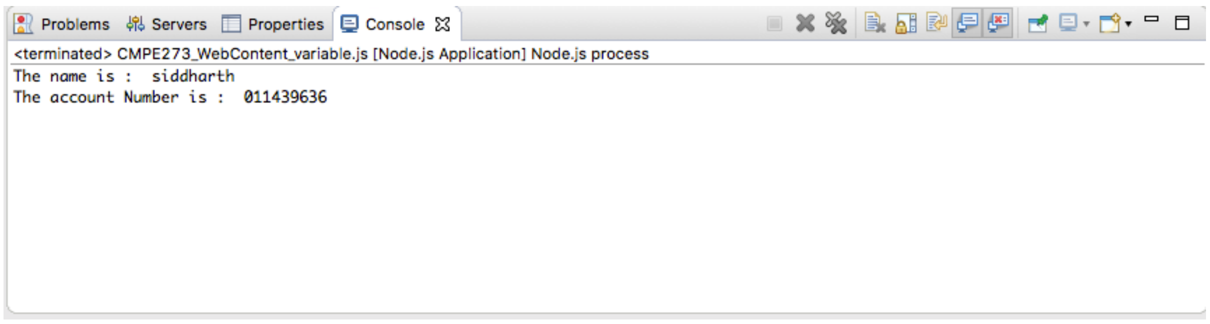
Assignment 1

Topic 1:

Variable – It is something that you use to store a value .

```
/*
 *
 *   Question : write a program to store a persons Name and and his Account-number ?
 *
 */


let name = "siddharth" ;
let Account_Number = "011439636" ;

console.log("The name is : " ,name );
console.log("The account Number is : " , Account_Number) ;
```



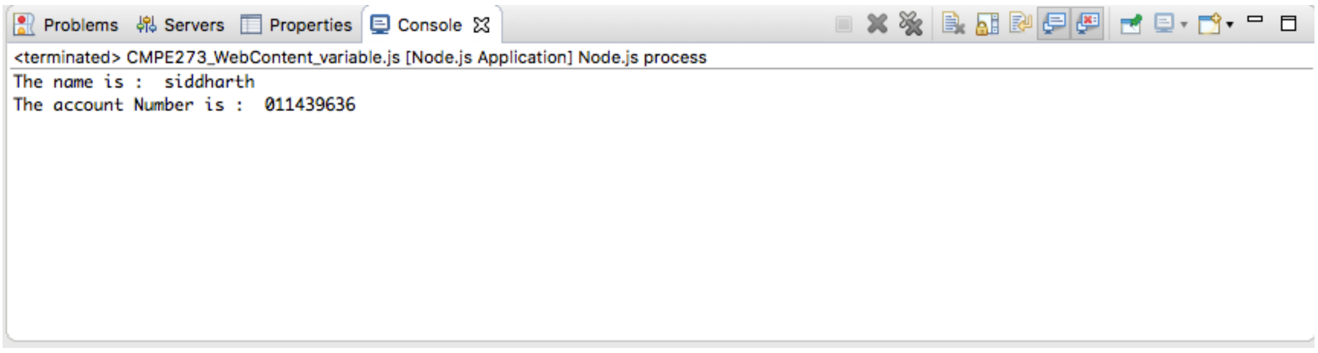
```
<terminated> CMPE273_WebContent_variable.js [Node.js Application] Node.js process
The name is :  siddharth
The account Number is :  011439636
```

Topic 2: Objects

Objects are real world representations, for example a car is an object , a person is an object. An object can store multiple variables of multiple data types. This is the biggest advantage of using an object.

```
/**
 *   Question : write a program to store information with a key value pair for example
store a persons detail in a single
 *   variable or object ;
 *   or better make the first topic(variable) more optimized by storing two different
variables into one object.
 */


let person = {name : "siddharth" , account : "011439636"}

console.log(person) ;
console.log("The persons name is :" , person.name);
console.log("The persons Accoount number is :" , person.account);
```

output:



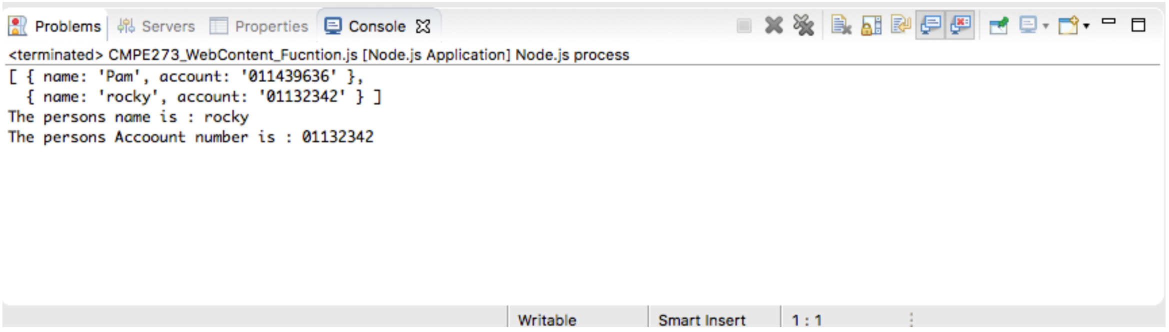
```
<terminated> CMPE273_WebContent_variable.js [Node.js Application] Node.js process
The name is :  siddharth
The account Number is :  011439636
```

Topic 3 : Functions

Functions are a set steps of that needs to be executed multiple number of times , so instead of writing that again and again we write a function and call it … that saves memory .

```
/**
 *  Question : Write a function to add a new person in the person object(previous topic) array .
 *
 */

 function addPerson(name , acc){
        count++ ;
        person[count] = {name : name , account : acc}
}

let count = 0 ;
let person = [];
person[count] = {name : "Pam" , account : "011439636"}
addPerson("rocky" , "01132342") ;


console.log(person) ;
console.log("The persons name is :" , person[count].name);
console.log("The persons Accoount number is :" , person[count].account);
```

output:



```
<terminated> CMPE273_WebContent_Fucntion.js [Node.js Application] Node.js process
[ { name: 'Pam', account: '011439636' },
  { name: 'rocky', account: '01132342' } ]
The persons name is : rocky
The persons Accoount number is : 01132342
```

Topic 4 : Events

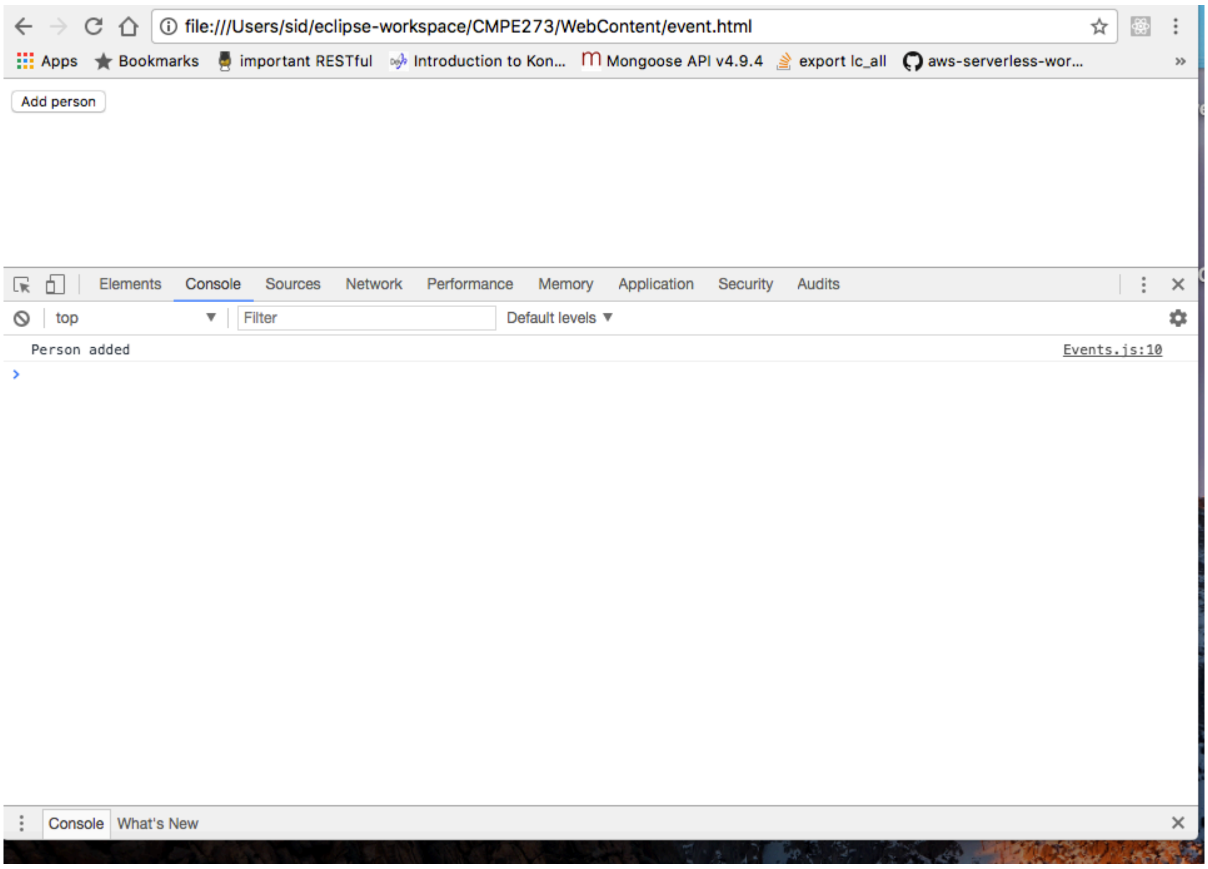
Events are basically something that changes the state , for example keyboard click , mouse hover ,mouse click , anything that changes can be considered as an event . they are widely used in the websites to keep the track of where the mouse pointer is and what is the keyboard inputing.

```
/**
 *  Question : Write a program to do something(call the function created in the previous
topic)
 *   when you click on a button.
 */


function addPerson(name , acc){
       //count++ ;
       person = {name : name , account : acc}
       console.log("Person added");

}
```

HTML CODE :

```
  DOCTYPE
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type = "text/javascript" src = "Events.js" >

</script>
</head>
<body>

<input type = "button" value  = "Add person"  onclick = "addPerson('rocky','01132342')">

</body>
</html>
```

Output:



```
Person added                                                    Events.js:10
>
```

