**JAVA PROJECT REPORT**

(Project Term January-May 2023)

**NUMBER CONVERTER SYSTEM**

Submitted by

**Pruthvidhar baddam Registration Number :12112847**

**Project Group Number 5**

**Course Code: CSE 310**

Under the Guidance of

# Dr.Ranjith kumar

# School of Computer Science and Engineering



**DECLARATION**

We hereby declare that the project work entitled (“**NUMBER CONVERTER SYSTEM**”) is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in COMPUTER SCEINCE from Lovely Professional University, Phagwara, under the guidance of (Dr.Ranjith kumar), during January to may 2023. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number: …5………

**Name of Student : Pruthvidhar Baddam**

**Registration Number: 12112847**

Date:20/04/23

Pruthvidhar

(Signature of Student 2)

TABLE OF CONTENT

* Abstract
* introduction
* scope of project
* modules
* design
* screen shot
* explanation
* sample code
* outcome

ABSTRACT

A number converter is a software that can convert numbers from one base system to another. It can be used to convert binary, decimal, hexadecimal, and other number systems. The conversion process involves changing the number's format while keeping its value intact. This is useful for various applications, including computer programming, engineering, and mathematicsand networking

The number converter works by taking the input number and converting it to a common base system, usually decimal. From there, the number can be converted to the desired base system. The conversion process may involve using algorithms and mathematical operations to ensure accurate and efficient conversion.

Number converters are widely available online and as software applications. They are simple to use and can save time and effort when working with numbers. With a number converter, users can easily switch between number systems, enabling them to perform complex calculations and operations with ease.

INTRODUCTION

Number system’;become increasingly popular as computers and digital technology have become more prevalent in our daily lives. There are several different numerical systems, including binary, decimal, octal, and hexadecimal, each with its unique representation and use cases.

numerical systems. In the following sections, we will explore different types of number conversions and how they work

* They are widely used in fields such as computer programming, data analysis, and scientific research. This introduction will provide a detailed overview of

* number converters
* features
* functionality
* use cases.

NUMBER SYSTEMS

**DECIMAL NUMBER SYSTEM:**

\the numbers are represented with base 10. The way of denoting the decimal numbers with base 10 is also termed as decimal notation. It consists of 10 digits, such as, 0,1,2,3,4,5,6,7,8,9. Each digit in the decimal system has a position and every digit is ten times more significant than the previous digit. Suppose, 25 is a decimal number, then 2 is ten times more than 5. Some examples of decimal numbers are:-

(12)10, (345)10

**BINARY NUMBER SYSTEM:**

According to digital electronics and mathematics, a binary number is defined as a number that is expressed in the binary system or base 2 numeral system. It describes numeric values by two separate symbols; 1 (one) and 0 (zero). The base-2 system is the positional notation with 2 as a radix.

**HEXA DECIMAL NUMBER SYSTEM:**

The **hexadecimal number system** is a type of number system, that has a base value equal to 16. It is also pronounced sometimes as .’hex Hexadecimal numbers are represented by only 16 symbols. These symbols or values are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Each digit represents a decimal value. For example, D is equal to base-10 13.

**OCTAL NUMBER SYSTEM:**

**Octal Number System**has a base of eight and uses the numbers from 0 to 7. The octal numbers, in the are usually represented by binary numbers when they are grouped in pairs of three. For example, an octal number 128 is expressed as 0010102in the binary system, where 1 is equivalent to 001 and 2 is equivalent to 010

**J FRAME:**

In Java, "JFrame" is a class in the Swing user interface toolkit that is used to create windows with decorations such as a title bar, maximize and minimize buttons, and borders.

JFrame is a subclass of the java.awt.Frame class and provides more advanced features for creating and managing windows. It allows you to add other

Swing components such as buttons, text fields, and menus to the window, and provides layout managers to arrange these components within the window.

**SCOPE:**

The scope of a number system converter is to provide a tool that can convert numbers from one numeral system to another. It can be used by students, professionals, and anyone who needs to work with different numeral systems, including decimal, binary, octal, hexadecimal, and others.

