KAKATIYA GOVERNMENT COLLEGE

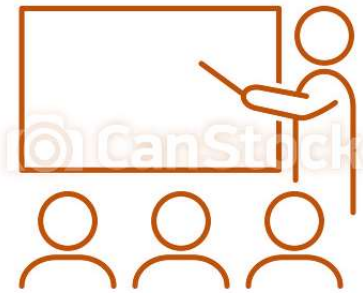
HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

DETAILS OF STUDENT SEMINARS OF I,III & V SEMESTER

FOR

THE YEAR 2021-22



DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Consolidate details of student seminar of I-III-V Semesters for the Academic year 2021-22

S.No	Date	Group	Student Name	Topic	Lecturer Name
1.	09-12-2021	BSC(MPCs) IIYr	E. Akhila	Stacks	M. Ramanakar
2.	10-12-2021	MStDs IIYr	V. Vinayak	HTML	M. Ramanakar
3.	20-12-2021	BSC(MPCs) IYr - ISem	K. Akhila	C - Tokens	T. Ragotham Reddy
4.	20-12-2021	BSC(MPCs) IYr - ISem	K. Pravalika	Area of Circle Program	T. Ragotham Reddy
5.	06-01-2022	BSC(MPCs) _ IIIYr – V Sem	M. Rohitha	Stacks	D. Rajkumar
6.	06-01-2022	B.Com CA IIYr - IIISem	SK. Sameer	SQL	K. Ramesh
7.	07-01-2022	BSC(MPCs) – IIIYr – V Sem	A. Sai Sumalya	Constructor	Dr. D. Suresh Babu
8.	11-02-2022	BSC(MPCs) – IIIYR –V Sem-SEC-B	V. Tharun	Inheritance in Java	V. Ramesh
9.	14-02-2022	B.Com CA IIYr - IIISem	K. Prudhviraj	SQL	K. Sravana Kumari
10.	21-02-2022	B.Com CAIIYr - IIISem	K. Abhinav Kalyan	Concurrency control in RDBMS	D. Praveen

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

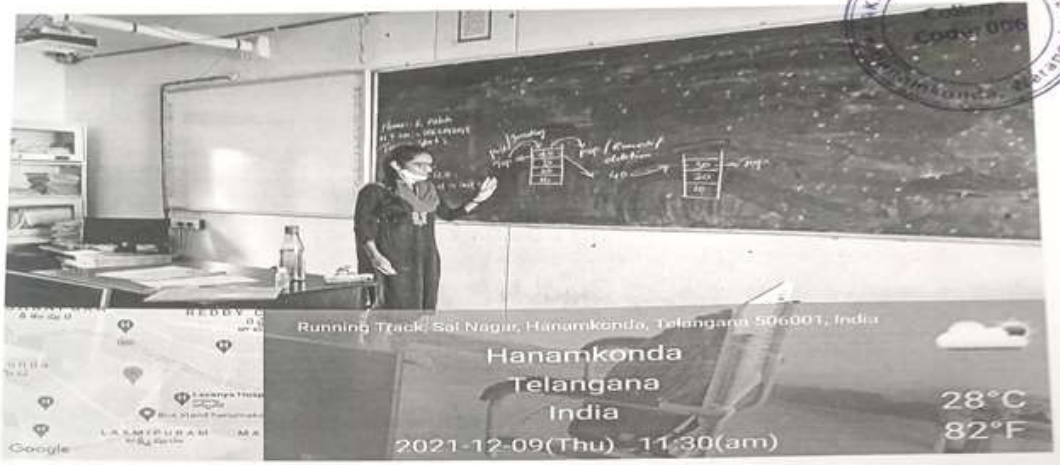
DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Student Seminar

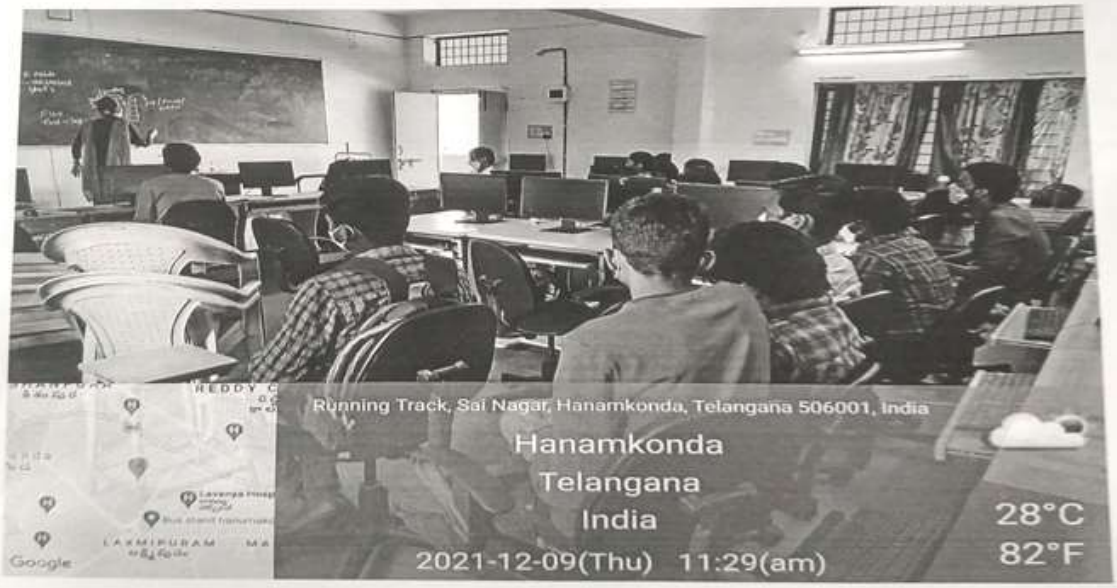
Student Name : E. Akhila | **Date : 09-12-2021**

Group & Year : BSC(MPCs) – IIYR – SEC-A | **Topic : STACK**

Subject : Data Structure Using C++ | **Faculty Involved: M. Ramanakar**



E Akhila, MPCS-II demonstrating operations on Stacks



STUDENT SEMINAR

Course : MPCS-II/A
Subject : Data Structures using C++
Topic : Stacks
Faculty Involved : M. Ramakar
HTNo : 006214248
Name of the Student : E. Akhila
Date : 9/12/2021
Signature of the Student : E. Akhila

KAKATIYA GOVT. COLLEGE Hanamkonda. College Code: 006

SNo	Hall Ticket No	Name of the Student	Signature
1	006214208	A. Saikireeti	
2	006214270	K. Cushma	
3	006214211	B. Manasa	Manasa
4	006214232	ch. Sindhuja	ch. Sindhuja
5	006214203	A. Sai Raj	A. Sai Raj
6	006214231	Ch. Venkateshwarlu	Ch. Venkateshwarlu
7	006214206	A. Srinivas	A. Srinivas
8	006214220	B. Anvesh	
9	006214245	A. Tharun	
10	006214226	B. Vinay kumar	B. Vinay kumar
11	006214253	E. Sandhya	E. Sandhya
12	006814274	K prathyusha	K. prathyu
13	006214216	B. Suretha	B. Suretha
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

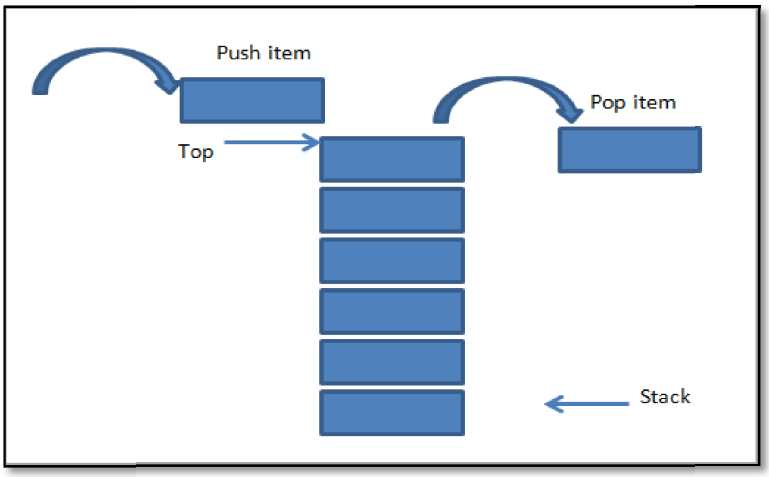
STACK

Stack is a fundamental data structure which is used to store elements in a linear fashion.
Stack follows **LIFO (last in, first out)** order or approach in which the operations are performed. This means that the element which was added last to the stack will be the first element to be removed from the stack.

Stack In C++

A stack is similar to real-life stack or a pile of things that we stack one above the other.

Given below is a pictorial representation of Stack.



As shown above, there is a pile of plates stacked on top of each other. If we want to add another item to it, then we add it at the top of the stack as shown in the above figure (left-hand side). This operation of adding an item to stack is called "**Push**".

On the right side, we have shown an opposite operation i.e. we remove an item from the stack. This is also done from the same end i.e. the top of the stack. This operation is called "**Pop**".

As shown in the above figure, we see that push and pop are carried out from the same end. This makes the stack to follow LIFO order. The position or end from which the items are pushed in or popped out to/from the stack is called the "**Top of the stack**".

Initially, when there are no items in the stack, the top of the stack is set to -1. When we add an item to the stack, the top of the stack is incremented by 1 indicating that the item is added. As opposed to this, the top of the stack is decremented by 1 when an item is popped out of the stack.

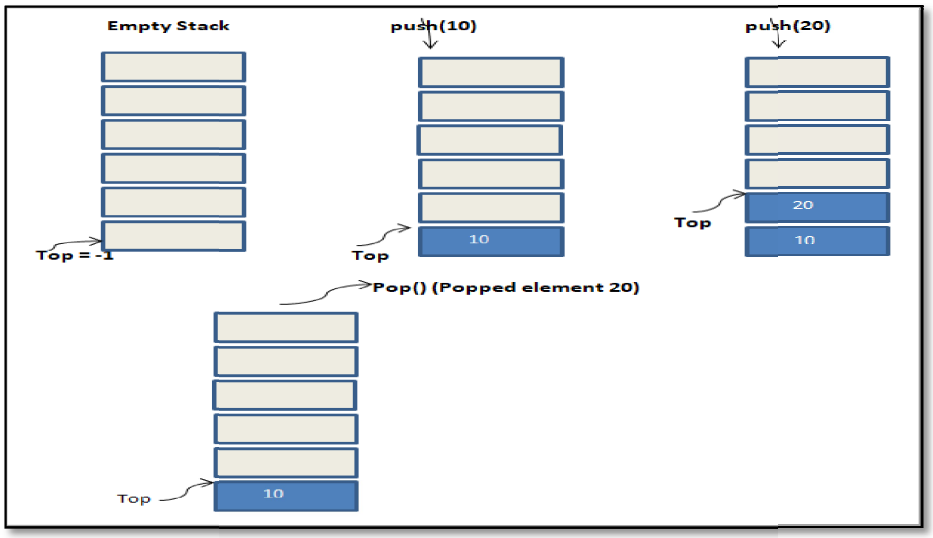
Next, we will see some of the basic operations of the stack data structure that we will require while implementing the stack.

Basic Operations

Following are the basic operations that are supported by the stack.

- **push** – Adds or pushes an element into the stack.
- **pop** – Removes or pops an element out of the stack.
- **peek** – Gets the top element of the stack but doesn't remove it.
- **isFull** – Tests if the stack is full.
- **isEmpty** – Tests if the stack is empty.

Illustration



The above illustration shows the sequence of operations that are performed on the stack. Initially, the stack is empty. For an empty stack, the top of the stack is set to -1.

Next, we push the element 10 into the stack. We see that the top of the stack now points to element 10.

Next, we perform another push operation with element 20, as a result of which the top of the stack now points to 20. This state is the third figure.

Now in the last figure, we perform a pop () operation. As a result of the pop operation, the element pointed at the top of the stack is removed from the stack. Hence in the figure, we see that element 20 is removed from the stack. Thus the top of the stack now points to 10.

In this way, we can easily make out the LIFO approach used by stack.

Using Arrays

Following is the C++ implementation of stack using arrays:

```
#include<iostream>
using namespace std;
  #define MAX 1000 //max size for stack
  class Stack
{
   int top;
   public:
   int myStack[MAX]; //stack array

   Stack() { top = -1; }
   bool push(int x);
   int pop();
   bool isEmpty();
};

   //pushes element on to the stack
```

```
  bool Stack::push(int item)
  {
    if (top >= (MAX-1)) {
    cout << "Stack Overflow!!!";
    return false;
  }
else {
  myStack[++top] = item;
  cout<<item<<endl;
  return true;
  }
}

//removes or pops elements out of the stack
int Stack::pop()
{
  if (top < 0) {
    cout << "Stack Underflow!!";
    return 0;
  }
else {
     int item = myStack[top--];
    return item;
  }
}

//check if stack is empty
bool Stack::isEmpty()
{
  return (top < 0);
}

// main program to demonstrate stack functions
int main()
{
  class Stack stack;
  cout<<"The Stack Push "<<endl;
  stack.push(2);
  stack.push(4);
  stack.push(6);
  cout<<"The Stack Pop : "<<endl;
  while(!stack.isEmpty())
    {
    cout<<stack.pop()<<endl;
    }
```

```
    return 0;
}
```

Output:

The Stack Push
2
4
6
The Stack Pop:
6
4
2

In the output, we can see that the elements are pushed into the stack in one order and are popped out of the stack in the reverse order. This exhibits the LIFO (Last in, First out) approach for the stack.