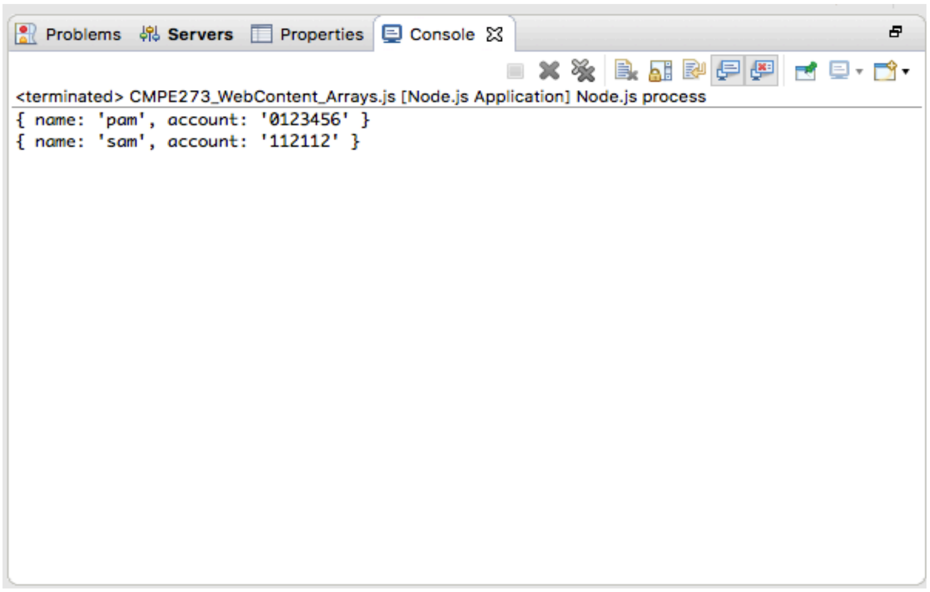
Topic 5 : Arrays

Arrays are a user defined data types that are used to store a collection of data in a single variable, for example in the current case I have used an array of the object person to store multiple persons.
Arrays are of various types , you can create an array on any primitive data type.

```
/**
 *  Question :Write a program to store name and account of multiple people. ( we can do
this by creating and array , so basically
 *  instead of creating multiple variables we just create one array)
 */



let person =[];  //here we have created an empty array of person
//let count = person.length ;

person[person.length] = {
              name : "pam" , account :"0123456"    // A person array with the 0th
position with name as pam and account number as 0123456
};
person[person.length] = {
              name : "sam" , account : "112112"
}
```

```
for(let i = 0 ; i< person.length ; i++)
       {
       console.log(person[i]);        // A simple for loop to display all the values of
the array person
       }
```

output :


```
<terminated> CMPE273_WebContent_Arrays.js [Node.js Application] Node.js process
{ name: 'pam', account: '0123456' }
{ name: 'sam', account: '112112' }
```

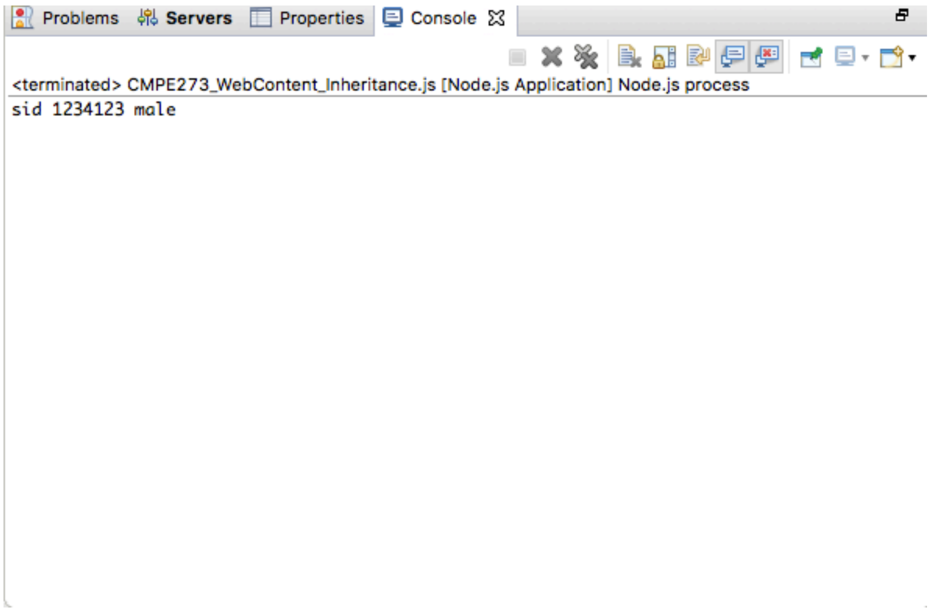
Topic 6 : Inheritance

Inheritance is the ability of a program or class or a function to “inherit” the properties of its parent class for example if there is a class or a function called Vehicle which has properties like wheels , engine etc etc and if we have a class called cars , and logically cars should also have all the properties same as vehicle then instead of doing all the work from scratch for cars we can just inherit the class vehicle which in turn will be able to use all the properties of vehicle in cars.

```
/**
 *   Question : write a program to add another object teacher which should have all the
properties of the object person created above.
 */

function Person(name , account) {
  this.name = name;
  this.account = account

//console.log(name , account)
  };


//Person("sid" , "1234123");

function Teacher(nme , acc , gen){
       Person.call(this ,nme , acc) ;   //instead of writing a code for assigining the
name and account individually we can do this..
                                                              // this could
also have been done  this.name = nme , this.account = acc but then imagine a situation
where
```

```
                                                                    // a lot of
things are needed to be done , so this is the reason why inheritance is needed
       this.gender = gen ;
       console.log(name,account,this.gender);
}
  Teacher("sid" , "1234123" , "male");
```

Output:



```
<terminated> CMPE273_WebContent_Inheritance.js [Node.js Application] Node.js process
sid 1234123 male
```

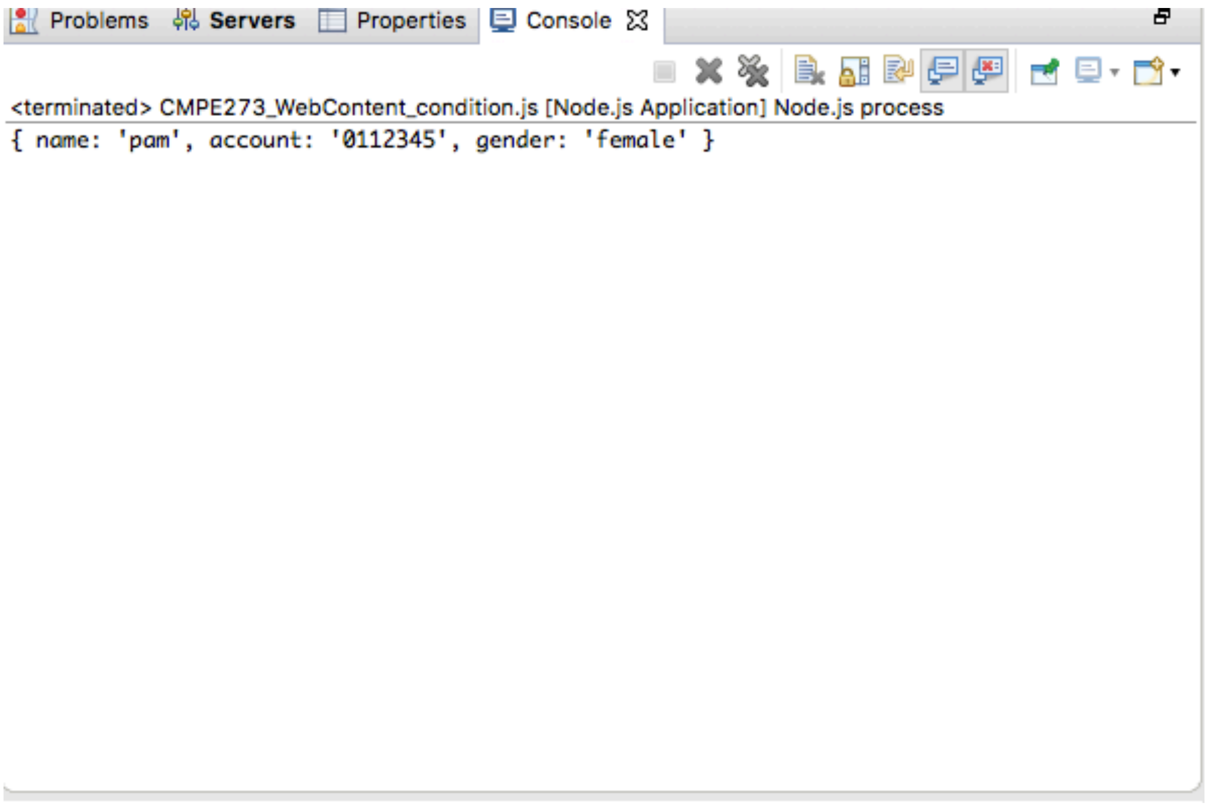
Topic 7 : Conditions

Conditions means exactly what it means in English , something that checks , basically something that helps us perform different things for different outcomes . ie "If this happens than do this . or else do this."

```
/**
 *  Question : Write a program to add the gender of a person as male or female according
to the variable.
 */


let person = [] ;

person[0] = {
              name : "pam",
         account : "0112345"
}
```

```
//console.log(person[0]) ;
let value = 1 ;
switch(value) {    //this is a typical example of a switch case where we can compare
more than one value , the same can be implemented using
                        //and if else conditions as well  like

case 0 :                                                                      /*
if(value ==0){  do something } elseif (value ==1) {do something else} else{do this} */
       person[0].gender = "male" ;
       break;

case 1 :
       person[0].gender = "female" ;
    break;

case 2 :
       person[0].gender = "other" ;
}

console.log(person[0]);
```

Output :



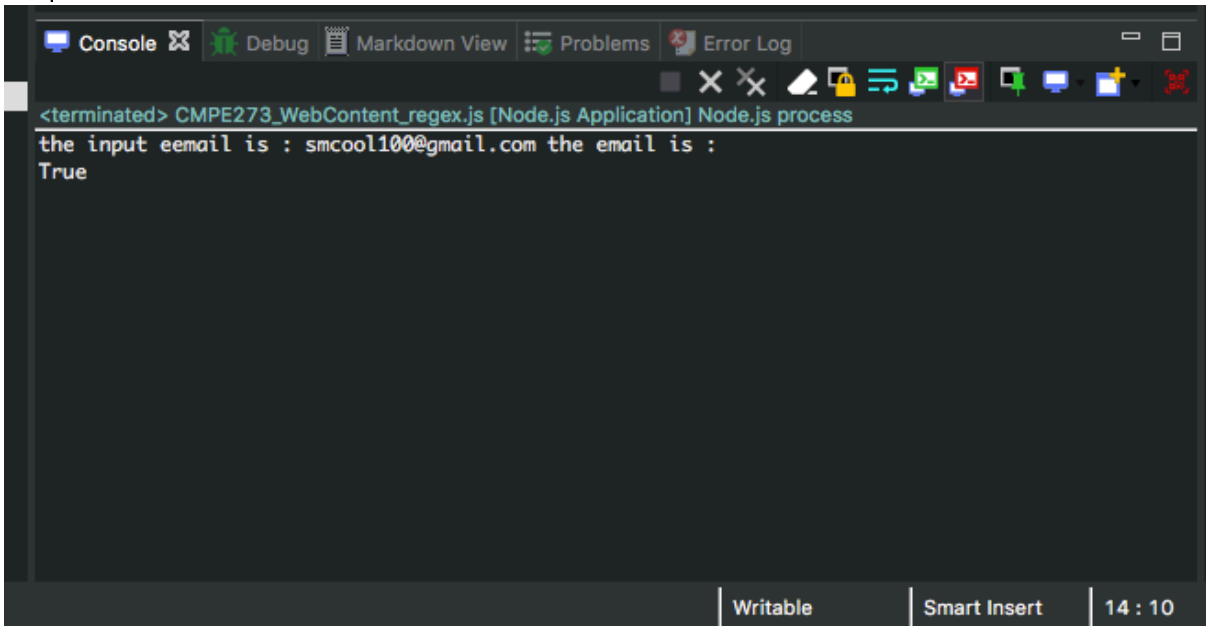
```
<terminated> CMPE273_WebContent_condition.js [Node.js Application] Node.js process
{ name: 'pam', account: '0112345', gender: 'female' }
```

Topic 8 : Regular Expressions.

