of the classified tokens, so I converted the txt printed file and searched each token value type and using adobe it counted exactly how many they were. Below is the data obtained.

Identifiers = 1126

Keywords = 36

Operators = 59

Comments = 21

Hence The Total number of tokens obtained from the lexical structure point of view above is 1,242.

Chapter 7

Conclusion

JavaParser is a good jar file that enables us to check for the lexical structure and as well as the AST structure among other tools it provides. It's able to check for identifiers present in the code, operators, keywords etc. getting the tokens one at a time by analysing the code has dimmed to be tedious but has also proven that tokens obtains from the code manually tend to be a lot compared to the processed ones (JavaParser, n.d.) (GitHub, n.d.)

References

GitHub. (n.d.). Retrieved from GitHub: https://github.com/javaparser/javaparser

JavaParser. (n.d.). Retrieved from JavaParser: https://javaparser.org/