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

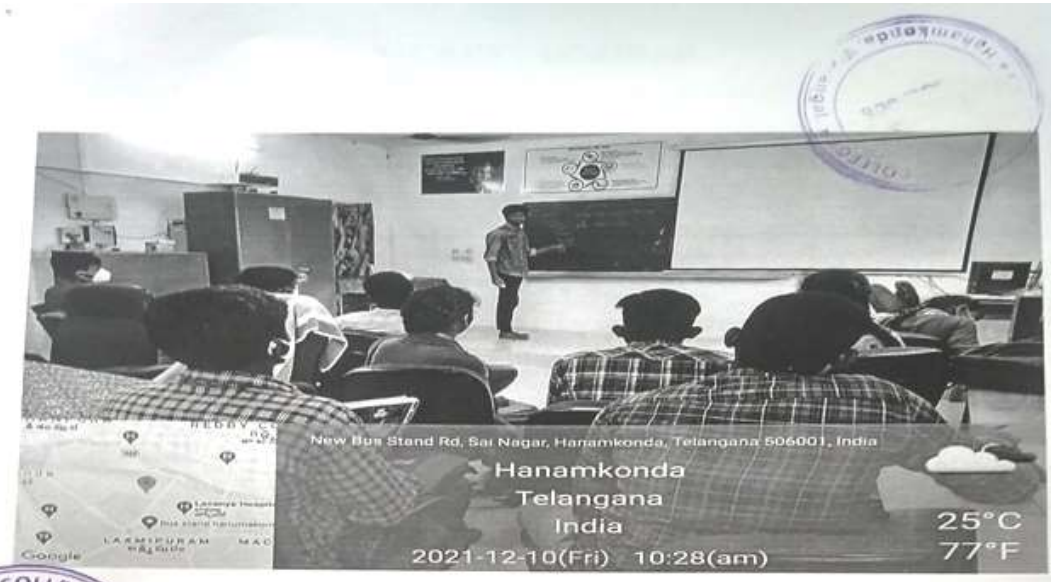
(Affiliated to Kakatiya University)

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

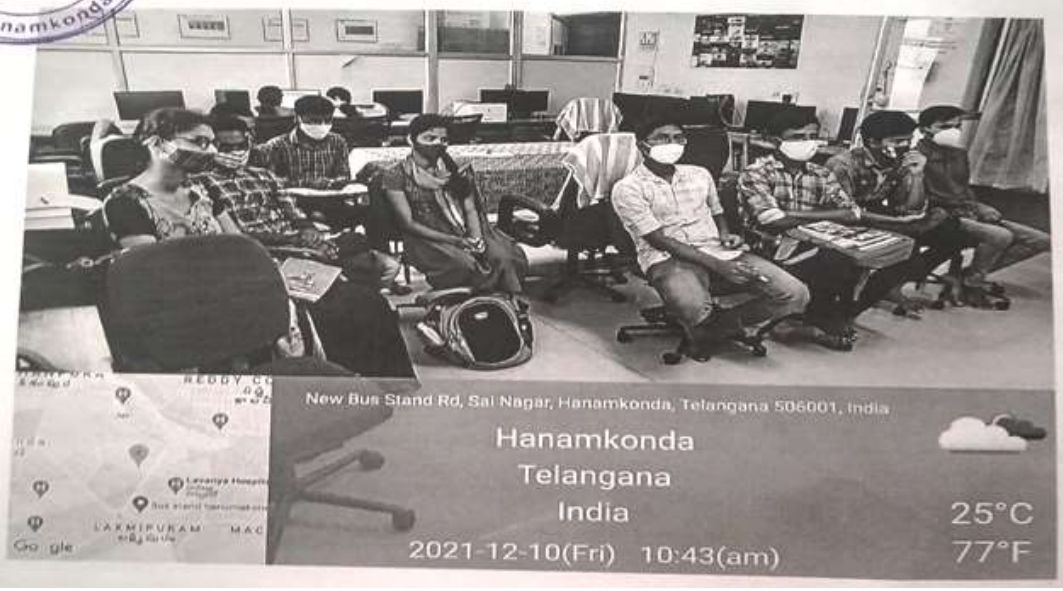
Student Seminar

Student Name : V. Vinayak
Group & Year : BSC(MStCs) – IIYR
Subject : Data Engineering

Date : 10-12-2021
Topic : HTML
Faculty Involved: M. Ramanakar



V. Vinayak, MStDS-II demonstrating HTML elements



STUDENT SEMINAR

Course : MSc-DS - II
Subject : Data Engineering
Topic : HTML
Faculty Involved : M. Ramanaker
HTNo : 006214628
Name of the Student : V. Vinayak
Date : 10/12/2021
Signature of the Student : V. Vinayak

SNo	Hall Ticket No	Name of the Student	Signature
1	006214616	K. Jaya Krishna	K. Jayakrishna
2	006214614	Keerthan. M	Keerthan M
3	006214618	L. Prem Kumar	[illegible]
4	006214629	V. Chandana	V. Chandana
5	006214607	G. Shanthi Sharmila	G. Shanthi Sharmila
6	006214606	G. Manasa	G. Manasa
7	006214631	MD. Yakub	[illegible]
8	006214622	P. Ramu	P. Ramu
9	006214603	B. Nagaraju	B. Nagaraju
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

H T M L

HTML is an acronym which stands for **Hyper Text Markup Language** which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. **With the help of HTML only, we can create static web pages**.

Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

Let's see a simple example of HTML.

```
1. <!DOCTYPE>
2. <html>
3. <head>
4. <title>Web page title</title>
5. </head>
6. <body>
7. <h1>Write Your First Heading</h1>
8. <p>Write Your First Paragraph.</p>
9. </body>
10. </html>
```


Description of HTML Example

<!DOCTYPE>: It defines the document type or it instruct the browser about the version of HTML.

<html > :This tag informs the browser that it is an HTML document. Text between html tag describes the web document. It is a container for all other elements of HTML except <!DOCTYPE>

<head>: It should be the first element inside the <html> element, which contains the metadata(information about the document). It must be closed before the body tag opens.

<title>: As its name suggested, it is used to add title of that HTML page which appears at the top of the browser window. It must be placed inside the head tag and should close immediately. (Optional)

<body> : Text between body tag describes the body content of the page that is visible to the end user. This tag contains the main content of the HTML document.

<h1> : Text between <h1> tag describes the first level heading of the webpage.

<p> : Text between <p> tag describes the paragraph of the webpage.

Features of HTML

1) It is a very **easy and simple language**. It can be easily understood and modified.

2) It is very easy to make an **effective presentation** with HTML because it has a lot of formatting tags.

3) It is a **markup language**, so it provides a flexible way to design web pages along with the text.

4) It facilitates programmers to add a **link** on the web pages (by html anchor tag), so it enhances the interest of browsing of the user.

5) It is **platform-independent** because it can be displayed on any platform like Windows, Linux, and Macintosh, etc.

6) It facilitates the programmer to add **Graphics, Videos, and Sound** to the web pages which makes it more attractive and interactive.

7) HTML is a case-insensitive language, which means we can use tags either in lower-case or upper-case.

Building blocks of HTML

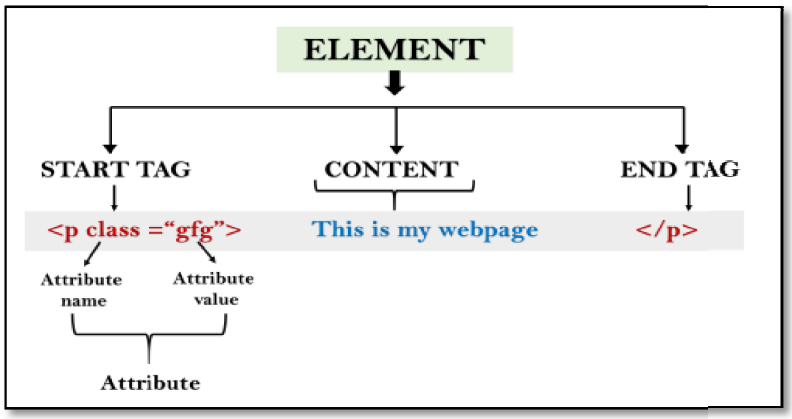
An HTML document consist of its basic building blocks which are:

- **Tags:** An HTML tag surrounds the content and apply meaning to it. It is written between < and > brackets.
- **Attribute:** An attribute in HTML provides extra information about the element, and it is applied within the start tag. An HTML attribute contains two fields: name & value.

  Syntax

  <tag name attribute_name= " attr_value"> content </ tag name>

- **Elements:** An HTML element is an individual component of an HTML file. In an HTML file, everything written within tags are termed as HTML elements.



Example:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title>The basic building blocks of HTML</title>
5.   </head>
6.   <body>
7.       <h2>The building blocks</h2>
8.       <p>This is a paragraph tag</p>
9.       <p style="color: red">The style is attribute of paragraph tag</p>
10.      <span>The element contains tag, attribute and content</span>
11.  </body>
12. </html>
```

Output:

The building blocks

This is a paragraph tag

The style is attribute of paragraph tag

The element contains tag, attribute and content

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Student Seminar

Student Name : K Akhila

Date: 20-12-2021

Group & Year : B.Sc (MPCs) -I/II

Topic : C Tokens

K. Akhila
Student Signature:

Signature of the Lecturer

H/c. Department of Computer Science
Kakatiya Govt. College,
HANAMKONDA.

Attendance of Participation Students Details

KAKATIYA GOVT COLLEGE, Hanmakonda, Warangal

S.NO.	H.T.NO.	Name of the student	sign.
1)	0062224246	Godumala Sathwika	G. Sathwika
2)	0062224213	Bangari. Nithin raaj	B. Nithin raaj
3)	0062224236	Egurla. Ashwini	Ashwini
3)	0062224226	chinthala Prakash	
4)	0062224270	K. shiny Kanishka	K. Kanishka
5)	0062224216	B. Raju kumar	
6)	0062224222	Ch. Aravind	
7)	0062224228	CH. Rohith	CH.
8)	0062224266	k. Rahul	
9)	0062224233	D. Sai kumar	
10)	0062224231	Dasari Supriya	Supriya.
11)	0062224250	G. Adarsh	
12)	0062224243	S. Ganesh	S. ganesh
13)	0062224242	G. Ravali	G. Ravali
14)	0062224251	G. Mamatha	G. Mamatha
15)	0062224252	G. Preethi	G. preethi
16)	0062224219	B. Sriharshitha	B. Sriharshitha

Tokens in C

Tokens are the smallest elements of a program, which are meaningful to the compiler.

The following are the types of tokens: Keywords, Identifiers, Constant, Strings, Operators, etc.

Keywords

Keywords are predefined, reserved words in C and each of which is associated with specific features. These words help us to use the functionality of C language. They have special meaning to the compilers.

There are total 32 keywords in C.

Identifiers

Each program element in C programming is known as an identifier. They are used for naming of variables, functions, array etc. These are user-defined names which consist of alphabets, number, underscore '_'. Identifier's name should not be same or same as keywords. Keywords are not used as identifiers.

Rules for naming C identifiers –

- It must begin with alphabets or underscore.
- Only alphabets, numbers, underscore can be used, no other special characters, punctuations are allowed.
- It must not contain white-space.
- It should not be a keyword.
- It should be up to 31 characters long.

Strings

A string is an array of characters ended with a null character(\0). This null character indicates that string has ended. Strings are always enclosed with double quotes(" ").

Let us see how to declare String in C language –

- char string[20] = {'s','t','u','d','y', '\0'};
- char string[20] = "demo";
- char string [] = "demo";

Special characters

- Special characters in 'C' are shown in the given table,

Special Character	Description
, (comma)	{ (opening curly bracket)
. (period)	} (closing curly bracket)
; (semi-colon)	[(left bracket)
: (colon)	] (right bracket)
? (question mark)	((opening left parenthesis) etc.

Constants in C

A constant is a value or variable that can't be changed in the program, for example: 10, 20, 'a', 3.4, "c programming" etc.

2 ways to define constant in C

There are two ways to define constant in C programming.

1. const keyword
2. #define preprocessor

1) C const keyword

The const keyword is used to define constant in C programming.

```
1. const float PI=3.14;
```

Now, the value of PI variable can't be changed.

```
1. #include<stdio.h>
2. int main(){
3.     const float PI=3.14;
4.     printf("The value of PI is: %f",PI);
5.     return 0;
6. }
```

Output:

```
The value of PI is: 3.140000
```

If you try to change the the value of PI, it will render compile time error.