Regular expressions are the objects that determine the pattern , for example it matches Whether the input matches the pattern that we want or not, for example
If we want an email address to have one '@' and only allowed special characters , then the easiest way to enforce that constraint is to use a regex.

```
/**
 *  Question : write a program to accept the email as an input in a specific format ie "
1 @ 1 . and .has to be there after @ and not right after it"
 *
 */
let pattern = /^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/;     //this is the
pattern that we need to watch and compare with our input to check whether the

                     //email is valid or not

let email = 'smcool100@gmail.com';

let n = email.match(pattern)[0];   //matches whether the string matches our pattern and
returns the matched string

console.log("the input eemail is :" , email , "the email is :" )
if(n===email){
       consoole.log("True");
}
else{
console.log("False");}
```

output :



```
<terminated> CMPE273_WebContent_regex.js [Node.js Application] Node.js process
the input eemail is : smcool100@gmail.com the email is :
True
```

Topic 9: Strict mode (I have newer version of eclipse (oxygen) , there is no strict mode everything is allowed in it.)

This is a mode in the javascript that help in the security and avoiding compile time errors. It is basically a way to introduce error checking in the code , when using strict mode we cannot use the implicitly declared variables , or assign a value to the read only property , or add a property to the object that is not extensible.

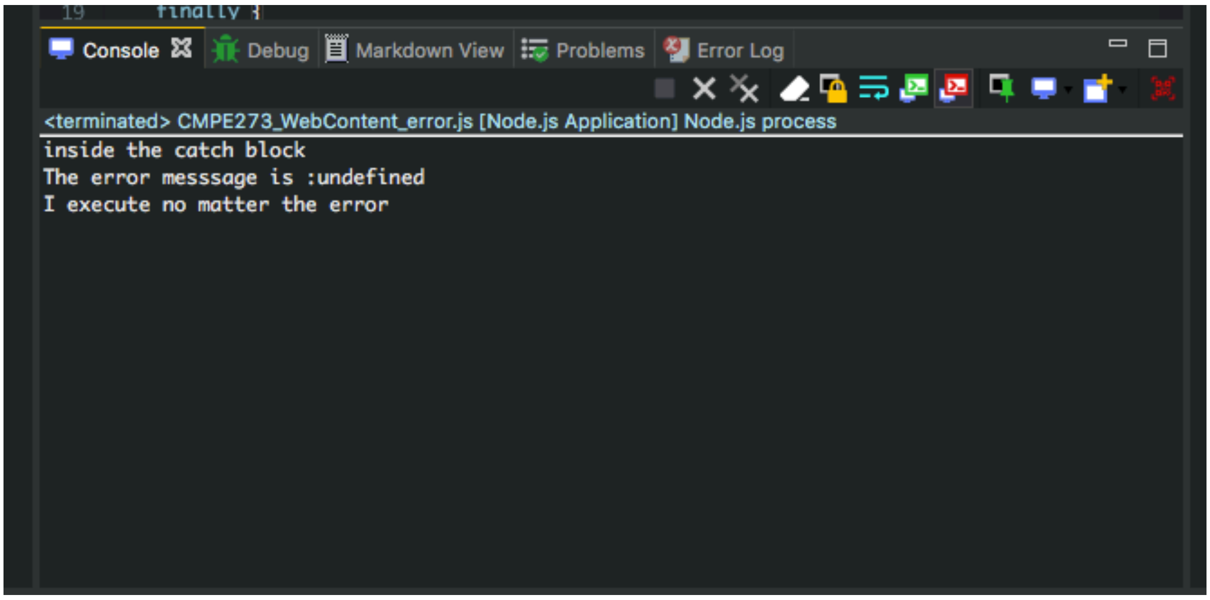
```
/**
 *  Question : write a program to show to difference between a strict mode and a regular
mode.
 */



num = 100 ;    //there will be no problem here

console.log(num);
p = {
              name  :" abc" ,
              address : "sanjose"
}                                                       //no problem here
"use strict" ;

num1 = 100;    // this will show error since we need to declare num1 first
console.log(num1)
p2 = {
              name  :" pqr" ,
              address : "sanjose"         //will not work , need to explicitly declare
the variable.
}
```

Topic 10: Errors

Errors play a very important role in a javascript , we can figure out the type of the error that we faced using the errors and exception handling. We can create our own error messages. There are two types of errors , compile time error and run time error. And we can catch errors with a try catch block.

```
/**
 * Question : Write a Program to show the working of a try catch and finally block
 */


 var  x;
     x = 11;
    try {
        if(x == "") throw "is empty";
        if(isNaN(x)) throw "is not a number";
        x = Number(x);
        if(x > 10) throw "is too high";
        if(x < 5) throw "is too low";
    }
    catch(err) {
        console.log("inside the catch block");
        console.log("The error messsage is :" + err.message);
    }
    finally {
       console.log("I execute no matter the error ")
    }
```

Output:



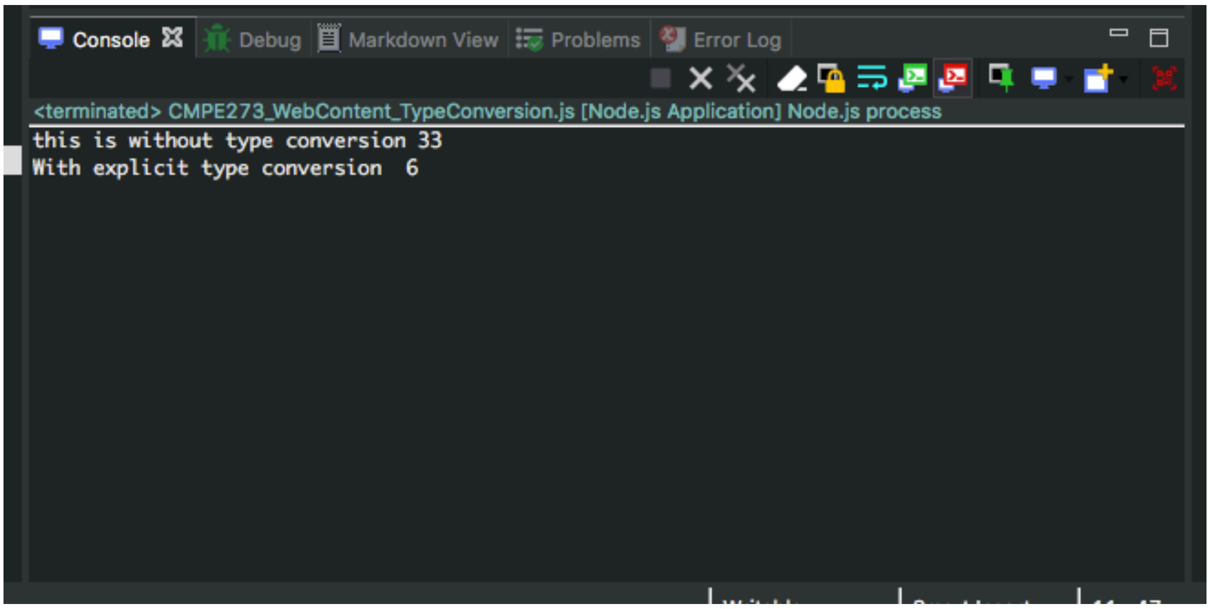
```
<terminated> CMPE273_WebContent_error.js [Node.js Application] Node.js process
inside the catch block
The error messsage is :undefined
I execute no matter the error
```

Topic 11: Type Conversion

Type conversion is basically converting a different type into the type that we want. There are two types of type conversion in java script. Implicit and explicit ,implicit does it automatically and explicit we have to explicitly convert it .

```
/**
 * Question : Write a Program to show the type Conversion
 */

let str = "3"
var num = 3 + str;
console.log(num);    //displays 33 since it converts it into a string
var num1 ;

 num1 = 3 + Number(str);
console.log(num1);    //displays 6 and not 33 , ie the type string has been converted
```

Output:



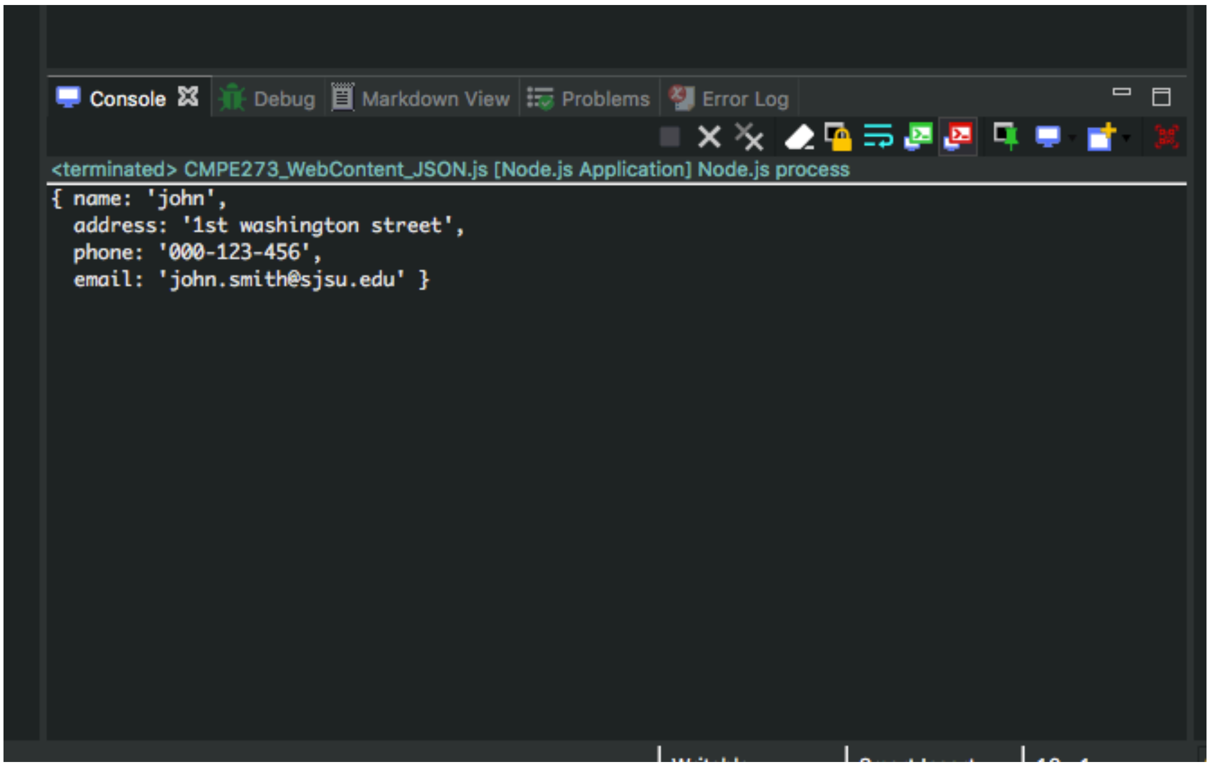
```
<terminated> CMPE273_WebContent_TypeConversion.js [Node.js Application] Node.js process
this is without type conversion 33
With explicit type conversion  6
```

Topic 12 : JSON

Json is basically Java script object notation , it is widely used to exchange data between server and a client. Json is a light weight data interchange format.

```
/**
 *  Question : Write a program to show how the json objects.
 */

let jsonobject = {
              name : "john" ,
              address : "1st washington street",
              phone : "000-123-456",
              email : "john.smith@sjsu.edu"
}



console.log(jsonobject);
```

Output:



```
<terminated> CMPE273_WebContent_JSON.js [Node.js Application] Node.js process
{ name: 'john',
  address: '1st washington street',
  phone: '000-123-456',
  email: 'john.smith@sjsu.edu' }
```

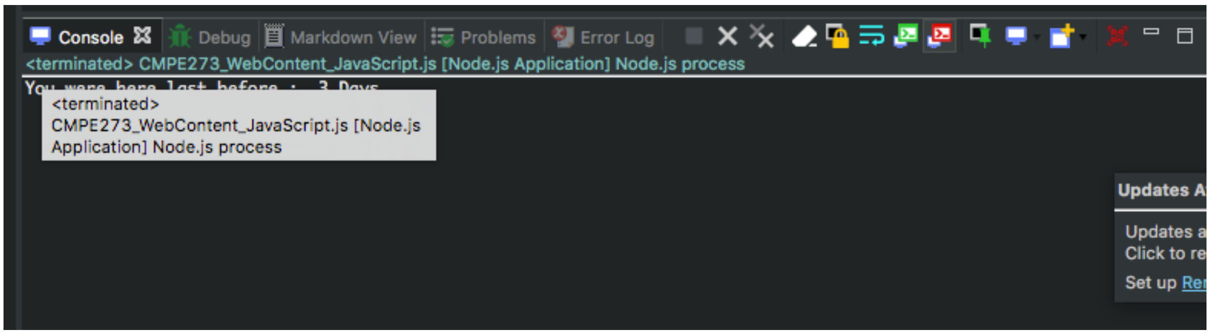
HTML 5 :

Topic 1 : Local Storage .

Local Storage stores the data locally instead of cookies , it stores the data that is more secure because the data is stored inside user’s local machine and the data stays stored inside the machine even if the user shuts down the machine , it will be stored unless the user explicitly removes the data.

```
<!-- Question : Write a program to show the user the last date he visited the page.
store the date in the
local storage and update it every time the user clicks the page -->



  DOCTYPE
<html>
<head>
<script>
function clickCounter() {
       if (localStorage.date){
       var d = new Date();
       localStorage.date = d;
       }
    if(typeof(Storage) !== "undefined") {
       document.getElementById("result").innerHTML = "You were here last before " +
(Date()-localStorage.date) ;
       localStorage.date = Date();
    console.log("You were here last before " , (Date()-localStorage.date))
    } else {
        document.getElementById("result").innerHTML = "Sorry, your browser does not
support web storage...";
    }
}
</script>
</head>
<body>
<p><button onclick="checkLastVisited()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the last visited date.</p>
<p>Close the browser tab (or window), and try again, and the counter is reset.</p>
</body>
</html>
```

Output :



```
<terminated> CMPE273_WebContent_JavaScript.js [Node.js Application] Node.js process
You were here last before :  3 Days
<terminated>
CMPE273_WebContent_JavaScript.js [Node.js
Application] Node.js process
```

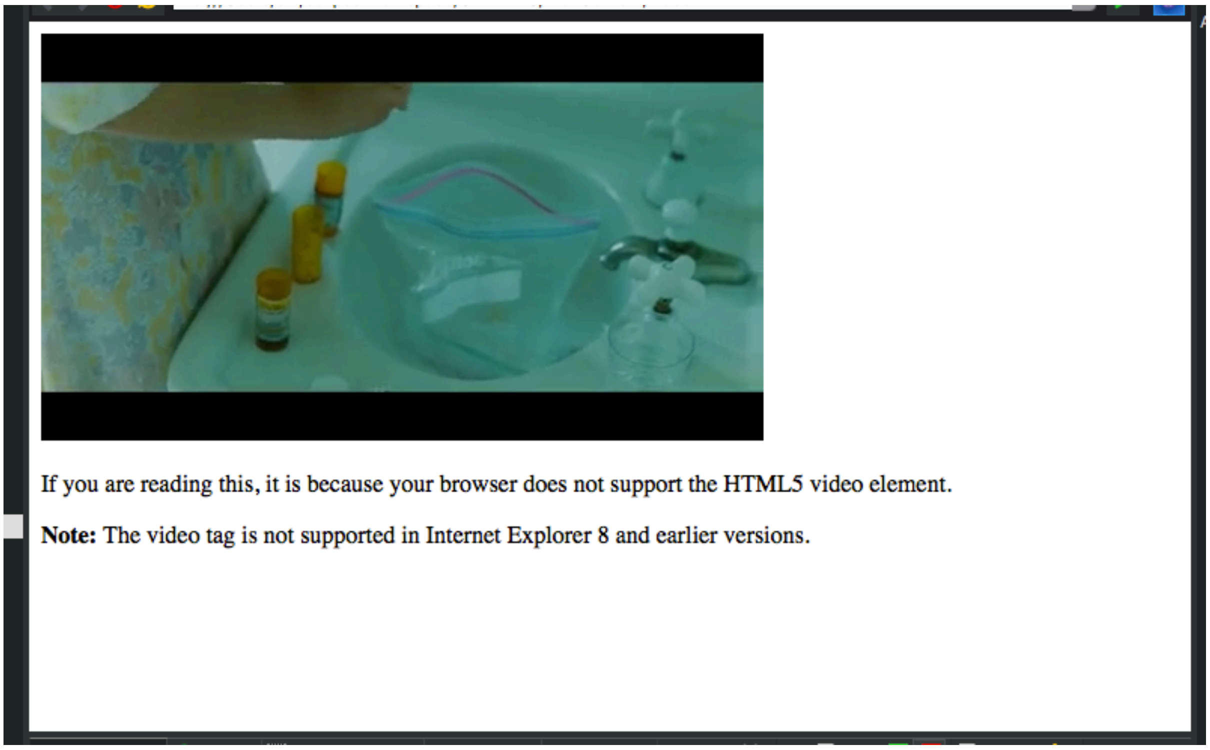
Topic 2 : Media (Video & Audio)

Video , it is a special tag in HTML 5 that helps us play a video on the webpage , it is just like an image tag , we need to provide the source of the video and other tags inside , like autoplay and controls(this shows the controls in the video)

```
<!-- Write a program to show how the video tag works and other tags present in the tag
like controls ,and
autoplay. -->



  DOCTYPE
<html>
<body>

<video src="./mov.mp4" controls>
</video>
<p>If you are reading this, it is because your browser does not support the HTML5 video
element.</p>
```

```
<p><strong>Note:</strong> The video tag is not supported in Internet Explorer 8 and
earlier versions.</p>

</body>
</html>
```

Output:



If you are reading this, it is because your browser does not support the HTML5 video element.

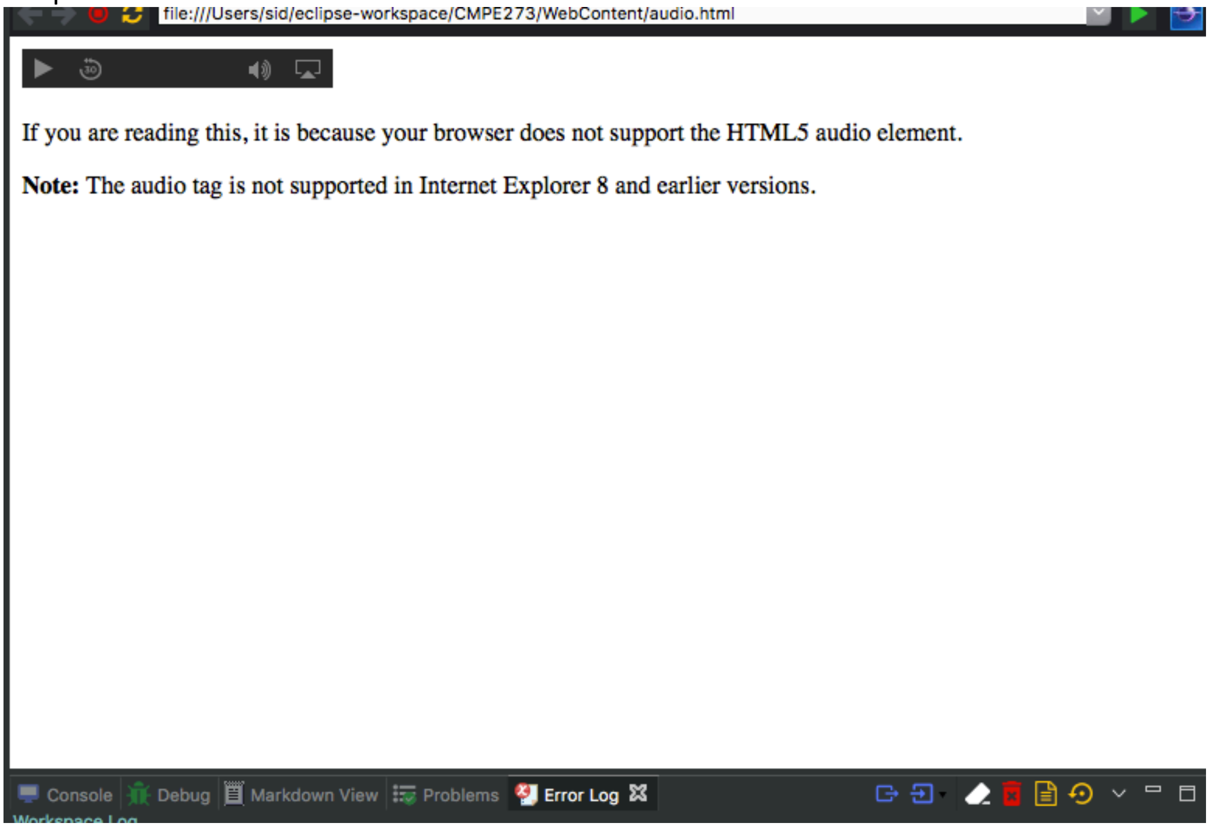
Note: The video tag is not supported in Internet Explorer 8 and earlier versions.