* A number system converter can be used for various purposes, such as:
* Converting numbers from one numeral system to another for academic or professional purposes.
* Understanding how different numeral systems work and their relationship with each other.
* Solving mathematical problems that involve different numeral systems.
* Overall, a number system converter can be a very useful tool for anyone who needs to work with numbers in different numeral systems.

import java.awt.\*;

import javax.swing.\*;

public class Converter extends JFrame {

    JLabel toLabel, fromLabel, inputLabel, outputLabel, titleLabel;

    JComboBox<String> to, from;

    JTextField input, output;

    JButton convert;

    ImageIcon icon = new ImageIcon("assets/convert-logo.png");

    Font labels, font, title, inputs;

    // defining constructor

    public Converter() {

        super("Converter");

        this.setVisible(true);

        this.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

        this.setSize(400, 300);

        this.setLayout(null);

        initialiseObjects();

    }

    public void initialiseObjects() {

        //setting up fonts

        font = new Font("Arial", Font.PLAIN, 20);

        labels = new Font("Arial", Font.PLAIN, 18);

        title = new Font("Arial", Font.PLAIN, 24);

        inputs = new Font("Arial", Font.PLAIN, 16);

        // setting up dropdown

        to = new JComboBox<>(new String[]{"Binary", "Octal", "Hexadecimal", "Decimal"});

        to.setFont(inputs);

        add(to);

        from = new JComboBox<String>(new String[]{"Binary", "Octal", "Hexadecimal", "Decimal"});

        from.setFont(inputs);

        add(from);

        // setting up input field

        input = new JTextField(20);

        input.setFont(inputs);

        add(input);

        // setting up output field

        output = new JTextField(20);

        output.setFont(inputs);

        output.setEditable(false);

        add(output);

        // setting up convert button

        convert = new JButton("Convert", icon);

        convert.setFont(inputs);

        convert.setBackground(Color.WHITE);

        add(convert);

        // setting up labels

        toLabel = new JLabel("Convert to:");

        toLabel.setFont(labels);

        add(toLabel);

        fromLabel = new JLabel("Convert from:");

        fromLabel.setFont(labels);

        add(fromLabel);

        inputLabel = new JLabel("Input:");

        inputLabel.setFont(labels);

        add(inputLabel);

        outputLabel = new JLabel("Output:");

        outputLabel.setFont(labels);

        add(outputLabel);

        titleLabel = new JLabel("Number Converter");

        titleLabel.setFont(title);

        add(titleLabel);

        // call makeGUI method

        makeGUI();

    }

* This Java code defines a class called "Converter" that extends the "JFrame" class from the Swing user interface toolkit. The class creates a graphical user interface (GUI) for a number system converter.
* The GUI includes a window with a title "Converter" and size of 400x300 pixels.
* The constructor for the "Converter" class sets up the basic window properties and calls a method called "initialiseObjects()" that sets up the components, such as the dropdown lists, text fields, button, and labels. It also sets up the fonts for each component.
* The "initialiseObjects()" method creates and adds the components to the window. The method sets the font, size, and layout of each component.

The "Converter" class defines several variables, such as "toLabel", "fromLabel", "inputLabel", "outputLabel", "titleLabel", "to", "from", "input", "output", and "convert". These variables represent the GUI components used in the window.

* Finally, the code uses an ImageIcon object to set the icon of the "Convert" button, and the "setVisible(true)" method to display the window on the screen.

public class BinaryOperations {

    public static String binaryToDecimal(String bin) throws Exception {

        return Integer.toString(Integer.parseInt(bin, 2));

    }

    public static String binaryToHex(String bin) throws Exception {

        return Integer.toHexString(Integer.parseInt(bin, 2)).toUpperCase();

    }

    public static String binaryToOctal(String bin) throws Exception {

        return Integer.toOctalString(Integer.parseInt(bin, 2));

    }

}

This code defines a Java class named `BinaryOperations` that contains three static methods: `binaryToDecimal()`, `binaryToHex()`, and `binaryToOctal()`. Each of these methods takes a `String` parameter called `bin`, which represents a binary number that the method will convert to a decimal, hexadecimal, or octal number, respectively.

The methods use the `parseInt()` method of the `Integer` class to convert the binary string to an integer representation of that number in the desired base (decimal, hexadecimal, or octal). Then, they use the `toString()` method of the `Integer` class to convert the integer value to a `String` representation in the desired base.

The `toUpperCase()` method is used in the `binaryToHex()` method to ensure that the hexadecimal output is in uppercase letters.