```
1. #include<stdio.h>
2. int main(){
3. const float PI=3.14;
4. PI=4.5;
5. printf("The value of PI is: %f",PI);
6.     return 0;
7. }
```

Output: Compile Time Error: Cannot modify a const object

2) C #define preprocessor: The #define preprocessor is also used to define constant. We will learn about #define preprocessor directive later.

C #define: The #define preprocessor directive is used to define constant or micro substitution. It can use any basic data type.

1. Syntax: #define token value

   Ex: #define PI 3.14

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

College Code: 006

Hanamkonda, Warangal

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Student Seminar

Date: 20-12-2021

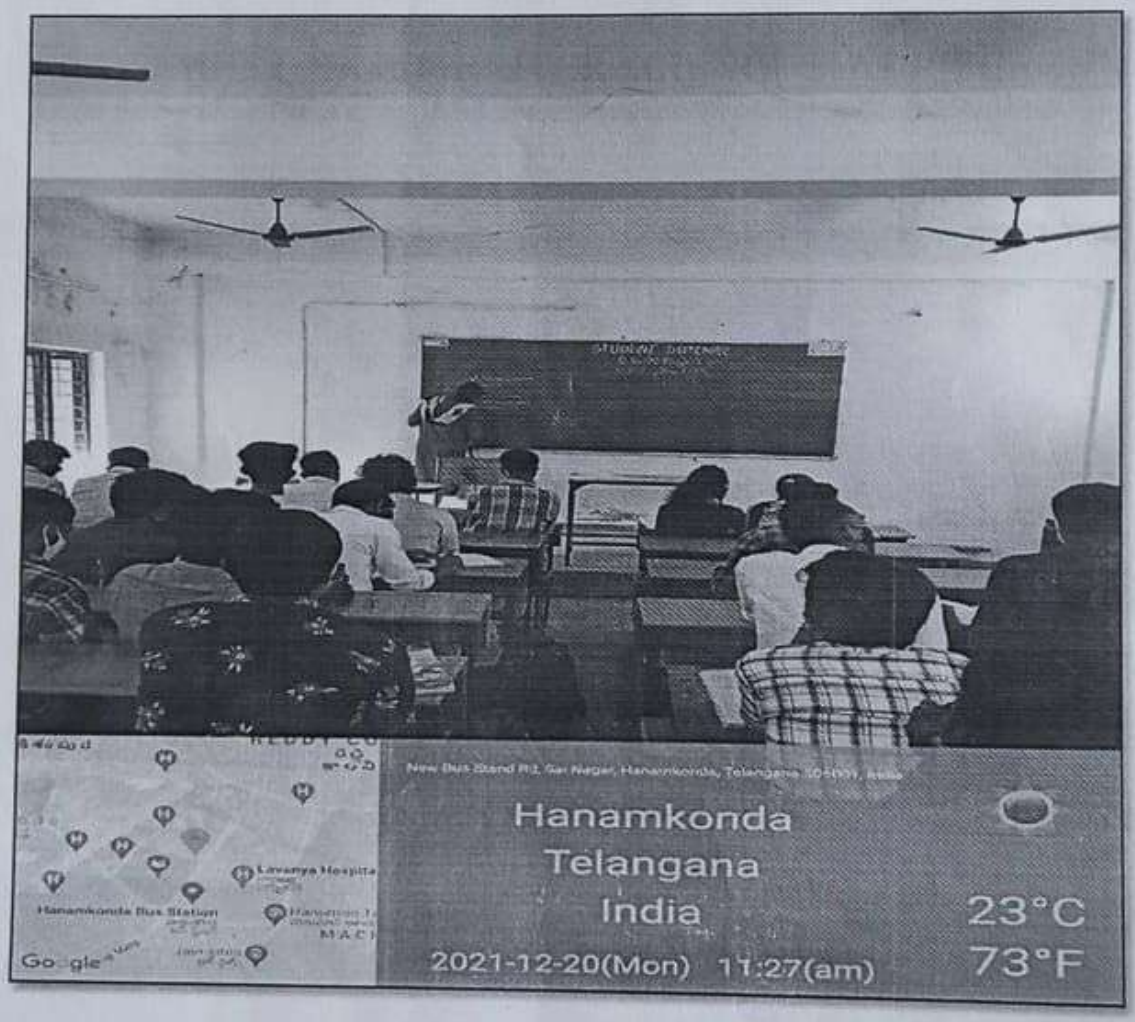
Student Name : Pravalika

Group & Year : B.Sc(MPCs)-I/IIsem

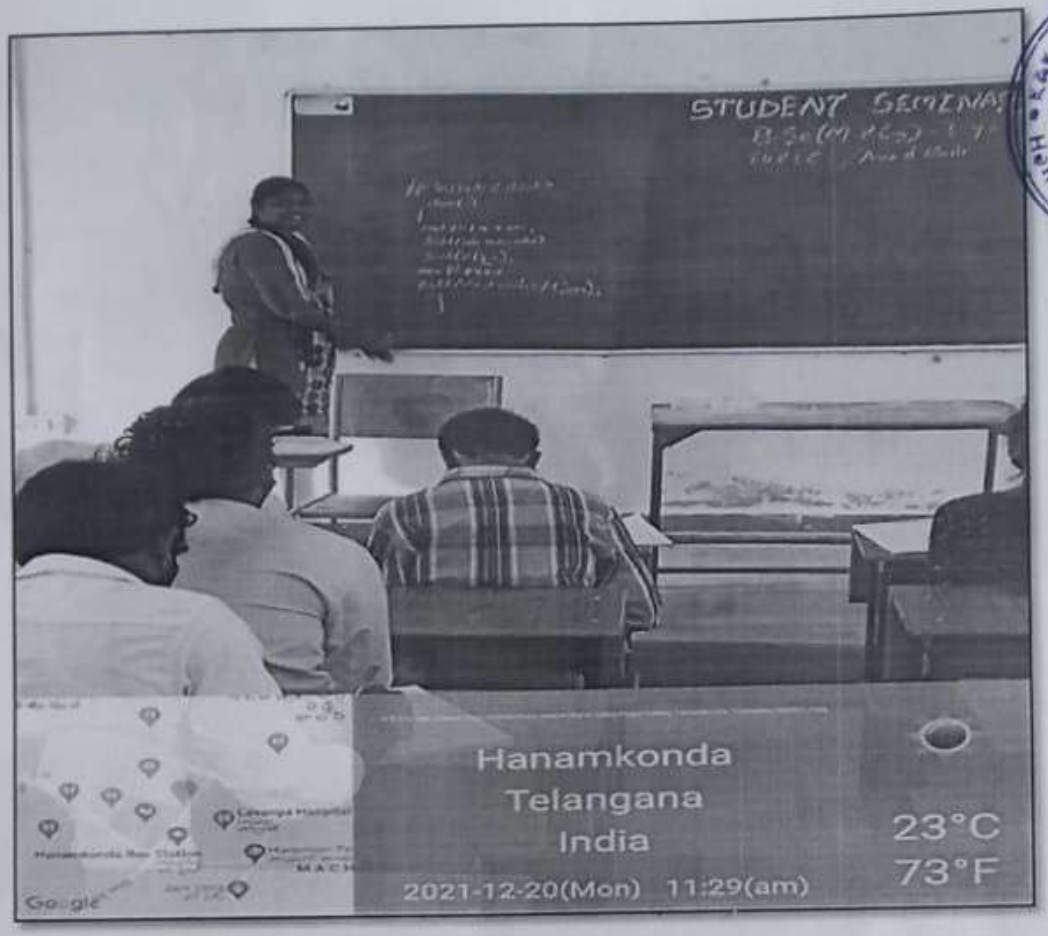
Topic : Area of circle program

Student Signature: Pravalika

Signature of the Lecturer



I/c. Department of Computer Science
Kakatiya Govt. College.
HANAMKONDA



KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal.

Attendance of Participation Students Details

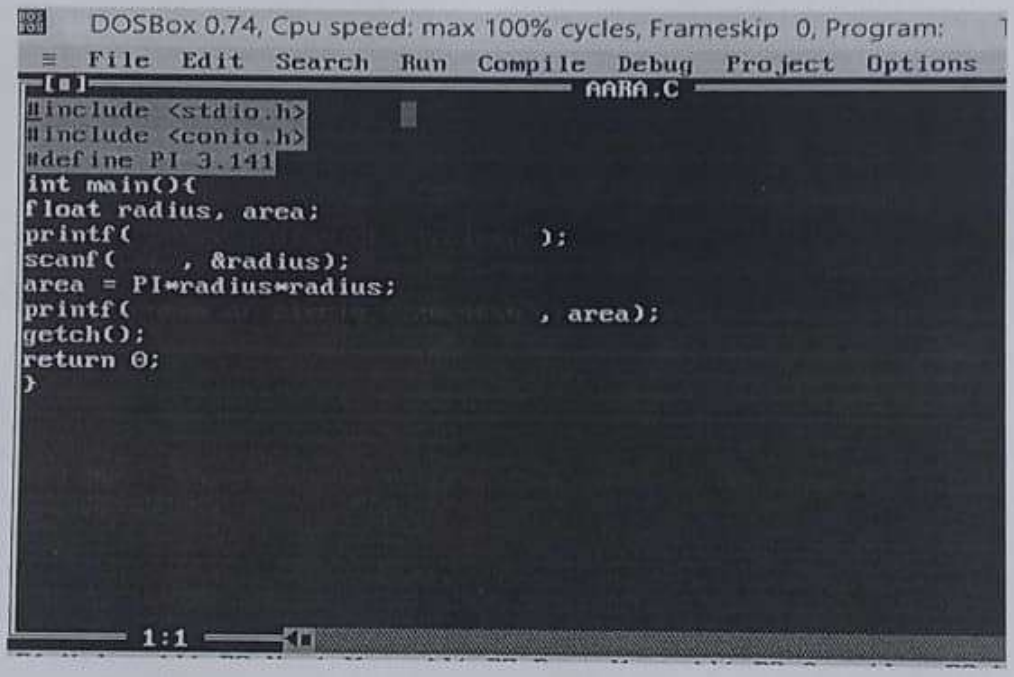
H.T.NO.	Name of the student	sign.
006224259	Kadiyam. Akhila	K. Akhila
006224231	Dasari Supriya	Supriya.
006224233	Dommata Saikumar	Saikumar
006224270	K. Shine Knight	K. Shine
006224216	B. Rajakumar	Raj.
006224222	Ch. Aravind	[illegible]
006224266	K. Rahul	[illegible]
006224228	CH. Rohith	CH [illegible]
006224236	E. Ashwini	Ashwini
006224226	ch. Prakash	[illegible]
006224213	B. Nithin vaas	B. Nithin vaas.

@@Area of Circle program Using C Language

```
1. #include < stdio.h >
2. #include < conio.h > #define PI 3.141
3. int main()
4. {
5.     float radius, area;
6.     printf("Enter radius of circle\n");
7.     scanf("%f", & radius);
8.     area = PI * radius * radius;
9.     printf("Area of circle : %0.4f\n", area);
10.    getch();
11.    return 0;
12.}
```

Explanation

- Through programming, finding an area of a circle is clearly understood.



Output

```
C:\TURBOC3\BIN>TC
Enter radius of circle
15
Area of circle : 706.7250
```

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Student Seminar

Student Name : M. Rohitha **Date: 06-01-2022**

Group & Year : BSC(MPCs) – IIYr

Subject : Data Structure Using C++ **Topic : Stack**

Faculty Involved : D. Rajkumar

KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal.

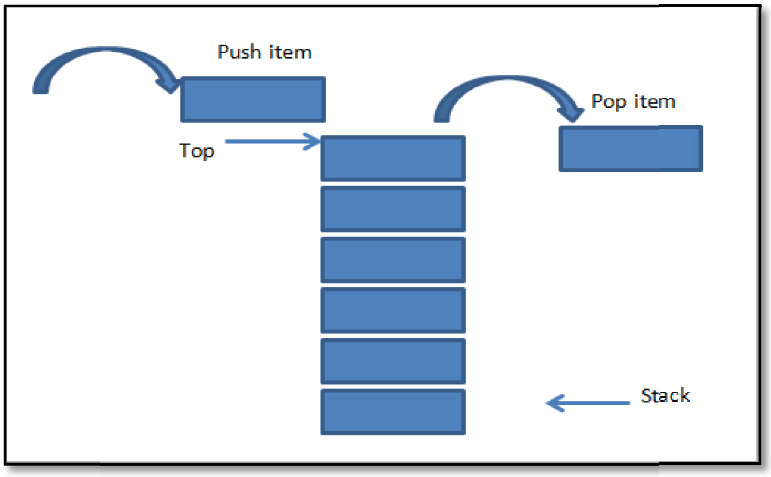
S T A C K

Stack is a fundamental data structure which is used to store elements in a linear fashion.
Stack follows **LIFO (last in, first out)** order or approach in which the operations are performed. This means that the element which was added last to the stack will be the first element to be removed from the stack.

Stack In C++

A stack is similar to real-life stack or a pile of things that we stack one above the other.

Given below is a pictorial representation of Stack.



As shown above, there is a pile of plates stacked on top of each other. If we want to add another item to it, then w add it at the top of the stack as shown in the above figure (left-hand side). This operation of adding an item t stack is called "**Push**".

On the right side, we have shown an opposite operation i.e. we remove an item from the stack. This is also don from the same end i.e. the top of the stack. This operation is called "**Pop**".

As shown in the above figure, we see that push and pop are carried out from the same end. This makes the stack t follow LIFO order. The position or end from which the items are pushed in or popped out to/from the stack called the "**Top of the stack**".

Initially, when there are no items in the stack, the top of the stack is set to -1. When we add an item to the stacl the top of the stack is incremented by 1 indicating that the item is added. As opposed to this, the top of the stack decremented by 1 when an item is popped out of the stack.

Next, we will see some of the basic operations of the stack data structure that we will require while implementin the stack.

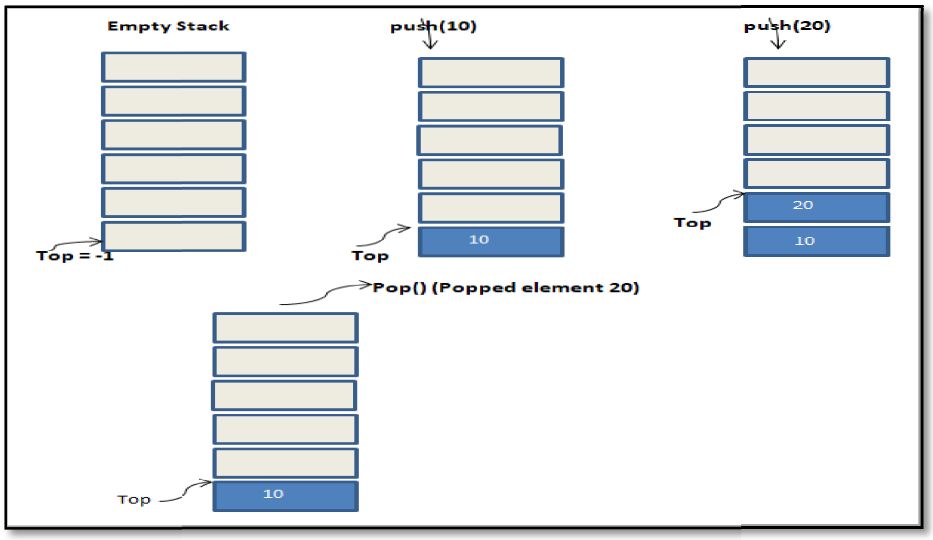
Basic Operations

Following are the basic operations that are supported by the stack.

- **push** – Adds or pushes an element into the stack.
- **pop** – Removes or pops an element out of the stack.
- **peek** – Gets the top element of the stack but doesn't remove it.

- **isFull** – Tests if the stack is full.
- **isEmpty** – Tests if the stack is empty.

Illustration



The above illustration shows the sequence of operations that are performed on the stack. Initially, the stack is empty. For an empty stack, the top of the stack is set to -1.

Next, we push the element 10 into the stack. We see that the top of the stack now points to element 10.

Next, we perform another push operation with element 20, as a result of which the top of the stack now points to 20. This state is the third figure.

Now in the last figure, we perform a pop () operation. As a result of the pop operation, the element pointed at the top of the stack is removed from the stack. Hence in the figure, we see that element 20 is removed from the stack. Thus the top of the stack now points to 10.

In this way, we can easily make out the LIFO approach used by stack.

Using Arrays

Following is the C++ implementation of stack using arrays:

```
#include<iostream>
using namespace std;
 #define MAX 1000 //max size for stack
 class Stack
{
  int top;
  public:
  int myStack[MAX]; //stack array

  Stack() { top = -1; }
```

```
  bool push(int x);
  int pop();
  bool isEmpty();
};

  //pushes element on to the stack
  bool Stack::push(int item)
  {
    if (top >= (MAX-1)) {
    cout << "Stack Overflow!!!";
    return false;
  }
else {
  myStack[++top] = item;
  cout<<item<<endl;
  return true;
  }
}