Audio , It is also a tag like video and an image tag that plays the audio on the webpage , it supports all the types of audio files , html 5 supports the almost all the types of audio files.

```
<!-- Write a program to show how the audio tag works and other tags present in the tag
like controls ,and
autoplay. -->



  DOCTYPE
<html>
<body>

<audio src="./audio.mp3" controls>  </audio>
<p>If you are reading this, it is because your browser does not support the HTML5 audio
element.</p>
<p><strong>Note:</strong> The audio tag is not supported in Internet Explorer 8 and
earlier versions.</p>

</body>
```

```
</html>
```

Output :


file:///Users/sid/eclipse-workspace/CMPE273/WebContent/audio.html

If you are reading this, it is because your browser does not support the HTML5 audio element.

Note: The audio tag is not supported in Internet Explorer 8 and earlier versions.

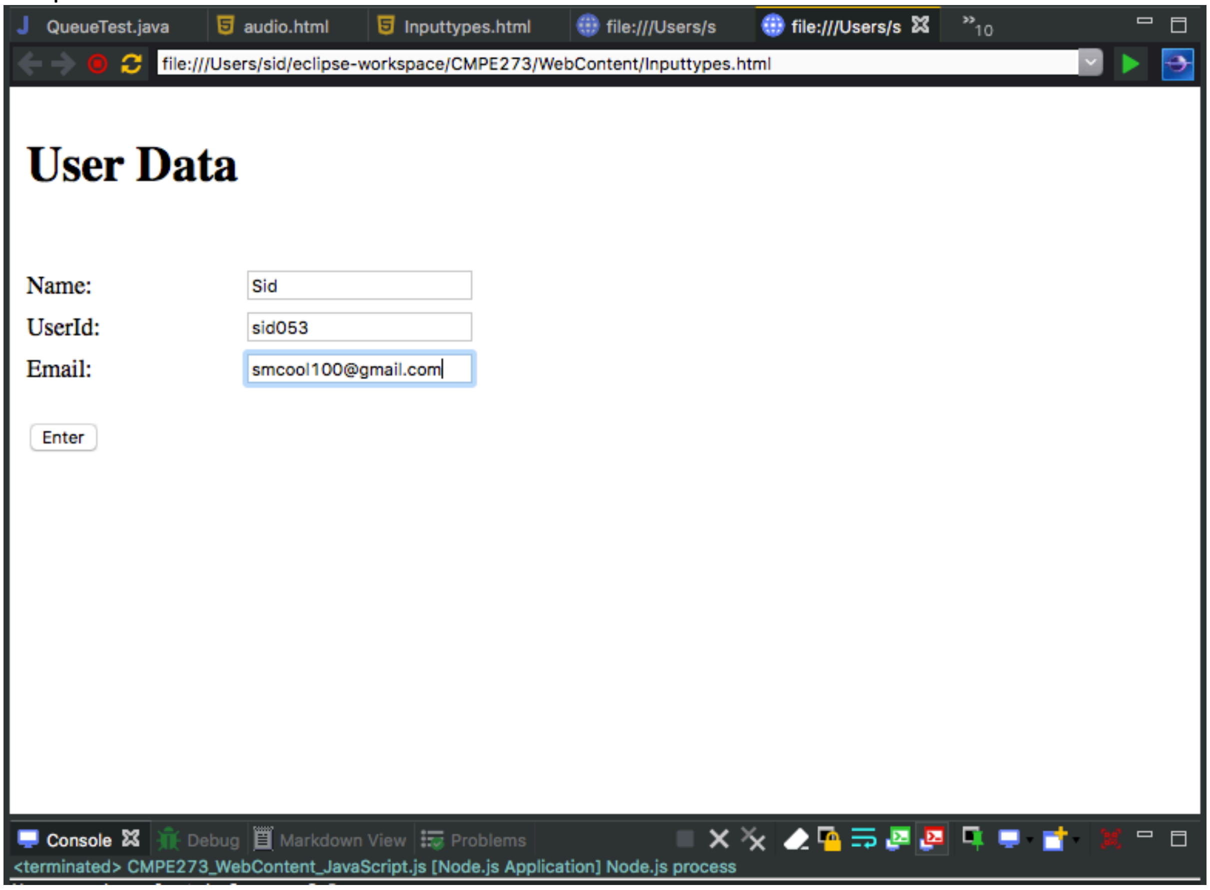
Topic 3 : Input types.

Input types are the different types of input tags that HTML 5 has , they are used to take inputs from the user.
Various types of input :
1)Text – usually used to take any kind of text as an input
2)required – it specifies that it should not be left empty
3)autofocus – brings the focus of the cursor where it is mentioned
4)email – defines a field for email
5)pattern – it is basically used to match the pattern of the input , as in this case used to match the pattern for the email , so that user cannot enter the wrong pattern.

```
  DOCTYPE
<html>
<head>
    <script type="text/javascript">

    function save(){
        document.getElementById("input").innerHTML="Entered Successfully!!";
    }
    </script>
```

```
</head>
<body>

<form>
<table>
<tr><td><h1><b>User Data</b></h1></td></tr>
  <tr><td><div id="input"></div><br></td></tr>
  <tr><td>Name:</td> <td><input type="text" name="fname" autofocus required></td></tr>
  <tr><td> UserId: </td> <td><input type="text" name="userid" required pattern="[a-z0-
9]*"></td></tr>
  <tr><td> Email: </td> <td><input type="email" name="email" required pattern="/^([a-z0-
9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/"></td></tr>


  <tr><td><br><input type="button" value="Enter" onclick="save()"></td>


  </table>

</form>

</body>
</html>
```

Output :



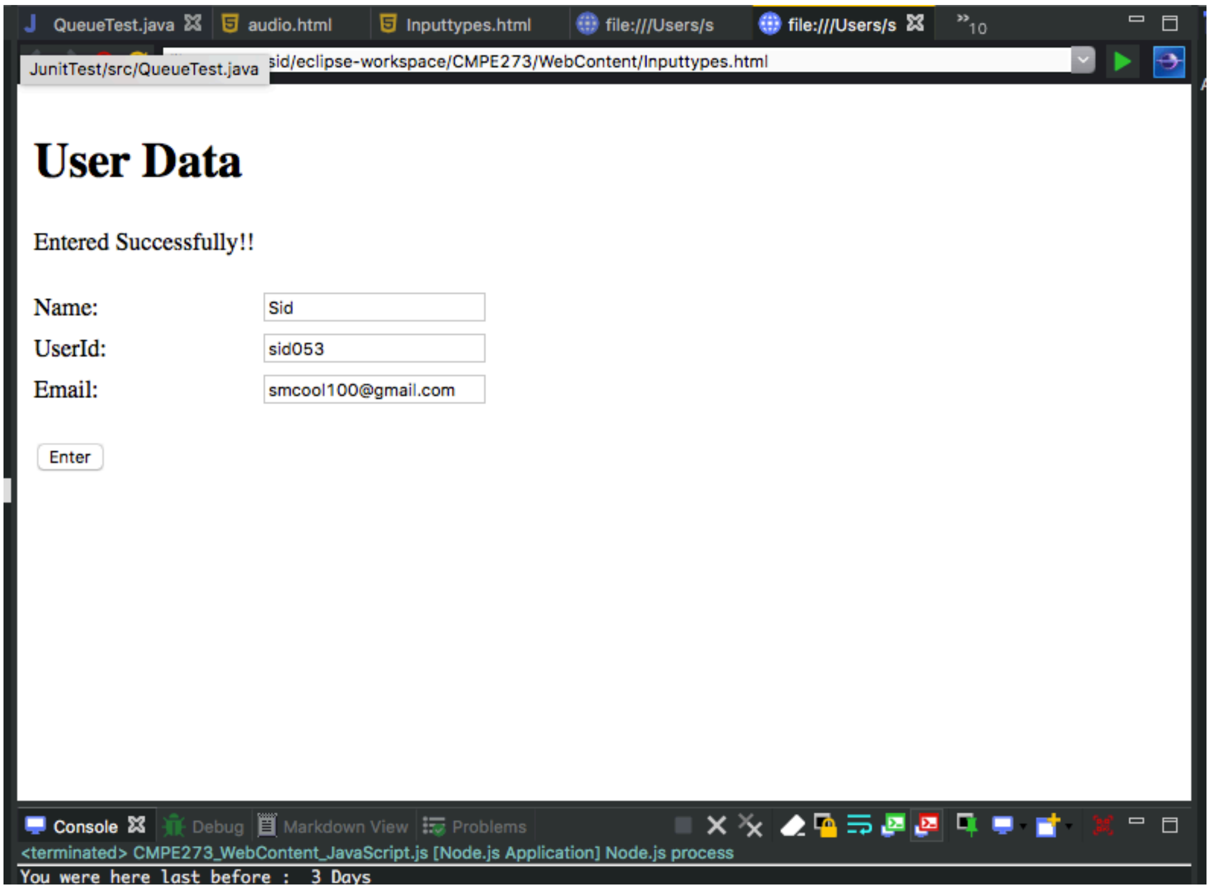
file:///Users/sid/eclipse-workspace/CMPE273/WebContent/Inputtypes.html

User Data

Name: Sid
UserId: sid053
Email: smcool100@gmail.com

Enter

<terminated> CMPE273_WebContent_JavaScript.js [Node.js Application] Node.js process



User Data

Entered Successfully!!

