

[Home](#)[Java](#)[Programs](#)[OOPs](#)[String](#)[Exception](#)[Multithreading](#)[Report this ad](#)

C PROGRAM TO IMPLEMENT STACK

```

#define max 3 → max is 3
void push(); → prototype
void pop(); → prototype
void pri(); → prototype
int st[max];
int top=-1;
void main()
{
    int ch;
    clrscr();
    while(1) → loop - always true
    {
        printf("\n 1. Push:");
        printf("\n 2. Pop:");
        printf("\n 3. Pri:");
        printf("\n 4. Exit:");

        printf("\n Enter the choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:push();
            break;
            case 2: pop();
            break;
            case 3: pri();
            break;
            case 4: exit();
            default:
                printf("\n wrong choice :");
        }
    }
}

```

Diagram of a stack: A vertical array of 3 cells. The top cell is labeled '2', the middle cell is labeled '1', and the bottom cell is labeled '0'. The array is labeled 'st' and 'top = -1'.

Can We Override Static Method in Java?

In Java, **overriding** and **overloading** are the two most important features of **object-oriented programming**. The feature is used when the programmer wants to achieve **polymorphism**. The question, **can we override static method in Java** or **can you overload static method in Java** are the two most important question asked in **Java interview**. In this section, we will understand overloading and overriding in short. We have also explained the answer of **why we cannot override the static method in Java?**

Static Method

The method that has a static keyword before the method name is known as a **static method**. It is also known as a **class-level method**. A copy of the static method is shared by all the objects of the class.

```
public static int sum()  
{  
}
```

We can invoke static methods by using the class name. For example, **Math.abs(a)** method. The method returns the absolute value of the passed argument. The static method cannot access instance variables or methods.

Method Overriding

It is a feature of **object-oriented programming**. It is used to achieve run-time polymorphism. The subclass provides a specific implementation of a method that is already provided by its parent class, known as **method overriding**. The signature of the method in parent and child class must be the same. In **method overriding**, which method is to be executed, decided at run-time. The decision is made according to the object that we called.



Method Overloading

It is also a feature of object-oriented programming. It is used to achieve compile-time polymorphism. It allows us to use the same method name but different signatures. If a class has more than one method with the same name but a different method signature, it is known as **method overloading**.

We have learned what is overloading and overriding. Now we move to the point.

Can we overload a static method?

The answer is **Yes**. We can overload static methods. But remember that the method signature must be different. For example, consider the following Java program.

OverloadStaticMethodExample1.java

```
public class OverloadStaticMethodExample1
{
    //static method
    public static void display()
    {
        System.out.println("Static method called.");
    }
    //overloaded static method
    public static void display(int x)
    {
        System.out.println("An overloaded static method called.");
    }
    //main method
    public static void main(String args[])
    {
        //calling static method by using the class name
        OverloadStaticMethodExample1.display();
        OverloadStaticMethodExample1.display(160);
    }
}
```

Output:

```
Static method called.
```

An overloaded static method called.

Here a question arises that **can we overload the methods if they are only different by static keyword.**

The answer is **No**. We cannot override two methods if they differ only by static keyword. For example, consider the following Java program.

OverloadStaticMethodExample2.java

```
public class OverloadStaticMethodExample2
{
    //static method
    public static void sum(int a, int b)
    {
        int c=a+b;
        System.out.println("The sum is: "+c);
    }
    //non-static method
    public void sum(int a, int b)
    {
        int c=a+b;
        System.out.println("The sum is: "+c);
    }
    //main method
    public static void main(String args[])
    {
        //calling static method by using the class name
        OverloadStaticMethodExample2.sum(12, 90);
    }
}
```



When we compile the above program, it shows the following error.

```
error: method sum(int,int) is already defined in class OverloadStaticMethodExample2
```

Can we override a static method?

No, we cannot override static methods because method overriding is based on dynamic binding at runtime and the static methods are bonded using static binding at compile time. So, we cannot override static methods.

The calling of method depends upon the type of object that calls the static method. It means:

- If we call a static method by using the parent class object, the original static method will be called from the parent class.
- If we call a static method by using the child class object, the static method of the child class will be called.

In the following example, the ParentClass has a static method named display() and the ChildClass also has the same method signature. The method in the derived class (ChildClass) hides the method in the base class. let's see an example.

OverloadStaticMethodExample3.java

```
public class OverloadStaticMethodExample3
{
    public static void main(String args[])
    {
        ParentClass pc = new ChildClass();
        //calling display() method by parent class objec
        pc.display();
    }
}
```

```
//parent class
class ParentClass
{
    //we cannot override the display() method
    public static void display()
    {
        System.out.printf("display() method of the parent class.");
    }
}

//child class
class ChildClass extends ParentClass
{
    //the same method also exists in the ParentClass
    //it does not override, actually it is method hiding
    public static void display()
    {
        System.out.println("Overridden static method in Child Class in Java");
    }
}
```

Output:

```
display() method of the parent class.
```

[< Prev](#)[Next >](#)



For Videos Join Our Youtube Channel: [Join Now](#)