//removes or pops elements out of the stack
int Stack::pop()
{
  if (top < 0) {
    cout << "Stack Underflow!!";
    return 0;
  }
else {
     int item = myStack[top--];
    return item;
  }
}

//check if stack is empty
bool Stack::isEmpty()
{
  return (top < 0);
}

// main program to demonstrate stack functions
int main()
{
  class Stack stack;
```

```
  cout<<"The Stack Push "<<endl;
  stack.push(2);
  stack.push(4);
  stack.push(6);
  cout<<"The Stack Pop : "<<endl;
  while(!stack.isEmpty())
    {
    cout<<stack.pop()<<endl;
    }
  return 0;
}
```

Output:

The Stack Push
2
4
6
The Stack Pop:
6
4
2

In the output, we can see that the elements are pushed into the stack in one order and are popped out of the stack i the reverse order. This exhibits the LIFO (Last in, First out) approach for the stack.

STUDENT SEMINAR

Course : B.Sc (MPCs)-II

Topic : Stack

HTNo : 0062142-91

Date : 06-01-2022

Subject : Data Structures Using C++

Faculty Involved : D Rajkumar

Name of the Student : M. Rohitha

Signature of the Student : Rohitha.

KAKATIYA GOVT. COLLEGE Hanamkonda Warangal College Code: 006

SNo	Hall Ticket No	Name of the Student	Signature
1	006214299	M. Aarthika	M. Aarthika
2	0062142777	K. Kavyasri	K. Kavyasri
3	006214296	M. Madhavi	M. Madhavi
4	0062143348	T. Saivamshi	T. Saivamshi
5	006214305	MD. Akbar pasha	[illegible]
6	006214302	M. Pranay	M. Pranay
7	006214275	K. Premender Reddy	[illegible]
8	006214289	M. Hemanth	[illegible]
9	006214298	M. Akshaykumar	[illegible]
10	006214310	M. Charan Kumar	[illegible]
11	006214315	N. Akash	[illegible]
12	006214324	P. Gangotri	P. Gangotri
13	006214338	S. Supraja	S. Supraja
14	006214345	T. Ramesh	T. Ramesh
15	006214321	P. Anvesh	P. Anvesh
16	006214320	O. Varaprasad	[illegible]
17	006214339	S. Prathyusha	[illegible]
18	006214346	T. Sravani	Sravani
19	006214342	S. Priyanka	[illegible]
20			
21			
22			
23			
24			
25			

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Student Seminar

Student Name : Sk. Sameer

Date: 06-01-2022

Group & Year : BCom (CA) III Sem

Topic : SQL

Student Signature:

Signature of the Lecturer

H.O.D. Department of Computer Science
Kakatiya Govt. College,
HANAMKONDA.

H/c. Department of Computer Science
Kakatiya Govt. College,
HANAMKONDA.

STUDENT SEMINAR

Course : BCom (CA) III Sem

Subject : RDBMS

Topic : SQL

Faculty Involved : K. Ramesh

HTNo : 006-21-2323

Name of the Student : SK. Sameer

Date : 06-01-2022

Signature of the Student :

KAKATIYA GOVT. COLLEGE Hanamkonda, Warangal. College Code: 006

SNo	Hall Ticket No	Name of the Student	Signature
1	006212345	V. Vigneshwar.	Vignesh
2	006212405	G. Anitha	Anit
3	006212404	Ch. Akshithaa	Akshithaa
4	006212403	Ch. Mohini	Mohini
5	006212279	P. Mukesh	Mukesh
6	006212289	P. Neha	Neha.
7	006212311	Sk. Azmat.	Azmat
8	006212407	S. Vindhya.	Vindhya.
9	006212398	G. Saipriya.	Saipriya
10	006212397	Ch. Srishanthi	Srei
11	006212399	K. Vyshnavi	Vyshnavi
12	006212401	E. Nandini	Nandini
13	006212284	P. Akhil	
14	006212338	T. Vamshi Krishna	
15	006212357	V. Rajesh	Rajesh
16	006212344	V. Sai Kumar	V. Sai kumar
17	006212272	P. Ravi kumar	Ravi kumar
18	006212381	G. Ajay	G. Ajay
19	006212336	T. Prasad	T. Prasad
20			
21			
22			
23			
24			
25			

KAKATIYA GOVT. COLLEGE Hanamkonda, Warangal. College Code: 006

Structured Query Language.

SQL Commands

SQL Commands can be classified into 4 types

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)
4. Transaction Control Language (T.CL)

Data Definition Language (OR) DDL Commands

DDL Commands are used to define the structure of a database. DDL deals with metadata. The Commands are: CREATE, ALTER, TRUNCATE, DROP and RENAME.

1) **CREATE COMMAND**: It is used to create database objects such as tables, views etc.

Syntax: CREATE TABLE <table-name>
(column_name_1 data type(size),
column name_2 data type(size),
:
Column_name_n data type (size)
);

Example : To create a table STUDENT with rno, name, marks attributes

⇒ CREATE TABLE STUDENT (rno Number(2), name varchar2 (20), marks number (3));

Now the table is created, to see the structure of the table, use DESC Command

⇒ DESC STUDENT;

Name	Null?	Type.
rno		Number (2)
name		varchar2 (20)
marks		Number (3)

2) ALTER Command : It is used to change the structure of a database object

Syntax : ALTER TABLE < table-name >
[ADD column_name data type (size)]
[MODIFY Column_name data type (size)]
[DROP COLUMN Column_name]
[RENAME COLUMN old-col-name to new-col-name]

Example : 1) To add a new column to an existing table:

SQL > ALTER TABLE STUDENT ADD (average Number (3,2));

2) To modify the size of an existing column of a table:

SQL> ALTER TABLE STUDENT MODIFY rno Number(3);

3, To delete an existing Column of a table;

SQL> ALTER TABLE STUDENT DROP COLUMN average;

4, To Rename an existing Column of a table:

SQL> ALTER TABLE STUDENT RENAME COLUMN

rno To rollno;

3) DROP Command : It is used to delete a database object.

Syntax : DROP object-type object-name;

Example : 1, To delete a table : DROP TABLE STUDENT

2, To delete an index : DROP INDEX dname-index

3, To delete view : DROP VIEW emp-view;

4, TRUNCATE Command : It is used to delete rows (not the table's structure) with auto Commit.

Syntax : TRUNCATE TABLE <table-name>;

Example : To delete rows from a table,

SQL> TRUNCATE TABLE STUDENT;

5, RENAME Command : It is used to rename the database object.

Example : To rename a table,

SQL> RENAME STUDENT TO STD;

Data manipulation language (OR) DML Commands

DML Commands are used to maintain and access a database, including updating, inserting, modifying and querying data. It deals with data. The Commands are : INSERT, UPDATE, DELETE and SELECT.

1. **INSERT Command** : It is used to store a new record into a database table.

Syntax :

INSERT INTO <table name> [(column1, column2, ... column)] values (value1, value2, ... valuen);

(OR)

INSERT INTO <table name> [column-list] VALUES (value-list)

Example : 1, Take an EMPLOYEE table with the columns eno, ename, job, sal, hiredate.

To insert a record into EMPLOYEE table:

```
SQL> insert into EMPLOYEE (eno, ename, job, sal,
     hiredate) values (101, 'vasu', 'clerk', 7000,
     12-jan-2012');
```

```
& sal);
SQL> insert into EMPLOYEE values (&eno, '&ename
     '&job', &sal);
```

```
Enter value for eno : 104
Enter value for ename : Anji
Enter value for job : Asst manager
Enter value for sal : 15000
```

2. Update Command : It is used to edit/ change the values of attributes in a table.

Syntax : UPDATE <table-name> set Column_name = value [, column_name = value, ...]
[WHERE Condition];

Example : To update (set) salary of 'Ravi' to 8000

SQL> update EMPLOYEE set sal = 8000 WHERE ename = 'Ravi';

3 DELETE Command : It is used to remove (delete) one or more rows from a table

Syntax : DELETE FROM < table-name> [WHERE Condit

Example : To delete the record of employee whose name is 'Ravi'.

SQL> DELETE FROM EMPLOYEE WHERE ename = 'Ravi';

4. Select Command : It is used to retrive data from a table, it allows filtering

Syntax : SELECT [DISTINCT] column-list FROM table-list
[WHERE Condition]

Example : 1. To select complete record (informa from a table EMPLOYEE

SQL> select * from EMPLOYEE

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

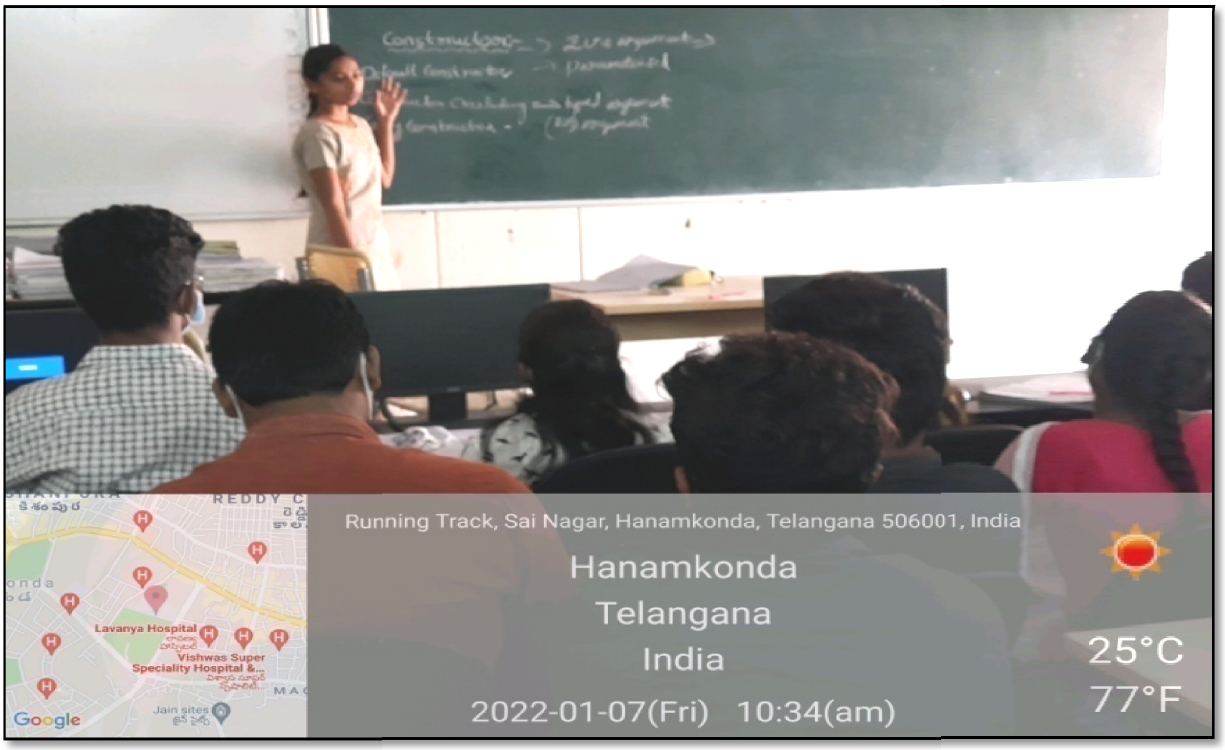
Student Seminar

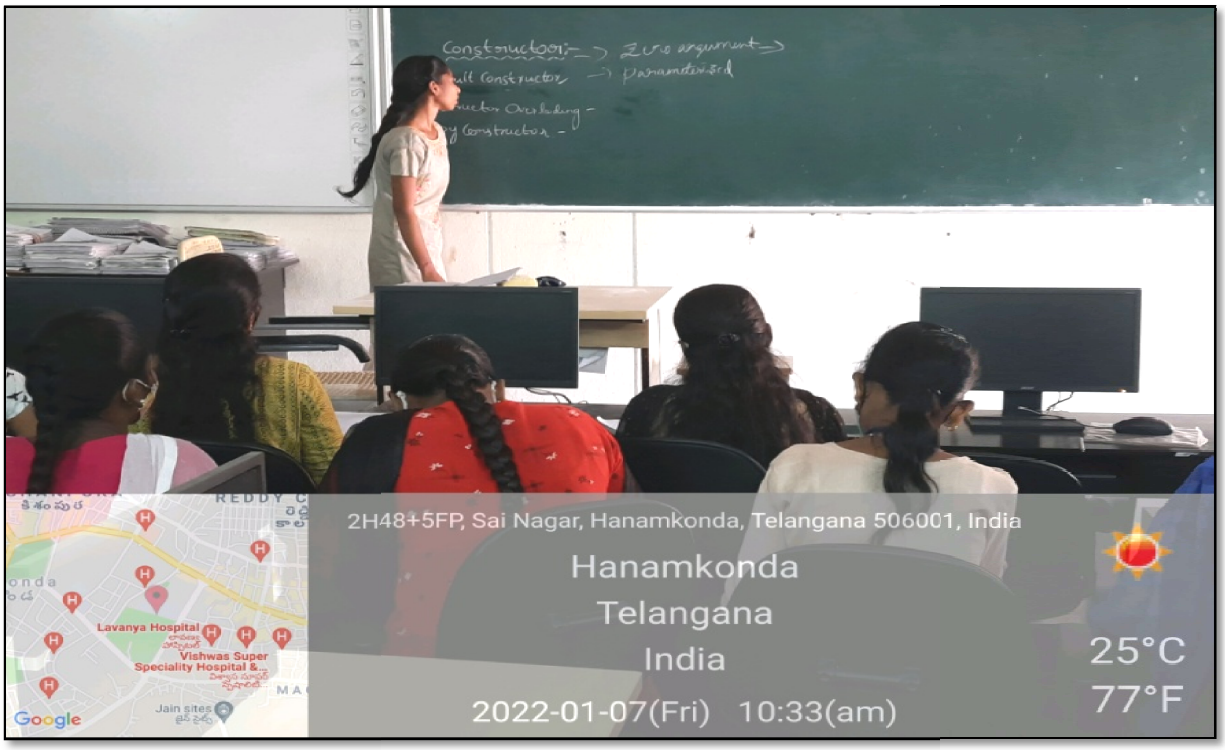
Student Name : B. Sai Sumalya **Date: 07-01-2022**

Group & Year : BSC(MPCs) – IIIYr

Subject : Programming in JAVA **Topic : Constructor**

Faculty Involved : Dr. D. Suresh Babu





KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal.

STUDENT SEMINAR

KAKATIYA GOVT. COLLEGE Hanamkonda, Warangal College Code: 006

Course : B.Sc (MPCs)-III
Subject : Programming in Java
Topic : Constructor
Faculty Involved : Dr. D. Suresh Babu
HTNo : 006204107
Name of the Student : B. Sai Sumalya
Date : 07-01-2022
Signature of the Student : B. Sai Sumalya

SNo	Hall Ticket No	Name of the Student	Signature
1	006204133	B. Akshitha	Akshitha
2	006204108	B. Manideepak	B.Manideepak
3	006204232	N. Akhilesh	N.Akhilesh
4	006204123	Bhukya Jhansi	Jhansi
5	006204101	A. Kalyan	Kalyan
6	006204130	B. Ravitheni	Ravitheni
7	006204174	G. Poojitha	G. Poojitha
8	006204190	K. Sravanthi	K. Sravanthi
9	006204189	K. Likhith	K. Likhith
10	006204164	G. Srikanth	G. Srikanth
11	006204160	G. Ishwarya	G. Ishwarya
12	006204126	B. Rajitha	B. Rajitha
13	006204142	CH. Manoharini	Manoharini
14	006204179	J. Pavan sairaj	J. Pavansairaj
15	006204137	Ch. Rakesh	Rakesh
16	006204153	Avinash. E	[illegible]
17	006204152	Thyagi	[illegible]
18	006204117	B. Prakash	Prakash
19	006204171	G. Sandeep kumar	Sandeep.
20			
21			
22			
23			
24			
25			

CONSTRUCTORS

CONSTRUCTOR:

C++ provides a special member function called the ***constructor*** which enables an object to initialize itself when it is created. This is known as ***automatic initialization*** of objects. It is special because its name is the same as the class name.

The constructor is invoked whenever an object of its associated class is created. It is called constructor because it construct the values of data members of the class.

Characteristics constructor 'or'

How constructors are different from a normal member function

- Constructor has same name as the class itself.
- Constructors don't have return type.
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

A constructor is declared and defined as follows:

```
                // class with a constructor
Class integer
{
    int m,n;
    public:
        integer (void);     //constructor declared
        --
        --
};
integer::integer (void)
{
  m=0,n=0;
}
```

// * Example program on Constructors ****

```
#include <iostream.h>
#include <conio.h>
class student
{
  char name[20];
  int s1,s2,s3,tm;
  public:
     student()
    {
       clrscr();
       cout<<"enter student name:";
       cin>>name;
       cout<<"enter 3 subject marks:";
       cin>>s1>>s2>>s3;
       tm=s1+s2+s3;
       cout<<"Student Name:"<<name<<endl;
       cout<<"Total Marks :"<<tm;
       }
};
void main()
{
   student stu;
   getch();
}
```

// Example for Constructors ****

```
#include <iostream.h>
#include <conio.h>
class sample
{
   public:
        sample();
};
sample ::sample()
{
  cout<<"  GOVERNMENT DEGREE COLLEGES, TELANGANA STATE"<<endl;
  cout<<"     CONTRACT LECTURER'S ASSOCIATION     ";
}
void main()
{
   sample s;
   clrscr();
   getch();

}
```

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

KAKATIYA GOVT. COLLEGE College Code: 006 Hanamkonda, Warangal

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Student Seminar

Student Name : V. Tharun

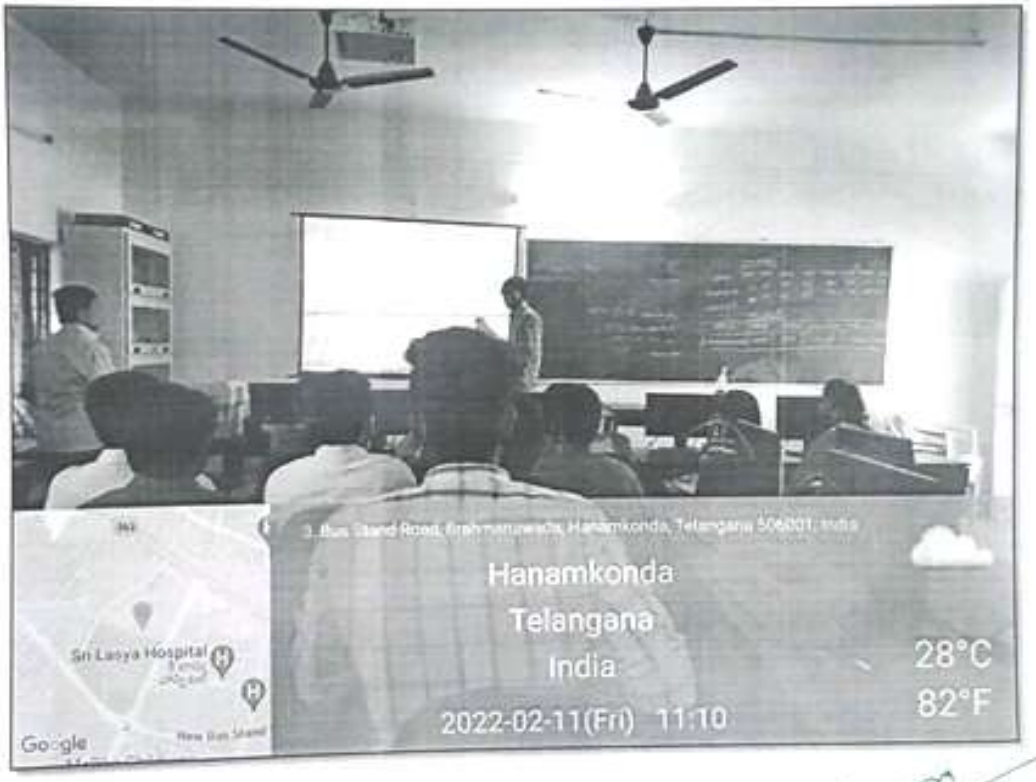
Date: 11-02-2022

Group & Year : BSc (MPCS) - IIIyr - V Sem Sec B

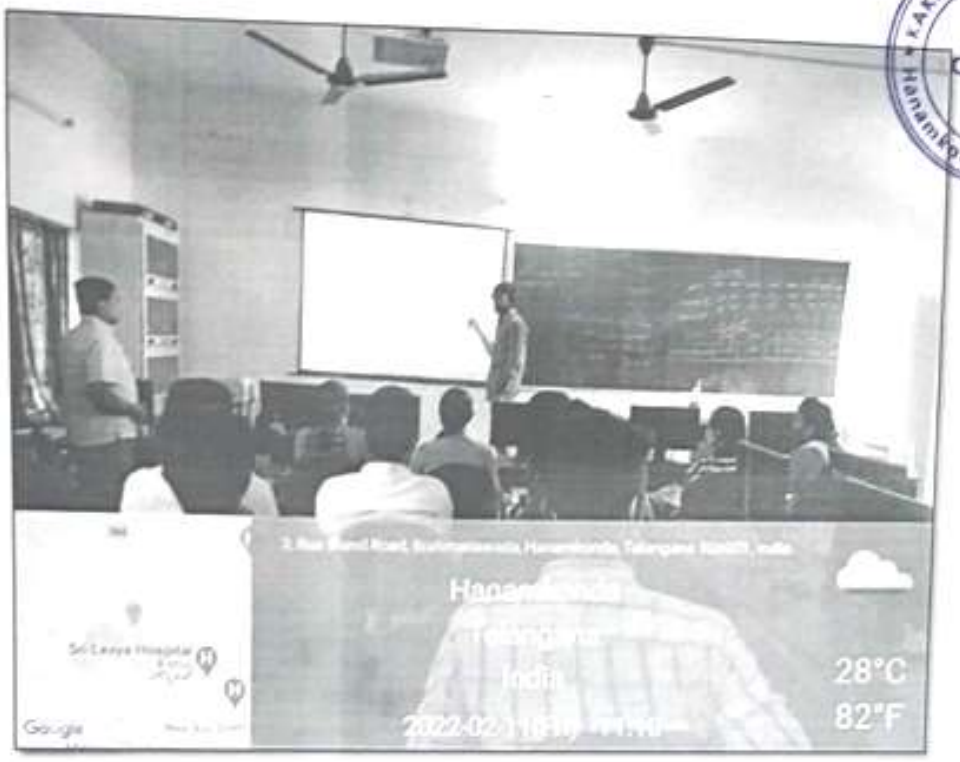
Topic : Inheritance in Java

Student Signature: V. Tharun

Signature of the Lecturer



Hc. Department of Computer Science
Kakatiya Govt. College,
HANAMKONDA.



KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal.

Dr. Department of Computer Science
Kakatiya Govt. College,
HANAMKONDA.

Attendance of Participation Students Details

S.No	HTNO	Student name	Signature
1.	006204287	K. Akshaya	Akshaya
2.	006204218	M. Bindu	Bindu
3.	006204297	U. Ashwini	Ashwini
4.	006204264	R. Rajakumari	Rajakumari
5.	006204247	P. Jyoshna	Jyshna
6.	006204246	P. Swathi	Swathi
7.	006204280	V. Mounika	Mounika
8.	006204281	V. Nagamani	Nagamani
9.	006204285	Y. Vishnu vardhini	Vishnuvardini
10.	006204211	M. Shireesha	Shireesha
11.	006204284	V. Pavan	Pavan.
12.	006204286	Y. Ganesh	Ganesh.
13.	006204265	S. Rohith	Rohith
14.	006204209	B. RavTeja	Raveteja
15.	006204255	P. Prasanth Kiran	Prasanth
16.	006204252	P. Shreeja	Shreeja
17.	006204251	P. Suman	Suma
18.	006204243	P. Sheshu	Sheshu
19.	006204240	N. Sai pranay	Sai pranay
20.	006204238	N. Vamshi	Vamshi.

KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal.

Incharge
Dept. of Computer Science
Kakatiya Government College
Hanamkonda, Warangal.

S.No.	HTNO	Student name	Signature
21.	0062042 32	N. Adrash	Adrash
22.	00620 4228	N. Sindhuja	Sindhuja
23.	00620 4214	Md. Masthana	Masthana
24.	0062042 69	Sh. kosar	kosar
25.	00620 4213	Md. Ashraf	Ashraf
26.	006 20 4207	M. prashanth kiran	prashanth
27.	00620 4200	M. kalyan	kalyan
28.	00620 4271	T Naresh	naresh
29.	0062042 58	R. Premalatha	Premalatha

KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal.

Incharge
Dept. of Computer Science
Kakatiya Government College
Hanamkonda, Warangal.

Inheritance in Java

Inheritance is most powerful features 'or' technique of Object Oriented Programming. In this technique we reuse something that already exists rather than trying to create the same all over again. It would not only save time and money but also reduce less confident and increase reliability.

In Java, Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.

Inheritance 'or' **Derivation** is the process of creating new classes from an existing class (in other words it is process of passing the properties of one class to another class). The existing class is known as the **'Base class'** or **'Parent Class'** and the created new class is called as a **'Derived Class'** or **'Child Class'**. In some OOP languages, the **base** and **derived** classes are called as **'Super'** and **'Subclasses'** respectively.

The derived class inherits all capabilities of the Base class. It can also ***add / extend*** some more features to this class. The Base class is unchanged by its process.

The main advantages of the inheritance are:

- Reusability of the code.
- To increase the reliability of the code, and
- To add some *enhancements / extend* to the base class.

The syntax of Java Inheritance:

```
Class derived-class-name extends Base-class-name
  {
   ......................
   ......................
   ...................... //members of derived class
   ......................
   ......................
  }
```

Here, we use ***extends*** keyword to making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

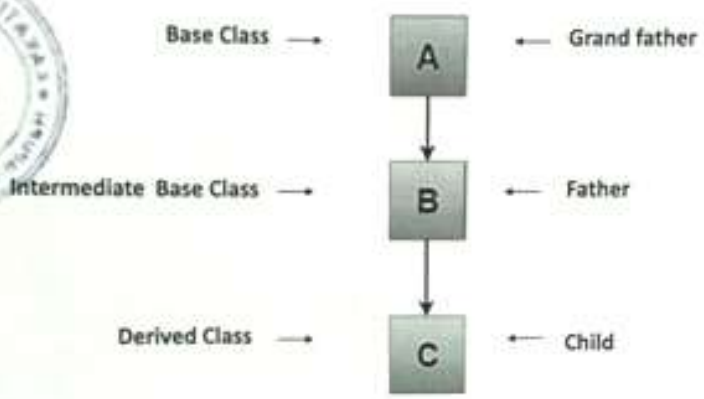
Types of Inheritance in Java

Java supports the following types of inheritance:

1. Single inheritance
2. Multilevel inheritance
3. Hierarchical inheritance
4. Multiple inheritance
5. Hybrid inheritance

a = 10
b = 20
Sum of a+b =30

2. Multilevel Inheritance : In the Multilevel Inheritance, we derived a class from another derived class. In this we create at least 3 Classes. One is the original **A** base class (Grand father), second one is a derived class from base class, the second class is also called intermediate **B** class
(Father), and the third class is another **C** derived class (Child). It is derived from, already derived class (Father). The chain is known as ***Inheritance Path***.



A derived class with multilevel Inheritance is declared a follows:

```
Class  A                  //Base class
{
  ..................
  ..................
}
Class  B extends  A       //B derived from A
{
  ..................
  ..................
}
Class  C extends  B       //C derived from B
{
  ..................
  ..................
}
```

This process can be extended to any number of levels. See the following program.

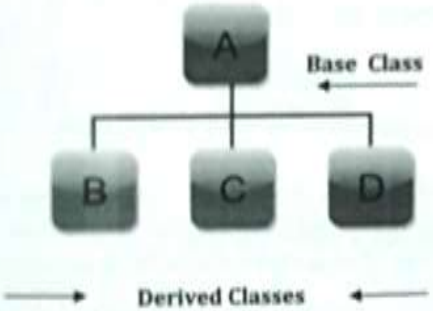
Example 1: // Program on Multi Level Inheritance