Name: Sid

UserId: sid053

Email: smcool100@gmail.com

Enter

```
<terminated> CMPE273_WebContent_JavaScript.js [Node.js Application] Node.js process
You were here last before :  3 Days
```

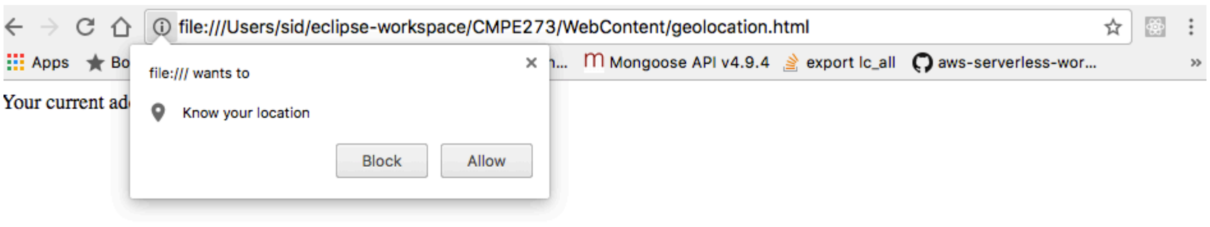
Topic 4 : Geo location

It is a special feature in html 5 that helps us in determining the latitude and longitude of our ip address , so basically we can determine our location using geolocation.

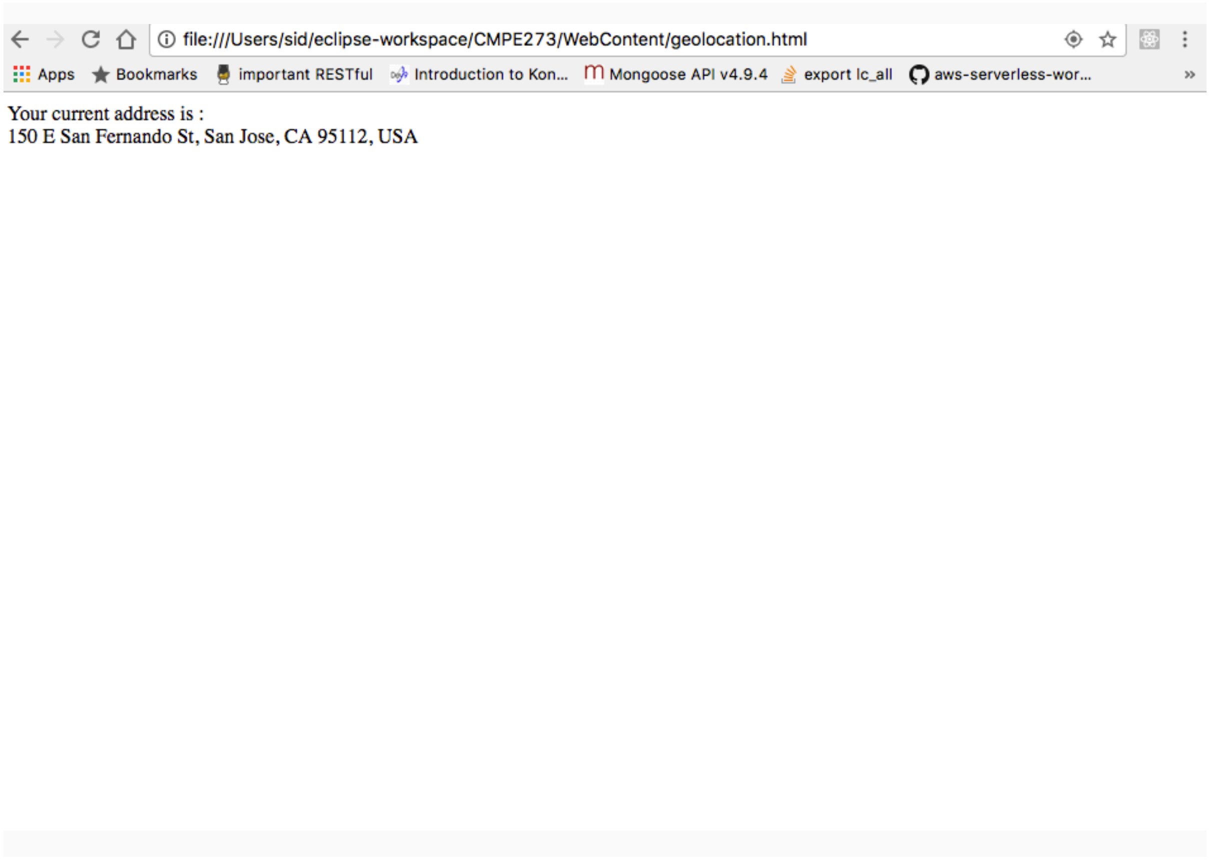
```
  DOCTYPE
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no"/>
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title>GeoLocation</title>

<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
<script type="text/javascript">
  var geocoder;

  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(successFunction, errorFunction);
}

function successFunction(position) {
    var lat = position.coords.latitude;
    var lng = position.coords.longitude;
    codeLatLng(lat, lng)
}
```

```
function errorFunction(){
    alert("Geocoder failed");
}

  function initialize() {
    geocoder = new google.maps.Geocoder();



  }

  function codeLatLng(lat, lng) {

    var latlng = new google.maps.LatLng(lat, lng);
    geocoder.geocode({'latLng': latlng}, function(results, status) {
      if (status == google.maps.GeocoderStatus.OK) {
      console.log(results)
        if (results[1]) {

        document.getElementById("result").innerHTML = results[0].formatted_address;

             for (var i=0; i<results[0].address_components.length; i++) {
            for (var b=0;b<results[0].address_components[i].types.length;b++) {


                if (results[0].address_components[i].types[b] ==
"administrative_area_level_1") {

                    city= results[0].address_components[i];
                    break;
                }
            }
        }
           document.getElementById("result1").innerHTML = city.short_name;



        } else {
          alert("No results found");
        }
      } else {
        alert("Geocoder failed due to: " + status);
      }
    });
  }
</script>
</head>
<body onload="initialize()">
Your current address is :
<div id="result"></div>



</body>
</html>
```

Output:



Your current ad



Your current address is :
150 E San Fernando St, San Jose, CA 95112, USA

JAVA

Topic 1 : Queue

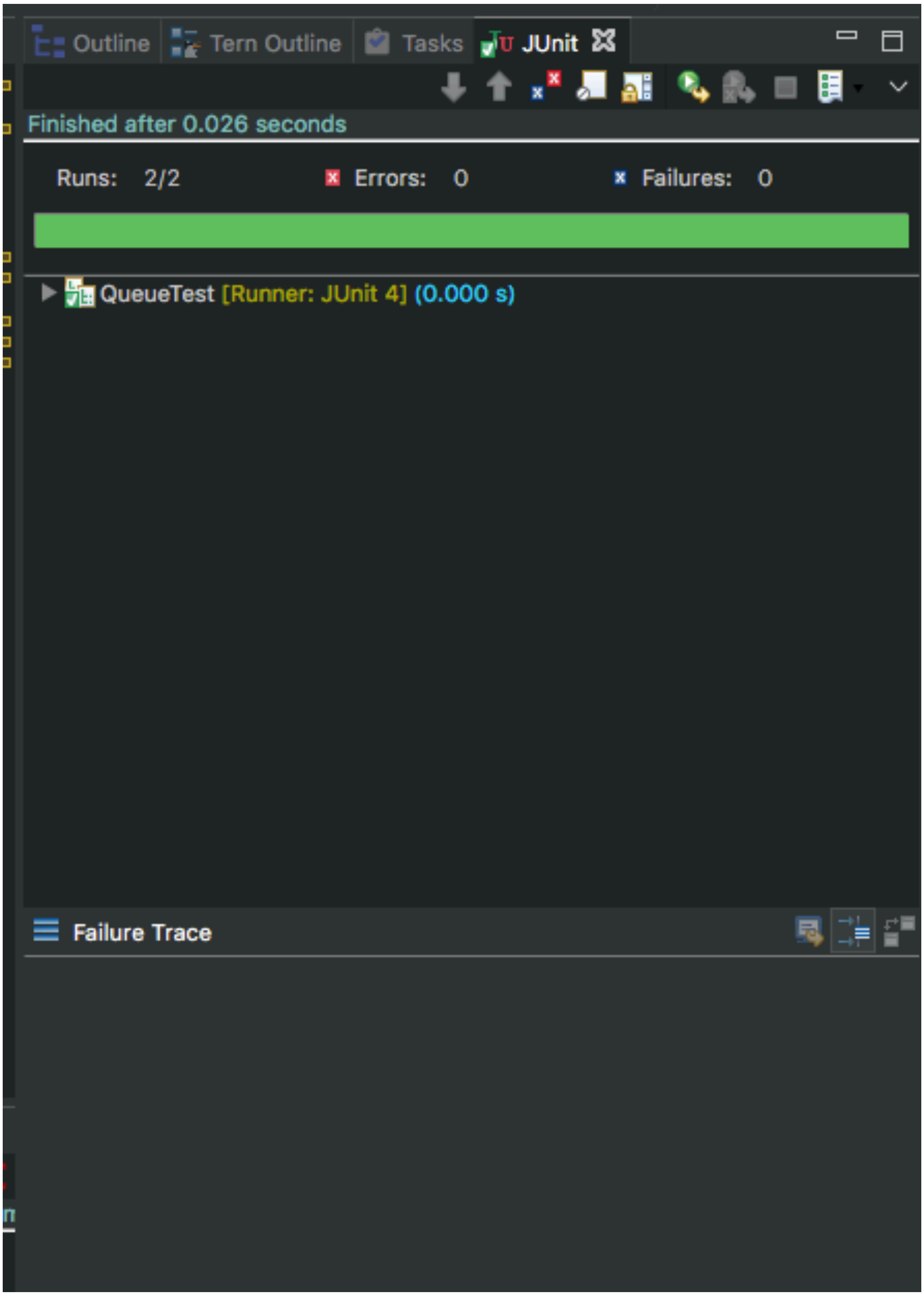
Queue is a user defined data type that works on first come first serve basis , it is like a simple queue. The first person to enter leaves first.

```
import java.util.Queue;

/*
 * Question : write a program to implement a queue and test it using JUnit
 *
 */


public class queueClass {
       int CustomerId ;
       String customername;
              boolean serviced;

       public queueClass(int CustomerId ,String customerName) {

              this.CustomerId = CustomerId ;
              this.customername = customerName ;
              this.serviced = false;

       }

       public queueClass service(Queue<queueClass> q) {

         queueClass obj = q.poll();
              obj.serviced = true;
              return obj;

       }

}
```

JUNIT :

```
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

package javapracticetest;

import static org.junit.Assert.*;

import java.util.LinkedList;
import java.util.Queue;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import javapractice.RestaurantQueue;

public class QueueTest {

    queueClass r;

    queueClass o1= new           (1, "Sam");
    queueClass o2= new           (2, "Kit");
    queueClass o3= new           (3, "Ron");

    Queue<queueClass> queue=new           <>();


    public void setUp() throws Exception {

        r=new           (4,"siddharth");



        queue.add(o1);
        queue.add(o2);
        queue.add(o3);

    }


    public void tearDown() throws Exception {
    }


    public void restaurantQueueTest() {


        assertEquals(o1, r.       (queue));
        assertEquals(o2, r.       (queue));
        assertEquals(o3, r.       (queue));
    }

}
```

Output:



```
Finished after 0.026 seconds

Runs:   2/2        Errors:   0        Failures:   0

QueueTest [Runner: JUnit 4] (0.000 s)
```

Topic 2 : Stack

Stack is a user defined data type that works on first come last served, it works as a stack in English , like a stack of plates , the last plate gets picked up first.