Feedback


- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share




Learn Latest Tutorials







Splunk




SPSS




Swagger




Transact-SQL




Tumblr




ReactJS




Regex




Reinforcement Learning




R Programming




RxJS




React Native




Python Design Patterns



Python Pillow




Python Turtle




Keras


Preparation




Aptitude




Reasoning



Verbal Ability




Interview Questions




Company Questions



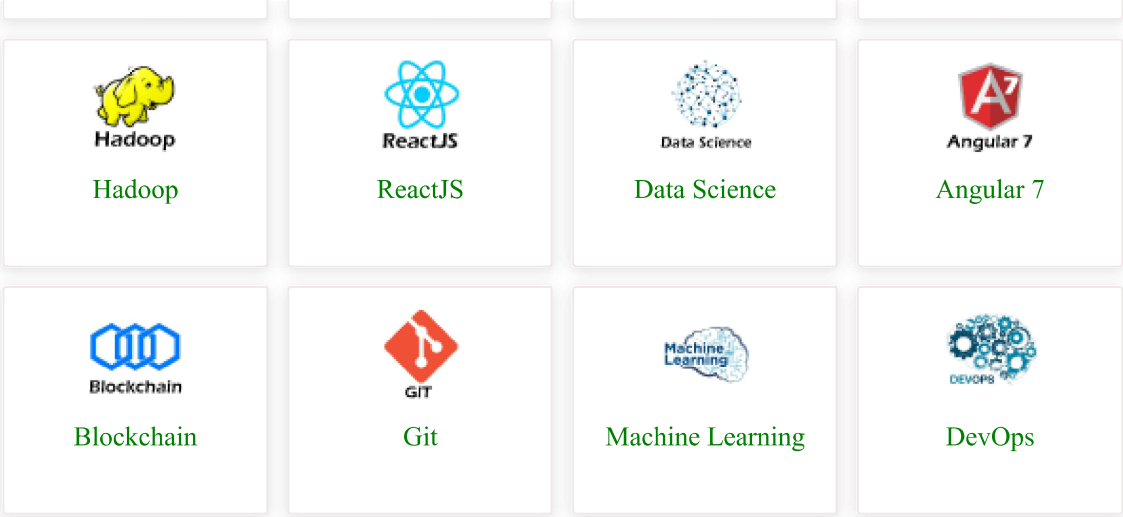
Trending Technologies



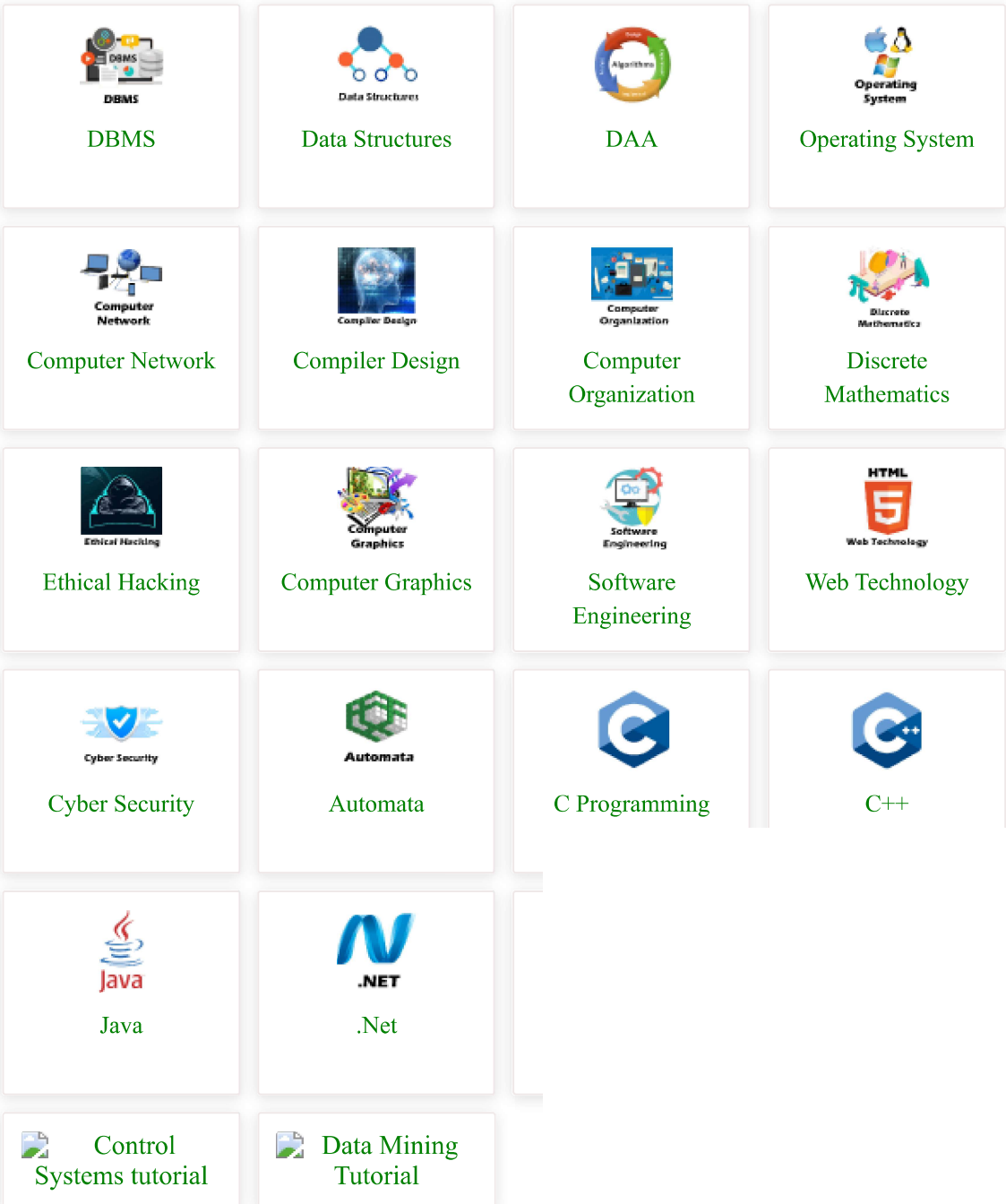
Artificial Intelligence



AWS



B.Tech / MCA



Control System

Data Mining



Data Warehouse Tutorial

Data Warehouse