If the input `bin` parameter is not a valid binary string, the `parseInt()` method will throw a `NumberFormatException`,

public class DecimalOperations {

    public static String decimalToBinary(String dec) throws Exception {

        return Integer.toBinaryString(Integer.parseInt(dec));

    }

    public static String decimalToHex(String dec) throws Exception {

        return Integer.toHexString(Integer.parseInt(dec)).toUpperCase();

    }

    public static String decimalToOctal(String dec) throws Exception {

        return Integer.toOctalString(Integer.parseInt(dec));

    }

}

This code defines a Java class named `DecimalOperations` that contains three static methods: `decimalToBinary()`, `decimalToHex()`, and `decimalToOctal()`. Each of these methods takes a `String` parameter called `dec`, which represents a decimal number that the method will convert to a binary, hexadecimal, or octal number, respectively.

The methods use the `parseInt()` method of the `Integer` class to convert the decimal string to an integer representation of that number. Then, they use the `toBinaryString()`, `toHexString()`, or `toOctalString()` method of the `Integer` class to convert the integer value to a `String` representation in the desired base (binary, hexadecimal, or octal).

The `toUpperCase()` method is used in the `decimalToHex()` method to ensure that the hexadecimal output is in uppercase letters.

If the input `dec` parameter is not a valid decimal string, the `parseInt()` method will throw a `NumberFormatException`, which is why the method signature includes a `throws` clause to declare this possibility.

public class HexadecimalOperations {

    public static String hexaToDecimal(String hexa) throws Exception {

        return Integer.toString(Integer.decode("0X".concat(hexa)));

    }

    public static String hexaToBinary(String hexa) throws Exception {

        return Integer.toBinaryString(Integer.decode("0X".concat(hexa)));

    }

    public static String hexaToOctal(String hexa) throws Exception {

        return Integer.toOctalString(Integer.decode("0X".concat(hexa)));

    }

}

This code defines a Java class named `HexadecimalOperations` that contains three static methods: `hexaToDecimal()`, `hexaToBinary()`, and `hexaToOctal()`. Each of these methods takes a `String` parameter called `hexa`, which represents a hexadecimal number that the method will convert to a decimal, binary, or octal number, respectively.

The methods use the `decode()` method of the `Integer` class to convert the hexadecimal string to an integer representation of that number. The `decode()` method is used to handle the `0x` prefix that is often used to indicate a hexadecimal value. Therefore, the methods first concatenate the input `hexa` parameter with the `0X` prefix, and then pass the resulting string to the `decode()` method.

Then, they use the `toString()`, `toBinaryString()`, or `toOctalString()` method of the `Integer` class to convert the integer value to a `String` representation in the desired base (decimal, binary, or octal).

If the input `hexa` parameter is not a valid hexadecimal string, the `decode()` method will throw a `NumberFormatException`, which is why the method signature includes a `throws` clause to declare this possibility.

Note that the `decode()` method can also handle negative hexadecimal values by representing them in two's complement notation

public class OctalOperations {

    public static String octalToDecimal(String octal) {

        return Integer.toString(Integer.parseInt(octal, 8));

    }

    public static String octalToBinary(String octal) throws Exception {

        return Integer.toBinaryString(Integer.parseInt(octal, 8));

    }

    public static String octalToHexa(String octal) throws Exception {

        return Integer.toHexString(Integer.parseInt(octal, 8)).toUpperCase();