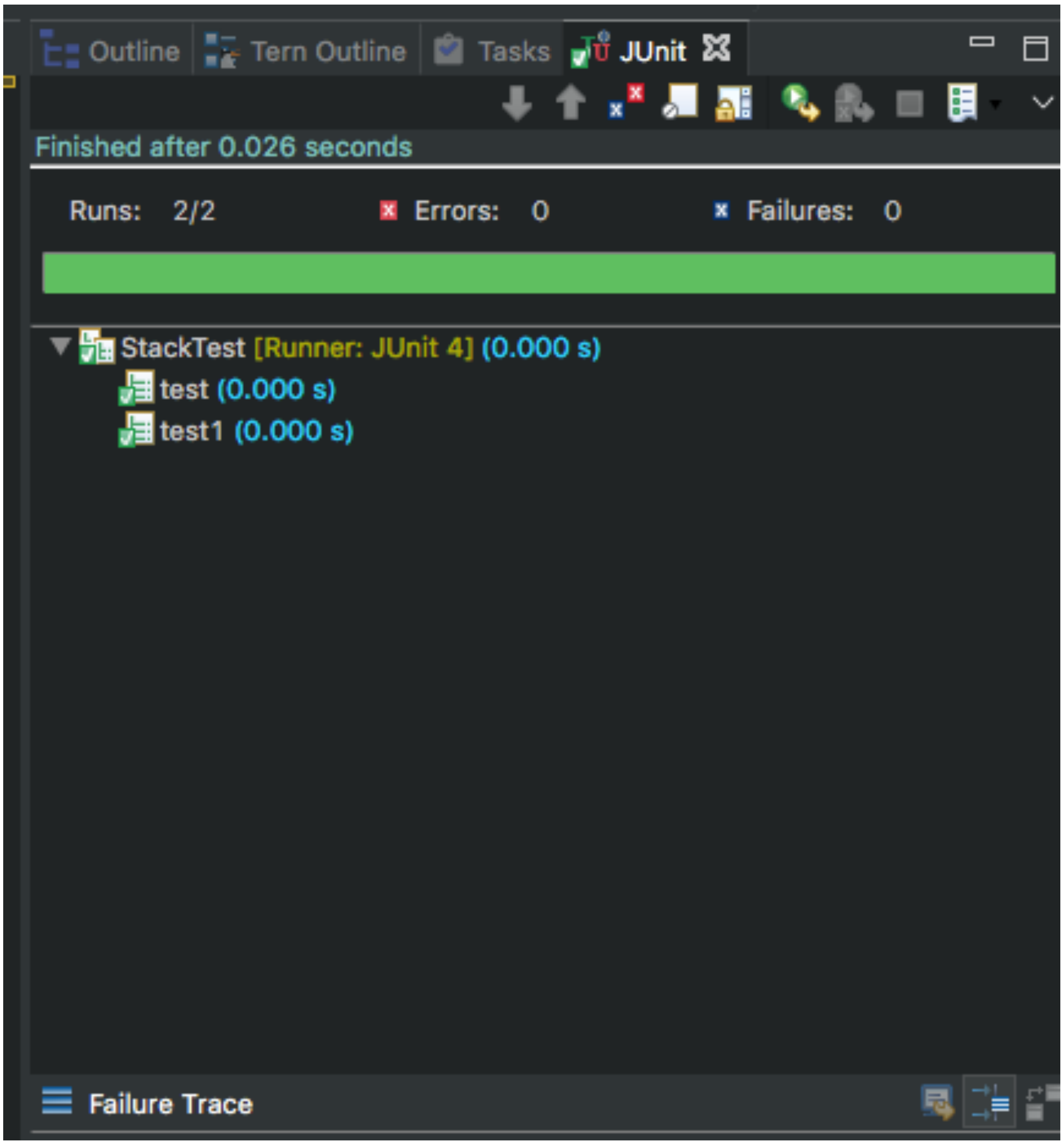
```
import java.util.Stack;

/* Question : write a program to test the stack functionality by reversing
 * the input number.
 */
public class stackClass {

       int input ;
         Stack<Integer>  stk = new       <Integer>();

   public stackClass(int inp) {
           int inp1;
         this.input = inp;

         while(inp>0) {

                inp1 = inp%10 ;
                inp = inp/10 ;
                stk.     (inp1);
                  // System.out.println(inp1);
         }

     }
   public String output(stackClass s) {
         String str = "";
       while(!stk.        ()) {
         str = stk.    () + str ;
              System.out.          (str);
   }
   return str;

}
}
```

JUNIT class :

```
import static org.junit.Assert.*;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

public class StackTest {

       stackClass stk,stk1;



       @Before
       public void setUp() throws Exception {
```

```
                stk = new stackClass(12345);
                stk1 = new stackClass(123);
        }

        @After
        public void tearDown() throws Exception {
        }

        @Test
        public void test() {


                assertEquals(stk.output(stk),"54321");
        }

        @Test
public void test1() {


                assertEquals(stk1.output(stk),"321");
        }

}
```

OutPut :



```
Finished after 0.026 seconds

Runs:   2/2          Errors:   0          Failures:   0

StackTest [Runner: JUnit 4] (0.000 s)
    test (0.000 s)
    test1 (0.000 s)
```

Topic 3 : Arrays

Arrays is a user defined data type where the data is stored in a sequential way , arrays can be multi dimensional .

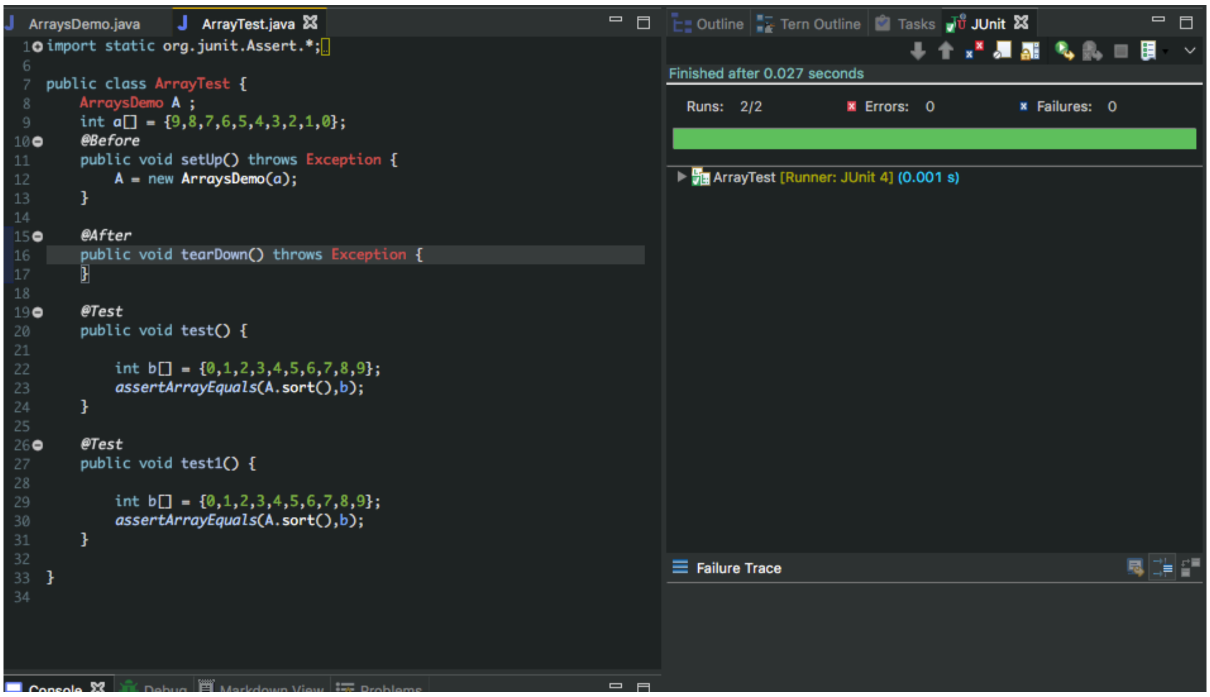
```
//Write  a program to show the working of an array using a sorting tecnique
public class ArraysDemo {

	int arr[] = new int[10];

	public ArraysDemo(int a[]) {

	for(int i = 0 ; i <10 ; i++) {
		arr[i] = a[i];
	}
```

```
    }

    public int[] sort() {
            int n = arr.length;
             for (int i = 0; i < n-1; i++)
                 {for (int j = 0; j < n-i-1; j++)
                     { if (arr[j] > arr[j+1])
                       {
                         // swap temp and arr[i]
                         int temp = arr[j];
                         arr[j] = arr[j+1];
                         arr[j+1] = temp;
                       }
                     }
                 }
             System.out.        (arr);
             return arr;
    }


}
```

JUNIT :

```
import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class ArrayTest {
   ArraysDemo A ;
   int a[] = {9,8,7,6,5,4,3,2,1,0};
          @Before
          public void setUp() throws Exception {
                    A = new ArraysDemo(a);
          }

          @After
          public void tearDown() throws Exception {
          }

          @Test
          public void test() {

                    int b[] = {0,1,2,3,4,5,6,7,8,9};
                    assertArrayEquals(A.sort(),b);
          }

          @Test
          public void test1() {
                int b[] = {0,1,2,3,4,5,6,7,8,9};
                assertArrayEquals(A.sort(),b);
        }

}
```

Output :



```
import static org.junit.Assert.*;

public class ArrayTest {
    ArraysDemo A ;
    int a[] = {9,8,7,6,5,4,3,2,1,0};
    @Before
    public void setUp() throws Exception {
        A = new ArraysDemo(a);
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void test() {

        int b[] = {0,1,2,3,4,5,6,7,8,9};
        assertArrayEquals(A.sort(),b);
    }

    @Test
    public void test1() {

        int b[] = {0,1,2,3,4,5,6,7,8,9};
        assertArrayEquals(A.sort(),b);
    }

}
```

```
Finished after 0.027 seconds
Runs:  2/2        Errors:  0        Failures:  0

ArrayTest [Runner: JUnit 4] (0.001 s)

Failure Trace
```