```
class Student
{
  int htno = 101;
  String sname = "Shiva Sai";
  void stu_display()
  {
   System.out.println("Student HTNO :"+htno);
   System.out.println("Student Name :"+sname);
  }
}

class Marks extends Student
{
  int s1 = 67,s2 = 87,s3 = 90;
  void marks_display()
  {
   System.out.println("Subject1 Marks ="+s1);
   System.out.println("Subject2 Marks ="+s2);
   System.out.println("Subject3 Marks ="+s3);
  }
}

class Result extends Marks
{
  int tm = s1+s2+s3;
  double avg = tm/3;
  void res_display()
  {
   System.out.println("Total Marks = "+tm);
   System.out.println("Average Marks= "+avg);
  }
}

class Multinher1
{
 public static void main(String[] args)
 {
  Result obj = new Result();
  obj.stu_display();
  obj.marks_display();
  obj.res_display();
 }
}
```

Output: *L:\Java RM Programs>javac Multinher1.java*

```
L:\Java RM Programs>java Multinher1
Student HTNO :101
Student Name :Shiva Sai
Subject1 Marks =67
Subject2 Marks =87
Subject3 Marks =90
Total Marks  = 244
Average Marks= 81.0
```

3. Hierarchical Inheritance : Hierarchical Inheritance is process of Creating, more the one Derived classes from one Base class only. So, additional members are added through inheritance to extend the capabilities of a Class. It support to the hierarchical design of a program.



Syntax of Hierarchical inheritance:

```
class A
    {
        // body of the class A.
    }
class B  extends  public A
    {
        // body of class B.
    }
class C  extends  public A
    {
        // body of class C.
    }
class D  extends  public A
    {
        // body of class D.
    }
```

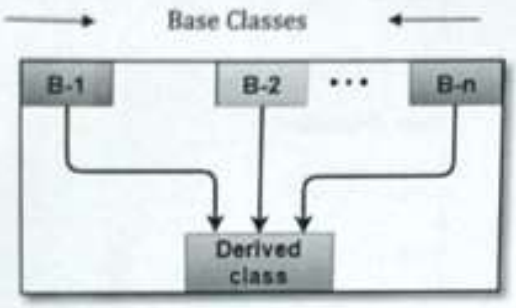
Example : // Java program to illustrate Hierarchical inheritance

```
class A
  {
    void print_A()
      {
        System.out.println("Class A");
      }
  }

class B extends A
  {
    void print_B()
      {
        System.out.println("Class B");
      }
  }

class C extends A
  {
    void print_C()
      {
        System.out.println("Class C");
      }
  }

class D extends A
  {
    void print_D()
      {
        System.out.println("Class D");
      }
  }

class Hierarchi
{
  public static void main(String[] args)
      {
        B obj_B = new B();
        obj_B.print_A();
        obj_B.print_B();

        C obj_C = new C();
        obj_C.print_A();
        obj_C.print_C();

        D obj_D = new D();
        obj_D.print_A();
        obj_D.print_D();
      }
}
```

Output: L:\Java RM Programs>javac Hierarchi.java

```
L:\Java RM Programs>java Hierarchi
Class A
Class B
Class A
Class C
Class A
Class D
```

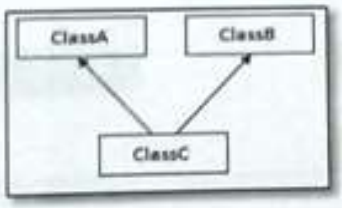
4. Multiple Inheritance (Through Interfaces): Multiple Inheritance is the process of creating a new Derived class from more than one Base classes. Multiple Inheritances allows us to combine the features of several existing classes as a starting point for defining new derived class.



In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes. Please note that Java does **not** support multiple inheritances with **classes.** In java, we can achieve multiple inheritances only through **Interfaces**.

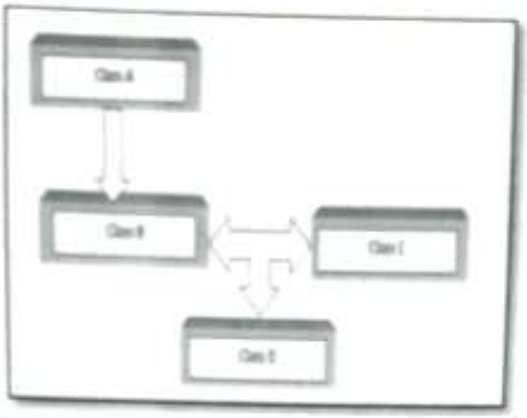
Why multiple inheritance is not supported in java?

Consider A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class. The compile give the `Compile Time Error`.



5. Hybrid Inheritance (Through Interfaces): It is a mix of two or more of the above types of inheritance. Since java doesn't support multiple inheritances with classes, hybrid inheritance is also not possible with classes. In java, we can achieve hybrid inheritance only through Interfaces.

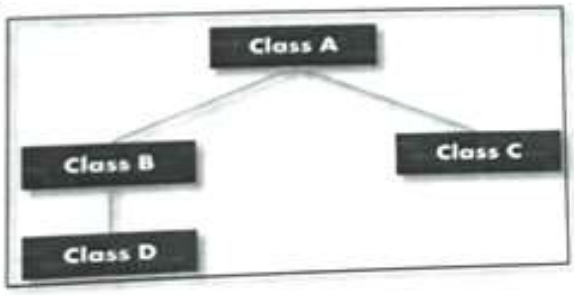
Figure 1:



Hybrid Inheritance Syntax for Figure 1:

```
Class  A
{
_____;
};
Class  B : public A
{
  _____;
};
Class  C
{
_____;
};
Class  D : public B, public C
{
  _____;
};
```

Figure 2:



// Program for above Figure Hybrid Inheritance in Java

```
class A
{
   int a;
}
class B extends A
```

```
{
  int b;
}
class C extends A
{
  int c;
  C()
  {
   a=10;
   c=20;
  }
  void sum()
  {
    System.out.println("Class C Sum() = "+(a+c));
  }
}
class D extends B
{
  int d;
  D()
  {
   a=10;
   b=20;
   d=30;
  }
  void mul()
  {
   System.out.println("Class D Mul() = "+(a*b*d));
  }
}
class Hybrid1
{
  public static void main(String[] args)
  {
    C objc = new C();
    objc.sum();
    D objd = new D();
    objd.mul();
  }
}
```

Output: L:\Java RM Programs>javac Hybrid1.java

```
L:\Java RM Programs>java Hybrid1
Class C Sum() = 30
Class D Mul() = 6000
```

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

KAKATIYA GOVT. COLLEGE College Code: 006 Hanamkonda, Warangal.

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Student Seminar

Student Name : K. Prudhviraj

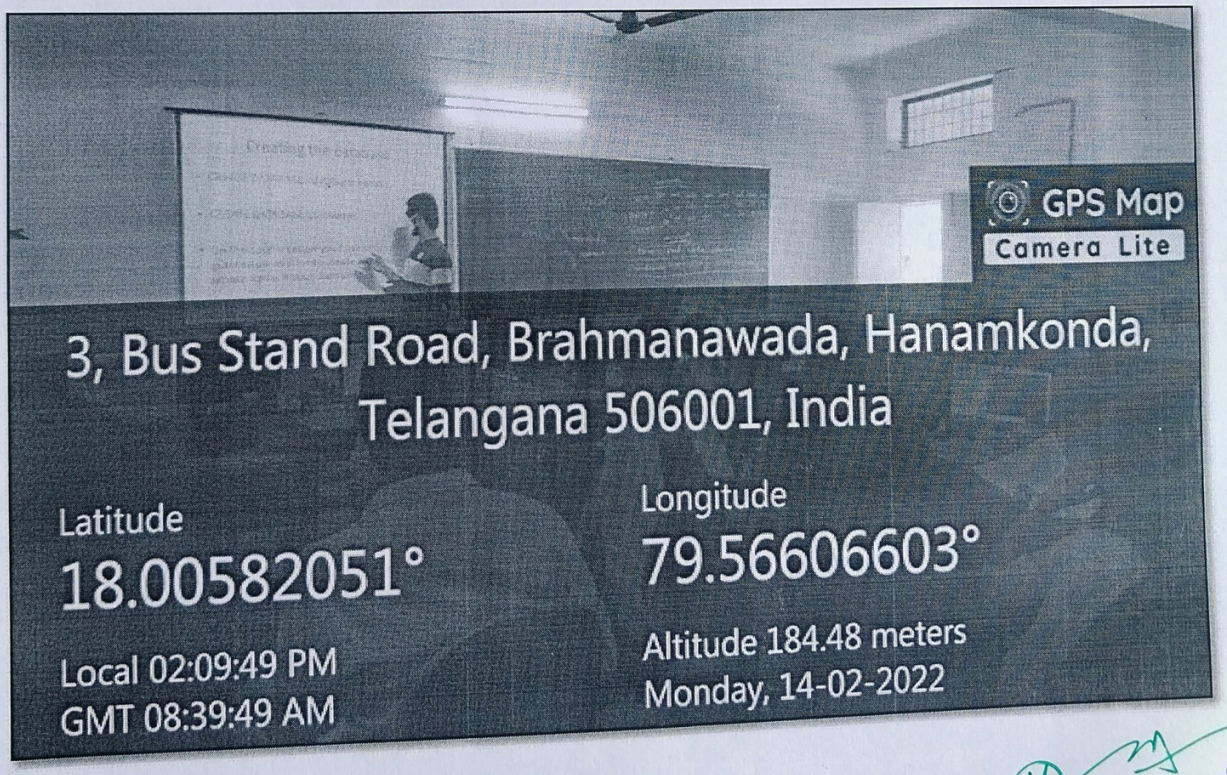
Date: 14-02-2022

Group & Year : B.Com II year, III sem

Topic : SQL

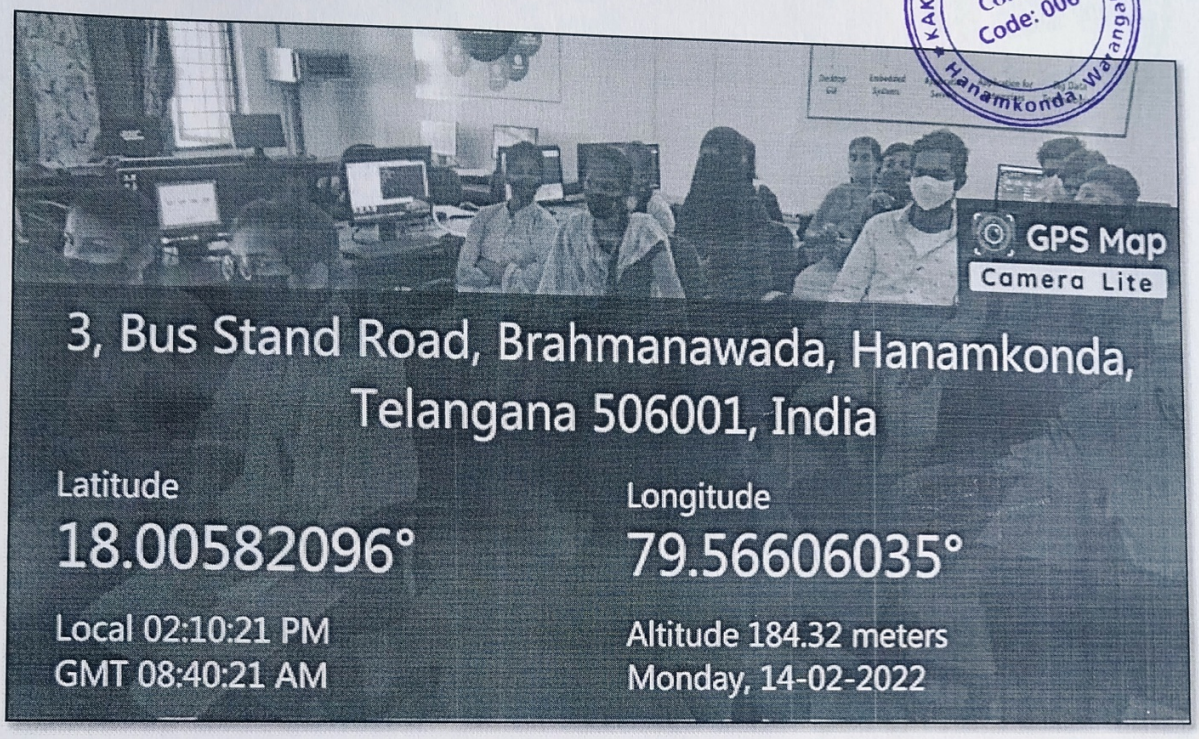
Student Signature: K. Prudhvi Raj

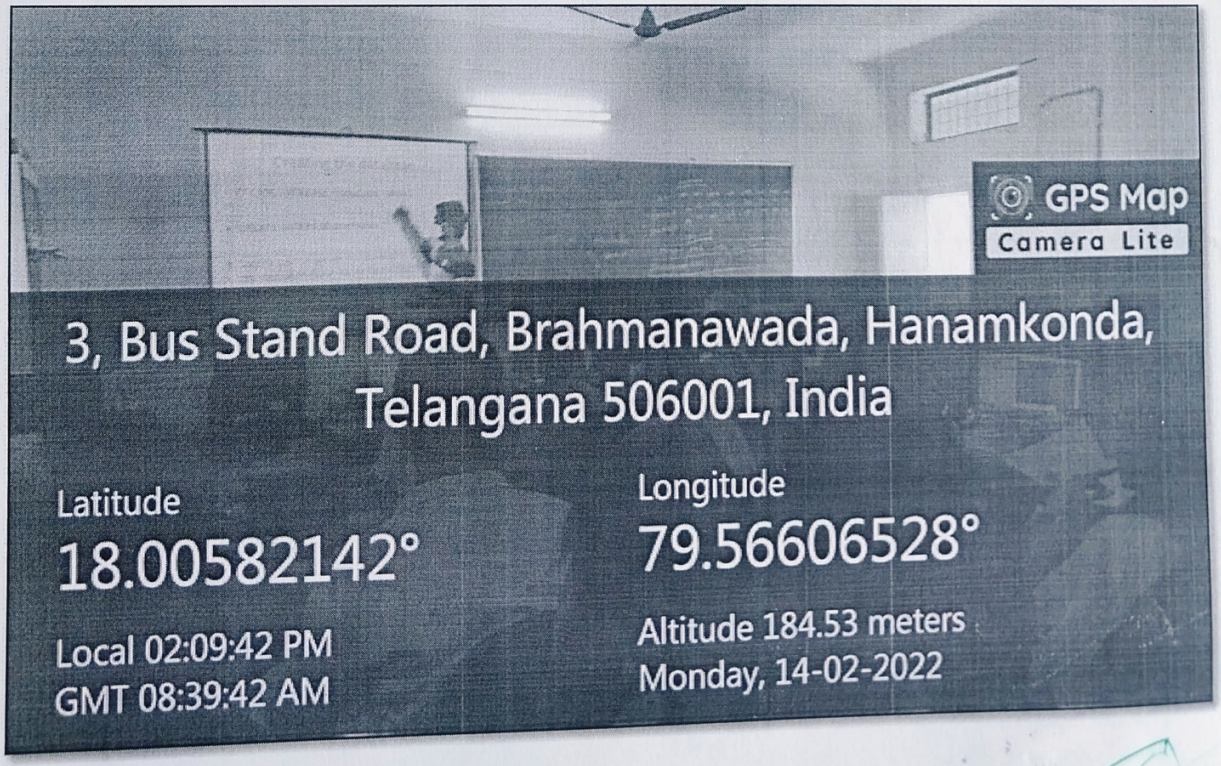
Signature of the Lecturer



H.O.D. Department of Computer Science
Kakatiya Govt. College
HANAMKONDA.

KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal.





I/c. Department of Computer Science
Kakatiya Govt. College,
HANAMKONDA.

Basic SQL Commands

examples from
Beginning MySQL

Starting MySQL

On the course server enter the command
mysql
You should then see the MySQL prompt
mysql>

To end your MySQL session use the quit command
mysql> quit;

Creating the database

- CREATE DATABASE <database name>;
- CREATE DATABASE username;
- On the course server you have only been granted permission to create a database whose name is your username.

Using a database

- USE <database name>;
- USE username;
- DROP <database name>;
- DROP username;

Deleting a database

- DROP DATABASE [IF EXISTS] <databasename>;
- DROP DATABASE username;
- This deletes the database and all tables and contents. Use with caution.

Create Table

Backus Naur Form (BNF) Notation

```
<table definition>::=
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] <table name>
{<table element> [{, <table element>}...]}
[<table option> [<table option>...]]

