// String objects are immutable

//concat () - S13.java
//length () - S1.java
// compareTo() - S2.java
// compareToIgnoreCase () - S2.java
// to UpperCase () , & toLowerCase - S3.java
// trim () - S4.java
// startsWith () , endsWith () - S5 java
// substring () - S6.java
// charAt () - S7.java
// Intern () - S8.java
// valueOf () ← S9.java
// replace () - S10.java
// split () - S11.java

WAP to count no of digits of a no. S12.java
Sample input - 7689
Sample output - 4

Why string objects are called immutable ?
↳ unchangeable
↳ un-modify

bcoz methods can't modify the original content of string.

StringBuffer class append() Method
↳ concatenates the given argument with this string.

```
class SB1
{
    public static void main (String args[])
    {
        StringBuffer sb = new StringBuffer ("Hello");
        Sopln (sb);
        sb.append ("Java");
        Sopln (sb);
    }
}
```

```
/* Output
    Hello
    Hello Java
```

replace ( ) - replaces the given String from the specified beginIndex & endIndex

```
class  SB3
 {
    public  static  void  main  ( String  args[])
      {
         String  Buffer  sb =  new  StringBuffer ("Hello");
          sb.Buffer  ( 0, 3 , "Java ");
          Sopln  ( sb) ;
      }
   }
```

Prog - 4

delete( )

```
class  SB4
{
    psvm  ( String   args[])
     {
        StringBuffer  sb =  new  StringBuffer ("Hello");
        sb.delete ( 1 , 3) ;                                    /* Output
        System.out.println ( sb) ;                                       Hlo
     }
 }
```

Prog 5

reverse ( ) - class reverse the current String.

```
class  SB5
 {
     psvm ( String  args[])
      {
         StringBuffer  sb =  new  StringBuffer  ("Hello");
         sb.reverse ( ) ;
         Sopln  ( sb) ;
      }
  }
```

replace ( ) - replaces the given String from the specified beginIndex & endIndex

```
class SB3
{
    public static void main (String args[])
    {
        String Buffer sb = new StringBuffer ("Hello");
        sb.Buffer ( 0,3 ,"Java");
        Sopln ( sb) ;
    }
}
```

Prog- 4

delete( )

```
class SB4
{
    psvm ( String args[])
    {
        StringBuffer sb= new StringBuffer ("Hello");
        sb.delete (1,3);                                    /* Output
        System.out.println (sb) ;                                  Hlo
    }
}
```

Prog 5

reverse ( ) - class reverse the current String.

```
class SB5
{
    psvm ( String args[])
    {
        StringBuffer sb = new StringBuffer ("Hello");
        sb.reverse ( );
        Sopln ( sb);
    }
}
```

Pg - 6

```
class  SB6
 {
     psvm ( String args [] )
       {
          StringBuffer  sb= new  StringBuffer ("KIIT java is my favourite language");
          Sopln ( sb.capacity() ) ;
          // sb. append (" Hello");
          // Sopln (sb. capacity ());
          // sb. append (" Java is my favourite language");
           sb. append (" KIIT java is my favourite language");
        // sb. append (" KIIT Universt");
        // System.out.println (sb);
        // Sopln (sb.capacity ());
      }
  }
```

Prog 7

length () method

```
 class  SB7
  {
      psvm  ( string  args [ ] )
        {
          StringBuffer  sb= new  StringBuffer ("KIIT  University")
           int  p =  sb. length() ;
           Sopln ( p);
         }
      }
```

Prog 8

it is used to text at the specified position.

```
StringBuffer  insert ( int  index , string str )
String Buffer  insert ( int  index, char ch)
class  SB8
 {
     psvm ( String ar[] )
       {
           String Buffer  sb =  new  StringBuffer ("KIIT  UNIVERSITY");
            // sb. insert ( 4 , " CSE");
           // sb. insert ( 0, 2025) ;
           // sb. insert ( 0, 'B');
             char  a[ ] =  { 'C' , 'S', 'E' } ;
              sb. insert  (0 , a) ;
              Sopln (sb);
           }
       }
```

// An interface can extends another interface

Case 2

```
interface A
{
  int a = 10;
  void show ( ) ;
 }
 interface B extends A
 {
  void display ( ) ;
 }
 class C implements B
 {
  public void show ( )
 {
   Sopln (" Value of a =" + a) ;
 }
  public void display ( )
 {
   Sopln (" I am the method of interface B") ;
 }
}
 class Main
 {
  public static void main ( String ar[ ])
  {
    C ob = new C ( ) ,
    ob. show ( ) ;
    ob. display ( ) ;
  }
}
```

Case 3 :- interface A

```
{
 int a = 10 ;
 void show ( ) ,
}
 interface B
 {
   void add (int a, int b) ;
   interface C extends A , B

                {
                  Sopln (" Value of a = "+a) ;
                }
                 public void add (int a, int b)
                 {
                  int c = a+b ;
                 Sopln (" Sum = "+c) ;
                }
                public void display ( )
                {
                 Sopln ("I am the method of class B")
                 }
                }
                class Main
                {
               public static void main (String ar[]) ;
                 {
                  D ob = new D ( ) ;
                  ob. show ( ) ;
                  ob. display ( ) ;
                  ob. add ( 10 , 12) ;
                 }
               }
```

Pg - 6

```
class SB6
 {
    psvm ( String args[])
     {
        StringBuffer sb= new StringBuffer ("KIIT java is my favourite language");
        Sopln ( sb.capacity()) ;
        // sb. append ("Hello");
        // Sopln (sb. capacity ());
        // sb. append ("Java is my favourite language");
         sb. append ("KIIT java is my favourite language");
       // sb. append ("KIIT Universt");
       // System. out. println (sb);
       // Sopln (sb.capacity ());
    }
 }
```

