GANAPA SATHISH

TITLE

white

KodNest

9701947117

sathishganapa7@gmail.com

Bangalore

TECHNICAL SKILLS

- Java SE
- Java EE
- Html
- CSS
- Java Script
- SQL

CERTIFICATIONS

- Certified Full Stack Developer from KodNest
- Certified in Developer Virtual Experience Program from Accenture

CAREER SKILLS

- Critical Thinking
- Problem Solving
- Teamwork & Collaboration
- Research & Analysis

AREA OF INTEREST

- REACT
- DEVOPS
- ML
- AI

CAREER OBJECTIVE

A creative and detailed individual has the capacity to develop an effective and efficient solution with no tolerance for errors and possess a positive attitude towards individual goals and an organizational goal.

EDUCATION DETAILS

Year	College/School	CGPA/Percentage
2023	B.Tech - SJCET	78%
2019	Intermediate-Narayana Junior College	98%
2017	SSC - ZPHS (W.C)	93%

INTERNSHIP

KodNest Technologies Pvt Ltd [Jan,2023 - May,2023]

Role : Full Stack Intern

- Trained in Java SE, Java EE, SQL, HTML, CSS, Java Script and Basics of Data Structures
- Executed Projects - Full Stack Web Application Shop Nest

PROJECT- 1

Shop Nest - E- Commerce Full Stack Web Application

Technologies Used: Oracle, Java SE, Java EE(Servlets & JSP's)

- The aim of an e-commerce website application is to facilitate online buying and selling of products or services.
- It includes features such as product catalog, shopping cart, payment gateway integration, order management, user accounts, and customer support. The main objectives of an e-commerce website application.

GANAPA SATHISH

TITLE

9701947117

sathishganapa7@gm

ail.comBangalore

LANGUAGES KNOWN

- English
- Telugu
- Kannada
- Hindi

HOBBIES

- Travelling
- Yoga

PROJECT - 2

E- Defence for People Safety:

- **Technologies used** - Java, XML, and Android Studio.
- This project, E-Defense for People Safety, proposes a system for detecting the problem and alerting the authorities about the situation using the most commonly available electronic devices like smartphones.

ACHIEVEMENTS

- Served as a Class Representative for four years in college
- Received a Medal in Intermediate for Securing Highest marks in Board Examination.

DECLARATION

DECLARATION

I solemnly declare that the information furnished above is free from errors to the best of my knowledge and belief.

Date:

Date:

Place:

Place:

Signature

Bitwise Operators

The Java Bitwise Operators allow access and modification of a particular bit inside a section of the data. It can be applied to integer types and bytes, and cannot be applied to float and double.

1.Bitwise AND(&):

The bitwise AND operator compares each bit of the first operand to corresponding bit of second operand. If both bits are 1, the result is 1, otherwise the result is 0.

2. Bitwise OR (|):

The bitwise OR operator compares each bit of the first operand to the corresponding bit of the second operand. If either of the bits is 1, the result is 1; otherwise the result is 0.

3.Bitwise XOR(^):

The bitwise XOR(exclusive OR) operator compares each bit if first operand to corresponding bit if second operand.
If the bits are different the result is 1, otherwise the result is 0.

4.Bitwise NOT(~):

The bitwise NOT operator inverts each bit of operand it flips 1s to 0s and 0s to 1s.

5.Left shift(<<):

The Left shift operator shifts the bits of the left operand to the left by specified number of positions.
The vacated bits are filled with zeros

6.Right shift(>>):

The Right shift operator shifts the bits of the left operand to the right by a specified number of positions.
The vacated bits are filled with sign bit(leftmost bit for signed integers)

7.Unsigned Right shift(>>>):

The Unsigned Right shift operator shifts the bits of the left operand to rightnby a specified number of positions.
The vacated bits are filled with zeros.

Example:

```
 class Bitwise {
 public static void main(String[] args) {
  int num1 = 30,  num2 = 6, num3 =0;

  //Bitwise AND
  System.out.println("num1 & num2 = " + (num1 & num2));

  //Bitwise OR
  System.out.println("num1 | num2 = " + (num1 | num2) );

  //Bitwise XOR
  System.out.println("num1 ^ num2 = " + (num1 ^ num2) );

  //Binary NOT
  System.out.println("~num1 = " + ~num1 );

  // Left Shift
  num3 = num1 <<  2;
  System.out.println("num1 << 1 = " + num3 );


 // Right Shift
  num3 = num1 >>  2;
  System.out.println("num1 >> 1  = " + num3 );

  //Unsigned Right shift
  num3 = num1 >>> 2;
  System.out.println("num1 >>> 1 = " + num3 );

 }
}
```

OUTPUT:

```
num1 & num2 = 6
num1 | num2 = 30
num1 ^ num2 = 24
~num1 = -31
num1 << 1 = 120
num1 >> 1  = 7
num1 >>> 1 = 7
```