<table element>::=
<column definition>
| {[CONSTRAINT <constraint name>] PRIMARY KEY
    (<column name> [{, <column name>}...]}}
| {[CONSTRAINT <constraint name>] FOREIGN KEY [<index name>]
    (<column name> [{, <column name>}...]) <reference definition>}
| {[CONSTRAINT <constraint name>] UNIQUE {INDEX} [<index name>]
    (<column name> [{, <column name>}...]}}
| {{INDEX | KEY} [<index name>] (<column name> [{, <column name>}...]}}
| {FULLTEXT {INDEX} [<index name>] (<column name> [{, <column name>}...]}}
```

Create Table (cont)

Backus Naur Form (BNF) Notation

```
<column definition>::=
<column name> <type> [NOT NULL | NULL] [DEFAULT <value>] [AUTO_INCREMENT]
[PRIMARY KEY] [COMMENT '<string>'] [<reference definition>]

<type>::=
<numeric data type>
| <string data type>
| <data/time data type>

<reference definition>::=
REFERENCES <table name> [(<column name> [(, <column name>)...])]
[ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT }]
[ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT }]
[MATCH FULL | MATCH PARTIAL]

<table option>::=
[ENGINE = {BDB | MEMORY | ISAM | INNODB | MERGE | MYISAM}]
| <additional table options>
```

Basic MySQL Data Types

Integer	Real	Text
TINYINT	FLOAT	CHAR (<length>)
SMALLINT	DOUBLE	VARCHAR (<maxlength>)
MEDIUMINT	DOUBLE PRECISION	
INT	REAL	
INTEGER	DECIMAL	
BIGINT	DEC	
	NUMERIC	
	FIXED	

Create Table

Example

```
CREATE TABLE Parts
(
PartID INT NOT NULL,
PartName VARCHAR(40) NOT NULL,
CatID INT NOT NULL,
PRIMARY KEY (PartID)
);
```

Special Note

- If you are using Putty you can copy & paste the SQL commands from the PowerPoint slides into MySQL.

TABLE Parts

PartID	PartName	CatID

Inserting elements

Backus Naur Form (BNF) Notation

```
<insert statement>::=
INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO]
{<values option> | <set option> | <select option>}

<values option>::=
<table name> [(<column name> [(, <column name>)...])]
VALUES ({<expression> | DEFAULT} [(, {<expression> | DEFAULT})...])
      [(, ({<expression> | DEFAULT} [(, <expression> | DEFAULT})...])...]

<set option>::=
<table name>
SET <column name>={<expression> | DEFAULT}
      [(, <column name>={<expression> | DEFAULT})...]

<select option>::=
<table name> [(<column name> [(, <column name>)...])]
<select statement>
```

Inserting elements

```
INSERT INTO Parts
   (PartID, PartName, CatID)
VALUES
   (1001,'Guy wire assembly',503),
   (1002,'Magnet',504);
```

```
INSERT INTO Parts
VALUES
   (1003,'Regulator',505);
```

TABLE Parts

PartID	PartName	CatilD
1001	Guy wire assembly	503
1002	Magnet	504
1003	Regulator	505
1004	Brushes	504
1006	Generator	506
1006	Dump load system	506
1007	Power assembly	501

Select Statement

Backus Naur Form (BNF) Notation

```
<select statement>::=
SELECT
[<select option> [<select option>...]]
{* | <select list>}
[<export definition>]
[
    FROM <table reference> [{, <table reference>}...]
    [WHERE <expression> [{<operator> <expression>}...]]
    [GROUP BY <group by definition>]
    [HAVING <expression> [{<operator> <expression>}...]]
    [ORDER BY <order by definition>]
    [LIMIT [<offset>,] <row count>]
    [PROCEDURE <procedure name> [{<argument> [{, <argument>}...]}]]
    [{FOR UPDATE} | {LOCK IN SHARE MODE}]
]
```

Select Statement (cont)

Backus Naur Form (BNF) Notation

```
<select option>::=
{ALL | DISTINCT | DISTINCTROW}
| HIGH_PRIORITY
| {SQL_BIG_RESULT | SQL_SMALL_RESULT}
| SQL_BUFFER_RESULT
| {SQL_CACHE | SQL_NO_CACHE}
| SQL_CALC_FOUND_ROWS
| STRAIGHT_JOIN

<select list>::=
{<column name> | <expression>} [[AS] <alias>]
[{, {<column name> | <expression>} [[AS] <alias>]}...]

<export definition>::=
INTO OUTFILE '<filename>' [<export option> [<export option>]]
| INTO DUMPFILE '<filename>'

<export option>::=
{FIELDS
    [TERMINATED BY '<value>']
    [[OPTIONALLY] ENCLOSED BY '<value>']
    [ESCAPED BY '<value>']}
| {LINES
```

SELECT Examples

```
SELECT * FROM Parts;
```

```
SELECT PartID, PartName FROM Parts;
```

```
SELECT PartID, PartName FROM Parts
WHERE
   CatiID = 504;
```

Joining Tables with SELECT

Backus Naur Form (BNF) Notation

```
<select statement>::=
SELECT
[<select option> [<select option>...]]
{* | <select list>}
[
    FROM {<table reference> | <join definition>}
    [WHERE <expression> [{<operator> <expression>}...]]
    [GROUP BY <group by definition>]
    [HAVING <expression> [{<operator> <expression>}...]]
    [ORDER BY <order by definition>]
    [LIMIT [<offset>,] <row count>]
]

<join definition>::=
{<table reference>, <table reference> [{, <table reference>}...]}
| {<table reference> [INNER | CROSS] JOIN <table reference> [<join condition>]}
| {<table reference> STRAIGHT_JOIN <table reference>}
| {<table reference> LEFT [OUTER] JOIN <table reference> [<join condition>]}
| {<table reference> RIGHT [OUTER] JOIN <table reference> [<join condition>]}
| {<table reference> NATURAL [{LEFT | RIGHT} [OUTER]] JOIN <table reference>}
| {<table reference> NATURAL [{LEFT | RIGHT} [OUTER]] JOIN <table reference>}

<table reference>::=
<table name> [[AS] <alias>]
[{USE | IGNORE | FORCE} INDEX <index name> [{, <index name>}...]]

<join condition>::=
ON <expression> [{<operator> <expression>}...]
| USING (<column> [{, <column>}...])
```

Create Books Table

```
CREATE TABLE Books
(
BookID SMALLINT NOT NULL PRIMARY KEY,
BookTitle VARCHAR(60) NOT NULL,
Copyright YEAR NOT NULL
);
```

Create Example Tables

- Books
- Authors
- AuthorBook

Insert data into Books

```
INSERT INTO Books
VALUES (12786, 'Letters to a Young Poet', 1934),
(13331, 'Winesburg, Ohio', 1919),
(14356, 'Hell\'s Angels', 1966),
(15729, 'Black Elk Speaks', 1932),
(16284, 'Noncomformity', 1996),
(17695, 'A Confederacy of Dunces', 1980),
(19264, 'Postcards', 1992),
(19354, 'The Shipping News', 1993);
```

Create Authors Table

```
CREATE TABLE Authors
(
AuthID SMALLINT NOT NULL PRIMARY KEY,
AuthFN VARCHAR(20),
AuthMN VARCHAR(20),
AuthLN VARCHAR(20)
);
```

Insert data into Books

```
INSERT INTO Authors
VALUES (1006, 'Hunter', 'S.', 'Thompson'),
(1007, 'Joyce', 'Carol', 'Oates'),
(1008, 'Black', NULL, 'Elk'),
(1009, 'Rainer', 'Maria', 'Rilke'),
(1010, 'John', 'Kennedy', 'Toole'),
(1011, 'John', 'G.', 'Neihardt'),
(1012, 'Annie', NULL, 'Proulx'),
(1013, 'Alan', NULL, 'Watts'),
(1014, 'Nelson', NULL, 'Algren');
```

Create AuthorBook Table

```
CREATE TABLE AuthorBook
(
AuthID SMALLINT NOT NULL,
BookID SMALLINT NOT NULL,
PRIMARY KEY (AuthID, BookID),
FOREIGN KEY (AuthID) REFERENCES Authors
  (AuthID),
FOREIGN KEY (BookID) REFERENCES Books (BookID)
);
```

S

Insert Data into AuthorBook

```
INSERT INTO AuthorBook
VALUES (1006, 14356), (1008, 15729),
       (1009, 12786), (1010, 17695),
       (1011, 15729), (1012, 19264),
       (1012, 19354), (1014, 16284);
```

Basic Join

```
SELECT BookTitle, Copyright, Authors.AuthID
FROM Books, AuthorBook, Authors
WHERE
  Books.BookID=AuthorBook.BookID
AND
  AuthorBook.AuthID=Authors.AuthID
ORDER BY Books.BookTitle;
```

Basic Join

```
SELECT BookTitle, Copyright, Authors.AuthID
FROM Books, AuthorBook, Authors
ORDER BY BookTitle;
```

What happens when we leave off the WHERE clause?

Basic Join