    }

}

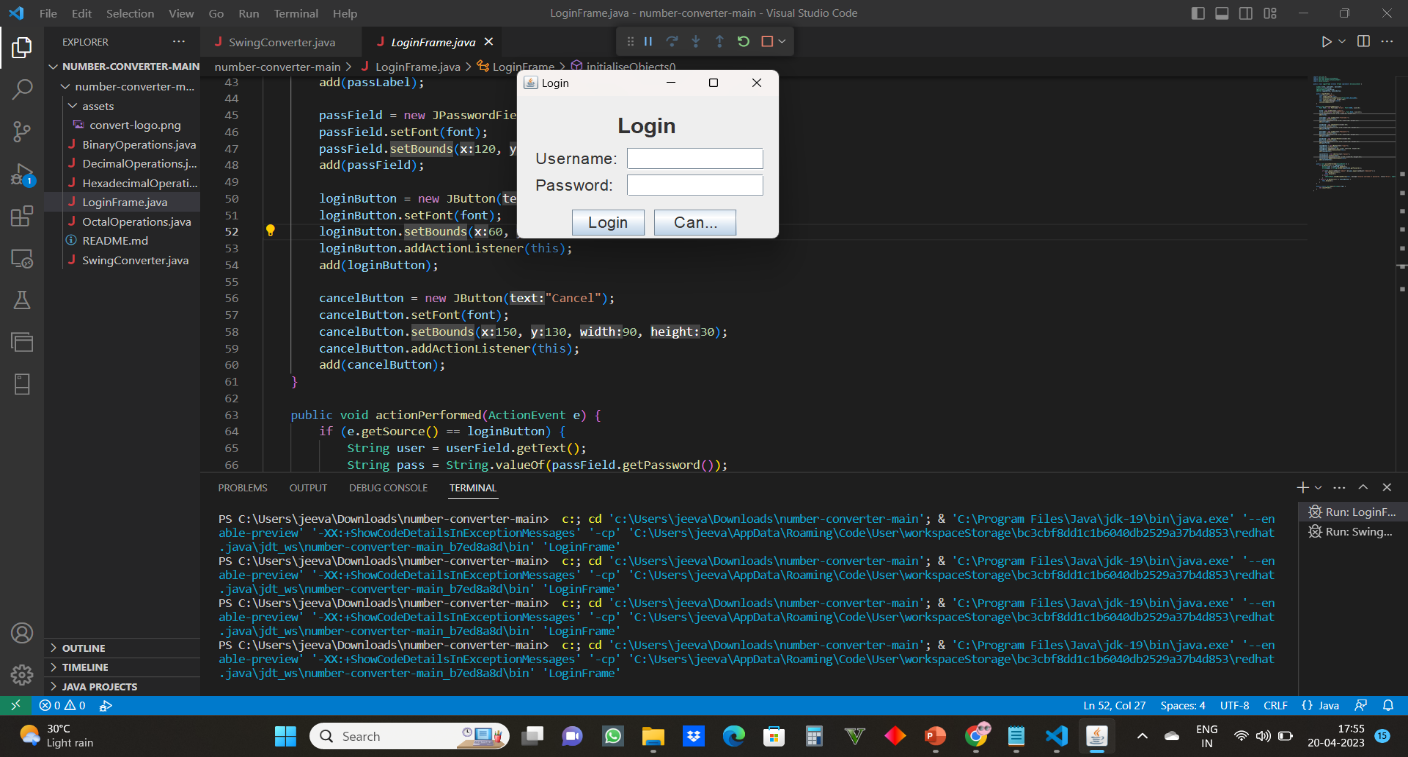
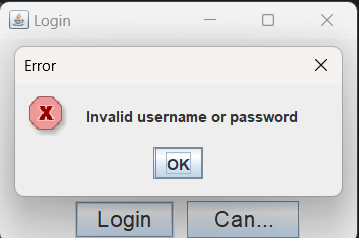
This code defines a Java class named `OctalOperations` that contains three static methods: `octalToDecimal()`, `octalToBinary()`, and `octalToHexa()`. Each of these methods takes a `String` parameter called `octal`, which represents an octal number that the method will convert to a decimal, binary, or hexadecimal number, respectively.

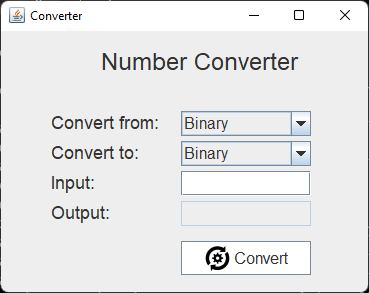
The methods use the `parseInt()` method of the `Integer` class to convert the octal string to an integer representation of that number in the desired base (decimal, binary, or hexadecimal). The second parameter of the `parseInt()` method specifies the base of the input string, which is set to 8 to indicate octal.

Then, they use the `toString()`, `toBinaryString()`, or `toHexString()` method of the `Integer` class to convert the integer value to a `String` representation in the desired base (decimal, binary, or hexadecimal).

The `toUpperCase()` method is used in the `octalToHexa()` method to ensure that the hexadecimal output is in uppercase letters.

If the input `octal` parameter is not a valid octal string, the `parseInt()` method will throw a `NumberFormatException`, which is why the method signature includes a `throws` clause to declare this possibility.



CONCLUSION:

* In conclusion, number converter systems are important tools that allow us to convert numbers from one base to another. The most commonly used number systems are the decimal system, which is used for everyday counting and calculations, and the binary system,
* IT is used in digital electronics and computing. The octal and hexadecimal systems are less commonly used, but they are still important in some computer systems and programming languages.

Converting numbers between different systems involves understanding the positional values of each digit in the given number and multiplying each digit by its corresponding positional value. With the help of number converter tools, this process can be done quickly and accurately, making it easier to work with numbers in different systems