Prog 7

length () method

```
class SB7
 {
    psvm (string args[])
     {
        StringBuffer sb= new StringBuffer ("KIIT University");
        int p= sb. length() ;
        Sopln (p);
     }
 }
```

Prog 8

it is used to test at the specified position.

```
StringBuffer insert (int index , string str)
String Buffer insert (int index, char ch)
class SB8
 {
    psvm ( String ar[])
     {
        String Buffer sb = new StringBuffer ("KIIT UNIVERSITY");
         // sb. insert (4 , "CSE");
        // sb. insert ( 0, 2025);
        // sb. insert (0, 'B');
         char a[] = {'C', 'S', 'E'};
          sb. insert (0 ,a) ;
          Sopln (sb);
     }
 }
```

I10. java

* child class of an abstract class is abstract.

* We can take abstract variable of an abstract class.

```
abstract class A
{
   A()
   {
      Sopln (" I am constructor of abstract class A");
   }
      static void show ()
       {
          Sopln (" I have completed the show") ;
       }
      abstract void add (int a, int b);
  }
  abstract class B extends A
   {
      B();
       {
          Sopln ("I am constructor of abstract class B");
       }
   }
 class C extends B
   {
      B()
       {
          Sopln ("I am constructor of abstract class B");
       }
   }
 class C extends B
   {
     C( )
      {
          Sopln ("I am constructor of class C");
       }

          void add (int a, int b)
           {
              int c ;
              c= a+b;
              Sopln (" Sum = " + c);
           }
       }
      class Main
       {
           public static void main (String args[])
            {
               A ob = new C ( );
               A.show ( );
               ob. add (12, 13);
```

// Thread Creation

```
public void run ()
  {
     for ( int i = 1; i<=10, i++)
           Sopln

class MT4
  {
     psvm ( string ar[])
      {
         Five f = new Five();
         Seven s = new Seven ();
         Thread t1 = new Thread(f),
         Thread t2 = new Thread (s),
          t1. start ();
          t2 .start ();
       }
     }
```

Public final int getPriority ( ) method of Thread class returns priority of the given thread.

public final void selfpriority (int new priority) method changes the priority of thread to the value newpriority.

MT11. java

Thread Synchronisation :-

3 approaches :-

1) Synchronized block

→ it allows execution of arbitrary code to be synchronized on the lock of an arbitrary object

```
class Share extends Thread
 {
   static string msg[] = {"This", "is", "a", "synchronized", "variable"};
   Share (String threadname)
    {
      super (threadname);
```

abstract - keyword

```
  abstract class A
   {
      abstract void show ();
      abstract void add (int a, int b);
   }
  class B extends A
   {
     void show ()
      {
        Sopln (" I have completed the show");
      }
     void add (int a, int b)
      {
        int c;
        c = a+b;
        Sopln (" Sum = " + c);
    }  }
  class Main
   {
     public static void main (String args[])
      {
         B b = new B() ;
         b.show () ;
         b.add (12, 13) ;
      }
   }
```

I.10. java

* child class of an abstract class is abstract.

* We can take abstract variable of an abstract class.

```
abstract class A
{
    A()
    {
        Sopln (" I am constructor of abstract class A");
    }
        static void show ()
        {
            Sopln (" I have completed the show") ;
        }
        abstract void add (int a, int b);
    }
    abstract class B extends A
    {
        B();
        {
            Sopln ("I am constructor of abstract class B");
        }
    }
    class C extends B
    {
        B()
        {
            Sopln ("I am constructor of abstract class B");
        }
    }
    class C extends B
    {
        C( )
        {
            Sopln ("I am constructor of class C");
        }
            void add (int a, int b)
            {
                int c ;
                c= a+b;
                Sopln (" Sum = " + c);
            }
        }
        class Main
        {
            public static void main (String args[])
            {
                A ob = new C ();
                A.show ();
                ob. add (12,13);
```

replace ( ) - replaces the given String from the specified beginIndex & endIndex

```
class SB3
 {
   public static void main ( String args[])
    {
      String Buffer sb = new StringBuffer ("Hello");
      sb.Buffer ( 0,3 , "Java ");
      Sopln ( sb) ;
    }
  }
```

Prog-4
delete( )

```
class SB4
{
   psvm ( string args[ ])
    {
      StringBuffer sb= new StringBuffer ("Hello");
      sb.delete ( 1 , 3) ;                                   /* Output
      System.out.println ( sb) ;                                    Hlo
    }
 }
```

Prog 5
reverse ( ) - class reverse the current String.

```
class SB5
 {
     psvm ( string args[])
      {
        StringBuffer sb = new StringBuffer ("Hello");
        sb.reverse ( ) ;
        Sopln ( sb) ;
      }
   }
```

```
class Customer
 {
    int amount = 10000;
    synchronized void withdraw ( int amount)
      {
         Sopln ( " Going to withdraw"); 
         if ( this.amount < amount)
           {
              Sopln ( "Less Balance ; waiting for deposit ");
              try
               {
                   wait ( ) ;
                }
                catch ( Exception e) {   }
                }
                 this.amount = amount ;
                Sopln (" Withdraw completed");
              }
             synchronized void deposit ( int amount )
              {
                 Sopln (" Going to deposit ");
                 this.amount += amount ;
                 Sopln ("Deposit completed ");
               notify( ) ;
              }
           }
             class MT6
             {
                psvm (String ar[ ])
                  {
                      final Customer c = new Customer ( ) ;
                    }
             }
```