```
SELECT BookTitle, Copyright, AuthID
FROM Books AS b, AuthorBook AS ab
WHERE b.BookID=ab.BookID
ORDER BY BookTitle;
```

KAKATIYA GOVERNMENT DEGREE COLLEGE,HANAMKONDA

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

BCOM CA -II YEAR - III SEM 2021-22 RDBMS

List of students who Attended student seminar

SNo	HTNo	Name	Signature
1	006212180	KOMMULA NITHIN	K.Nithin
2	006212181	KOMMULA NITHIN	
3	006212182	KONDAPALLY VENKATESH	K. Venkatesh
4	006212183	KONGARI PRUDHVI RAJ	K.prudhviRaj
5	006212184	KOPPULA HARI KRISHNA	
6	006212185	KOPPULA NAGARAJU	
7	006212186	KORUKOPPULA THARUN	K Tharun
8	006212187	KOTHAPALLI SUNNY	
9	006212188	KOUDAGANI ANIL	
10	006212189	KOULAGONI VINAY	K.Vinay
11	006212190	KOWDAGANI TEJA	K.Teja
12	006212191	KOYYADA VAMSHI	
13	006212192	KUMMARI MAHENDAR	
14	006212193	KUNUSOTH NAVEEN	K.Naveen
15	006212194	KURRI MOUNIKA	K. Mounika
16	006212195	KURSAM YASHODA	
17	006212196	LAKKARSU SAI	
18	006212197	LAVUDYA NAVYA	L.Navya
19	006212198	LAVUDYA THIRUPATHI	
20	006212199	LINGALA SPANDANA	
21	006212200	LODE SHRAVAN KUMAR	L. Sravan
22	006212201	LOTTI POOJITHA	L.Poojitha
23	006212202	LYADALLA KEERTHI	
24	006212203	MADDEBOINA PRASHANTH	
25	006212204	MADI SHASHI VARDHAN	
26	006212205	MAHANKALI NAGARAJU	
27	006212206	MALLABOINA RAKESH	M.Rakesh
28	006212207	MALOTH THIRUPATHI	

KAKATIYA GOVT. COLLEGE ★ Hanamkonda, Warangal. ★ College Code: 006

29	006212208	MAMIDI SRAVAN KUMAR	Sravan
30	006212209	MANCHALA RAJU	M. Raju
31	006212210	MANDA CHANDU	
32	006212211	MANDHA VAMSHI	
33	006212212	MANKURTHI SURESH	
34	006212213	MANTHURTHI RAKESH	
35	006212214	MANUPATI RAJASHEKER	M. [illegible]
36	006212215	MARAPALLI ASHISH	M. Ashish
37	006212216	MARGAM SAI RAM	
38	006212217	MARRI RANA PRATHAP	M. Rana Prathap
39	006212218	MATHANGI CHAITHANYA	
40	006212219	MATIKE DRAVENDER	M. Dravender
41	006212220	MATTAPALLI NIKHIL	
42	006212221	MATTEDA PREM KUMAR	Premkumar
43	006212222	MD ABDUL GAFFAR	GAFFAR
44	006212223	MD SULAIMAN ARSHAD	Sulaiman
45	006212224	MEDA KRANTHI	M. Kranthi
46	006212225	MEDIPALLI APARNA	
47	006212226	MEDIPALLI BHARATH	
48	006212227	METTUPALLY VINEETH	
49	006212228	MISSA ISHWARYA	M. Ishwarya
50	006212229	MODEM NITHIN	M. NITHIN
51	006212230	MODEM RAVI	
52	006212231	MODEM ROHITH	M. Rohith
53	006212232	MOGALICHARLA SRIKANTH	M. Srikanth
54	006212233	MOGILI NAVEEN KUMAR	
55	006212234	MOHAMMAD AJMER	MD. AJMER
56	006212235	MOHAMMAD KHAJAPASHA	
57	006212236	MOHAMMAD MUDHASIR AHMED	Mudhasir
58	006212237	MOHAMMAD MUHAFEEZ	
59	006212238	MOHAMMAD NADEEMUDDIN	Nadeemuddin
60	006212239	MOHAMMAD RAZZAQ	Razzaq
61	006212240	MOHAMMAD RIYAZ	Riyaz

KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal

62	006212241	MOHAMMAD YAKUB PASHA	
63	006212242	MOHAMMED SARWAR	Sarwar
64	006212243	MOLUGURI CHANDANA	
65	006212244	MOODU MAHABOOB	M. Mahaboob
66	006212245	MORE LAXMAN	M. Laxman
67	006212246	MOTHE RAJKUMAR	
68	006212247	MOTLA SHRUTHI	
69	006212248	MUDRABOINA SIDDHARTHA	M. Siddhartha
70	006212249	MUNAVATH PRAVEEN	M. Praveen
71	006212250	MUNIGELA RAJASHEKAR	
72	006212251	MUPPU ASHWINI	
73	006212252	MUTHYALA NAGESH	
74	006212253	MUTHYALA SAI KRISHNA	
75	006212254	NAGARAM NAGA RAJU	N. NagaRaju
76	006212255	NALLA SAMBARAJU	N. Sambaraju
77	006212256	NALLABOOGA SHASHI KUMAR	
78	006212257	NALLAM ADHITHYA	
79	006212258	NAMANI SHIVA SAI	N. Shivasai
80	006212259	NAMOJU BHAVANI	N. Bhavani
81	006212260	NAMPALLI MURALIDHAR	N. Muralidhar
82	006212261	NAMPALLY SAHITH	N. Sahith
83	006212262	NANABOINA JITHENDER	N. Jithender
84	006212263	NARAYANA RAVINDER	N. Ravinder
85	006212264	GAJARLA NAVEEN KUMAR	G. Naveen Kumar
86	006212265	NIDIGONDA PRAVEEN KUMAR	
87	006212266	MANCHALA NITHIN	M. Nithin
88	006212267	OJJALA ANUSHA	
89	006212268	ORSU KIRITI	
90	006212269	PADALA HARI CHARAN	P. Harichar
91	006212270	PAIDAKULA SAI VINEELA	
92	006212271	PAIDIPELLY SAI KUMAR	P. Sai Kumar

KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal

KAKATIYA GOVERNMENT COLLEGE

HANUMAKONDA, DIST. HANUMAKONDA.

(Affiliated to Kakatiya University)

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

Student Seminar

Student Name : K. Abhinav Kalyan

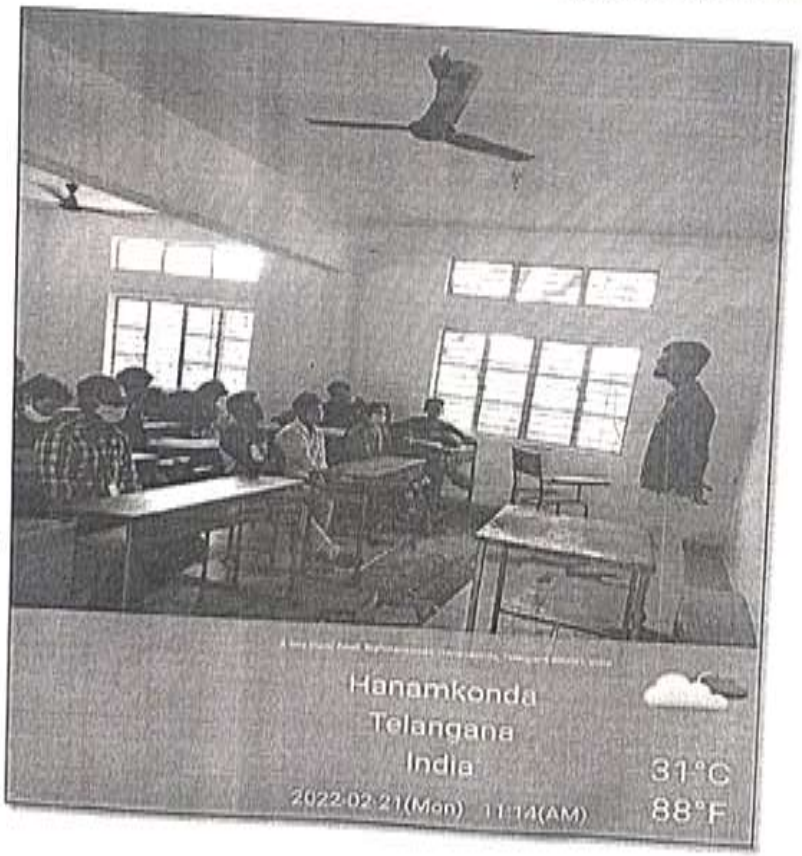
Date: 21-02-2022

Group & Year : B.Com(CA) IIyr III Sem.

Topic : Concurrency Control in RDBMS Packages.

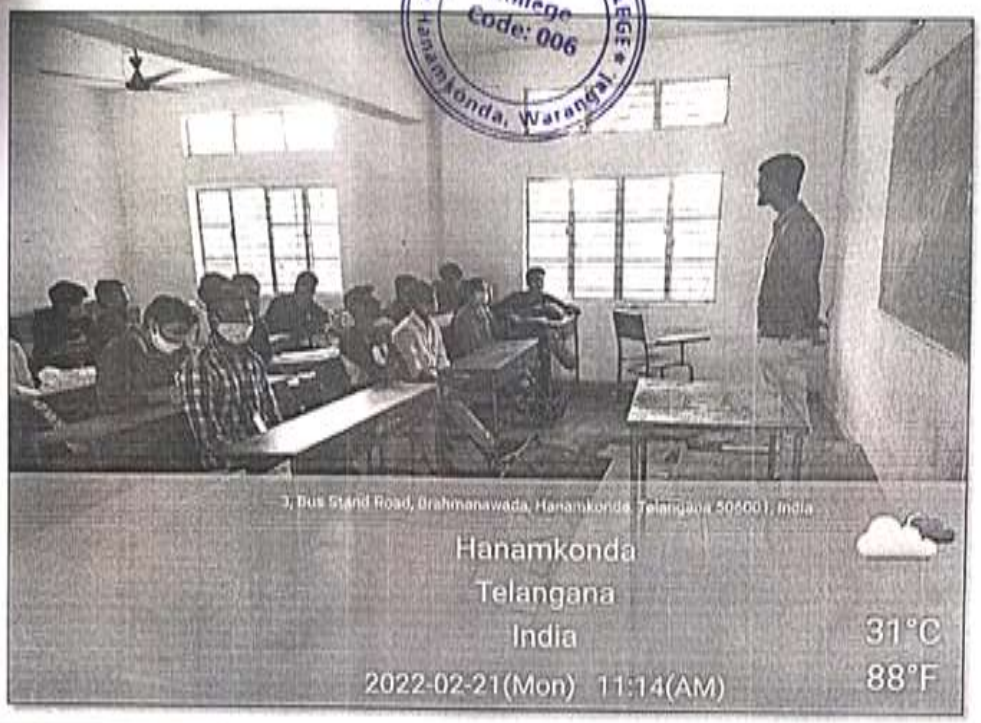
Student Signature: K Abhinav

D. Praveen
Signature of the Lecturer



Hc. Department of Computer Scienc.
Kakatiya Govt. College,
HANAMKONDA.

KAKATIYA GOVT. COLLEGE
College Code: 006
Hanamkonda, Warangal.



Attendance of Participation Students Details

S.No.	Hall Ticket No.	Student's Name	Signature
1)	006212138	G. Raj Kumar	G. Rajkumar
2)	006212149	J. Tharun kumar	Tharun
3)	006212177	K. vijay Kumar	K. V
4)	006212147	J. Ganesh.	J. Ganesh.
5)	006212152	J. Uday Kiran	J. uday
6)	006212119	G. Akhil	GAKhil
7)	006212128	G. Thirupathi	G. Thirupathi
8)	006212171	K. Rajeshwar Reddy	K. Rajesh
9)	006212168	K. Srikanth	K. Srikanth
10)	006212109	G. Kalyan	Gaddi Kalyan

Sl No.	Hall Ticket No.	Student's Name	Signature
11.	006212106	G. Poojitha	Poojitha
12.	006212111	G. Divya.	Divya
13.	006212127	G. Srikanth.	Srikanth
14.	006212130	G. Rambabu	Rambabu
15.	006212131	G. Suresh.	G. Suresh
16.	006212139	G Rani.	Rani
17.	006212144	J. Deepak.	Deepak
18.	006212153	J. Sai Kumar	Sai Kumar
19.	006212166	K. Bunny.	Bunny
20.	006212174	K. Zareen.	Zareena
21.	006212175	K. Kranthi Kumar	KKumar

Student Seminar on Concurrency Control in RDBMS

Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database.

But before knowing about concurrency control, we should know about concurrent execution.

Concurrent Execution in DBMS

- In a multi-user system, multiple users can access and use the same database at one time, which is known as the concurrent execution of the database. It means that the same database is executed simultaneously on a multi-user system by different users.
- While working on the database transactions, there occurs the requirement of using the database by multiple users for performing different operations, and in that case, concurrent execution of the database is performed.
- The thing is that the simultaneous execution that is performed should be done in an interleaved manner, and no operation should affect the other executing operations, thus maintaining the consistency of the database. Thus, on making the concurrent execution of the transaction operations, there occur several challenging problems that need to be solved.

Problems with Concurrent Execution

In a database transaction, the two main operations are **READ** and **WRITE** operations. So, there is a need to manage these two operations in the concurrent execution of the transactions as if these operations are not performed in an interleaved manner, and the data may become inconsistent. So, the following problems occur with the Concurrent Execution of the operations:

Problem 1: Lost Update Problems (W - W Conflict)

The problem occurs *when two different database transactions perform the read/write operations on the same database items in an interleaved manner (i.e., concurrent execution) that makes the values of the items incorrect hence making the database inconsistent.*

For example:

Consider the below diagram where two transactions T_X and T_Y, are performed on the same account A where the balance of account A is Rs.500.

Time	Tx	Ty
t_1	READ (A)	—
t_2	A = A - 50	
t_3	—	READ (A)
t_4	—	A = A + 100
t_5	—	—
t_6	WRITE (A)	—
t_7		WRITE (A)

LOST UPDATE PROBLEM

- At time t1, transaction T_X reads the value of account A, i.e., Rs.500 (only read).
- At time t2, transaction T_X deducts Rs.50 from account A that becomes Rs.450 (only deducted and not updated/write).
- Alternately, at time t3, transaction T_Y reads the value of account A that will be Rs.500 only because T_X didn't update the value yet.
- At time t4, transaction T_Y adds Rs.100 to account A that becomes Rs.600 (only added but not updated/write).
- At time t6, transaction T_X writes the value of account A that will be updated as Rs.450 only, as T_Y didn't update the value yet.
- Similarly, at time t7, transaction T_Y writes the values of account A, so it will write as done at time t4 that will be Rs.600. It means the value written by T_X is lost, i.e., Rs.450 is lost.

Hence data becomes incorrect, and database sets to inconsistent.

Dirty Read Problems (W-R Conflict)

The dirty read problem occurs *when one transaction updates an item of the database, and somehow the transaction fails, and before the data gets rollback, the updated database item is accessed by another transaction. There comes the Read-Write Conflict between both transactions.*

For example:

Consider two transactions T_X and T_Y in the below diagram performing read/write operations on account A where the available balance in account A is $300:

Time	T_X	T_Y
t_1	READ (A)	—
t_2	A = A + 50	—
t_3	WRITE (A)	—
t_4	—	READ (A)
t_5	SERVER DOWN ROLLBACK	—

DIRTY READ PROBLEM

- At time t1, transaction T_X reads the value of account A, i.e., Rs.500.
- At time t2, transaction T_X adds Rs.50 to account A that becomes Rs.550.
- At time t3, transaction T_X writes the updated value in account A, i.e., Rs.550.
- Then at time t4, transaction T_Y reads account A that will be read as Rs.550.
- Then at time t5, transaction T_X rollbacks due to server problem, and the value changes back to Rs.500 (as initially).
- But the value for account A remains Rs.550 for transaction T_Y as committed, which is the dirty read and therefore known as the Dirty Read Problem.

Unrepeatable Read Problem (W-R Conflict)

Also known as Inconsistent Retrievals Problem that occurs when in a transaction, two different values are read for the same database item.

For example:

Consider two transactions, T_X and T_Y, performing the read/write operations on account A, having an available balance = Rs.500. The diagram is shown below:

Time	T_X	T_Y
t_1	READ (A)	—
t_2	—	READ (A)
t_3	—	A = A + 100
t_4	—	WRITE (A)
t_5	READ (A)	—

UNREPEATABLE READ PROBLEM

- At time t1, transaction T_X reads the value from account A, i.e., Rs.500.
- At time t2, transaction T_Y reads the value from account A, i.e., Rs.500.
- At time t3, transaction T_Y updates the value of account A by adding Rs.100 to the available balance, and then it becomes Rs.600.
- At time t4, transaction T_Y writes the updated value, i.e., Rs.600.
- After that, at time t5, transaction T_X reads the available value of account A, and that will be read as Rs.600.
- It means that within the same transaction T_X, it reads two different values of account A, i.e., Rs.500 initially, and after updation made by transaction T_Y, it reads Rs.600. It is an unrepeatable read and is therefore known as the Unrepeatable read problem.

Thus, in order to maintain consistency in the database and avoid such problems that take place in concurrent execution, management is needed, and that is where the concept of Concurrency Control comes into role.

Concurrency Control

Concurrency Control is the working concept that is required for controlling and managing the concurrent execution of database operations and thus avoiding the inconsistencies in the database. Thus, for maintaining the concurrency of the database, we have the concurrency control protocols.

Concurrency Control Protocols

The concurrency control protocols ensure the *atomicity, consistency, isolation, durability* and *serializability* of the concurrent execution of the database transactions. Therefore, these protocols are categorized as:

- Lock Based Concurrency Control Protocol
- Time Stamp Concurrency Control Protocol
- Validation Based Concurrency Control Protocol.