Topic 4: Interfaces

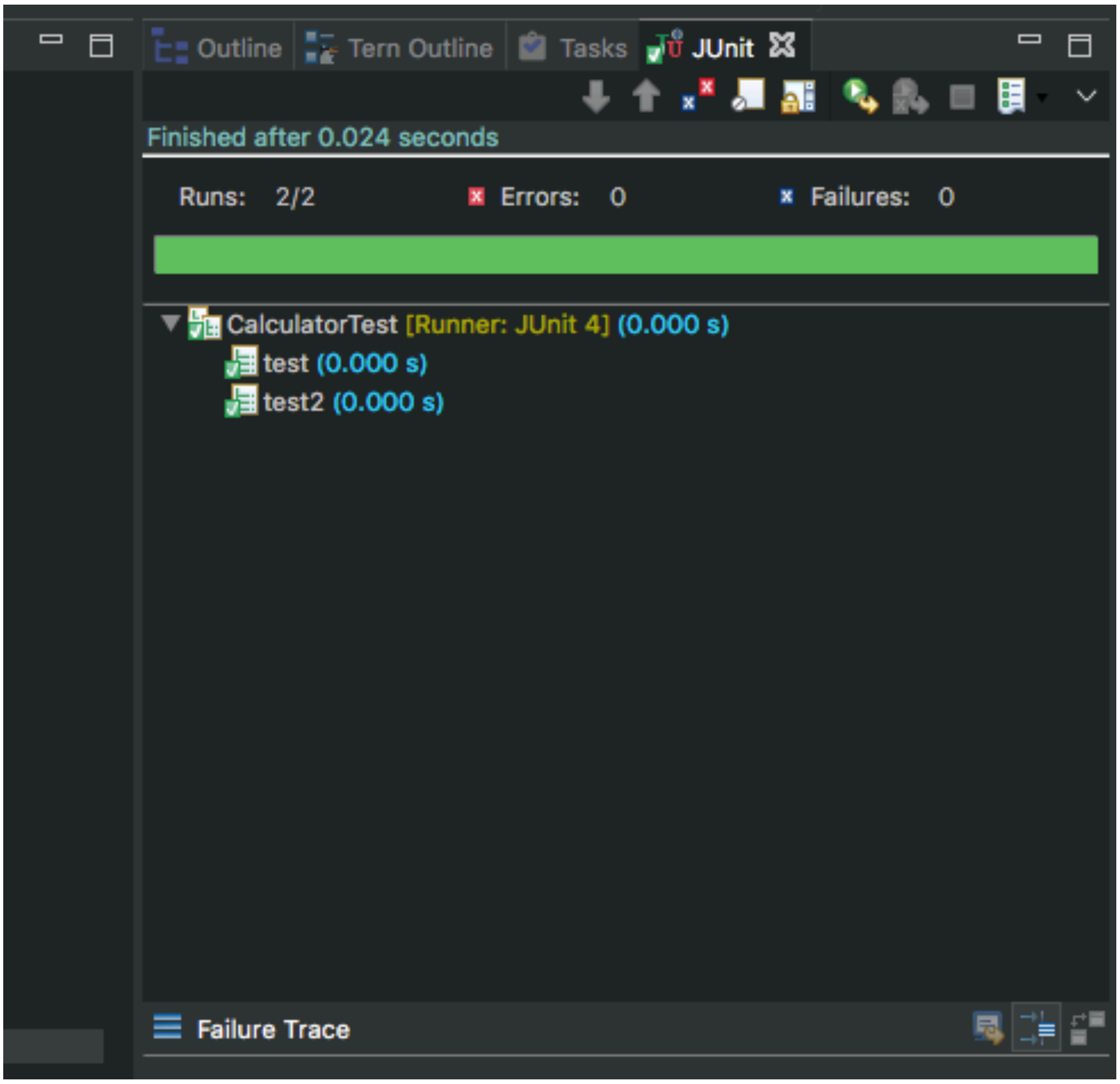
It's a special type of class where we donot define a method , we just declare the method ,definition is done in the classes that implement the method , one class can implement multiple interfaces , it was invented to solve the multiple inheritance problem.

```
public class Calculator implements InterfaceDemo{

public int add(int a , int b) {

                return a+b;
        }

public int sub(int a , int b) {

                return a-b;
        }

public int multiply(int a , int b) {
        return a*b;
}

public static void main(String[] args) {

Calculator c = new            ();
c.  (10, 15);
c.  (20, 5);
c.       (10, 3);

}

}
```

JUNIT :

```
import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class CalculatorTest {
    Calculator calc;
        @Before
        public void setUp() throws Exception {
                calc = new Calculator();
        }

        @After
        public void tearDown() throws Exception {
        }

        @Test
        public void test() {
                assertEquals(calc.add(10,5),15);
                assertEquals(calc.sub(10,5),5);
                assertEquals(calc.multiply(10,5),50);

        }
        @Test
        public void test2() {
                assertEquals(calc.add(11,5),16);
                assertEquals(calc.sub(11,5),6);
                assertEquals(calc.multiply(11,5),55);

        }
}
```

Output :


Outline   Tern Outline   Tasks   JUnit

```
Finished after 0.024 seconds

Runs:   2/2        Errors:   0        Failures:   0

▼ CalculatorTest [Runner: JUnit 4] (0.000 s)
    test (0.000 s)
    test2 (0.000 s)
```

Failure Trace

Topic 5 : Collections

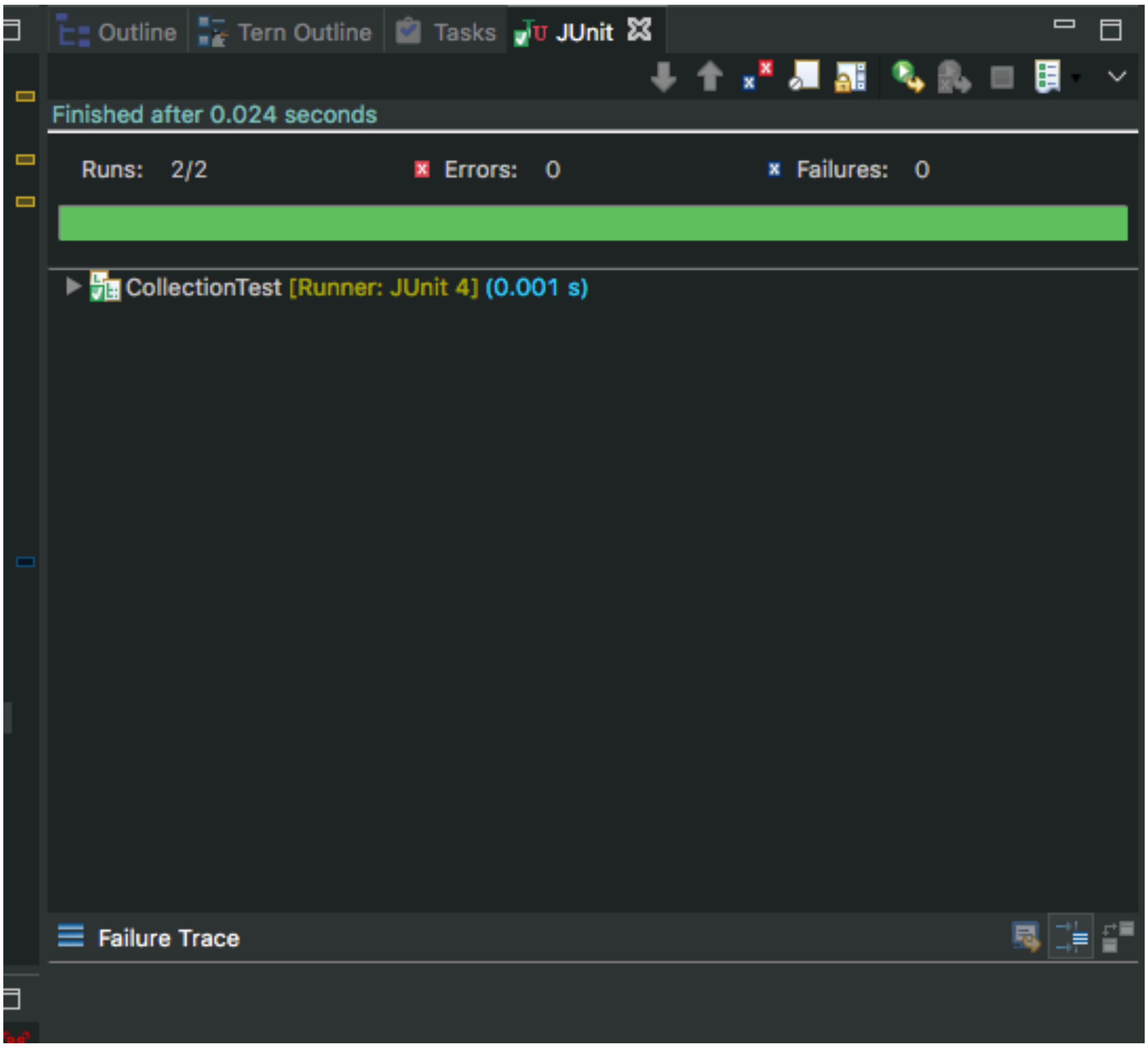
Collection is basically an object that stores a references to other objects , they are basically in the package java.util , java collection framework provides multiple interfaces.

```
//Question : write a program to return the persons name from the person object on the
// basis of the account number passed

import java.util.HashMap;


public class Collection {


        public String get(int acc , HashMap<Integer,Person> hm) {
            if(hm.             (acc)) {
                Person p = hm.   (acc);

                System.out.       (p.name);
                return p.name;


            }
            else {
                   return null;
            }



        }
}
```

JUNIT :

```
import static org.junit.Assert.*;

import java.util.HashMap;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

public class CollectionTest {
       HashMap<Integer,Person> hs = new        <Integer,Person>();
  //add elements to HashSet
       Person p,p1,p2 ;
       Collection c1 ;




       public void setUp() throws Exception {
                p = new       (102040 , "Pam");
                p1 = new       (102140 , "sam");
                p2 = new       (102540 , "dam");
                hs.   (1, p);
                hs.   (2, p1);
                hs.   (3, p2);
                c1 = new           ();

       }


       public void tearDown() throws Exception {
       }


       public void test() {
       assertEquals(c1.   (102140, hs),"sam");
       }

}
```

Output:



```
Finished after 0.024 seconds
Runs:   2/2        Errors:   0        Failures:   0
▶ CollectionTest [Runner: JUnit 4] (0.001 s)
```

Topic 6 : Generics

They are basically used to provide compile time safety , Generics are a facility to generic programming. Basically the programmer can specify what type of input can be passed , for example one can limit the input type in the list with generics , ie the programmer can mention that the list will only take input as a certain class objects.

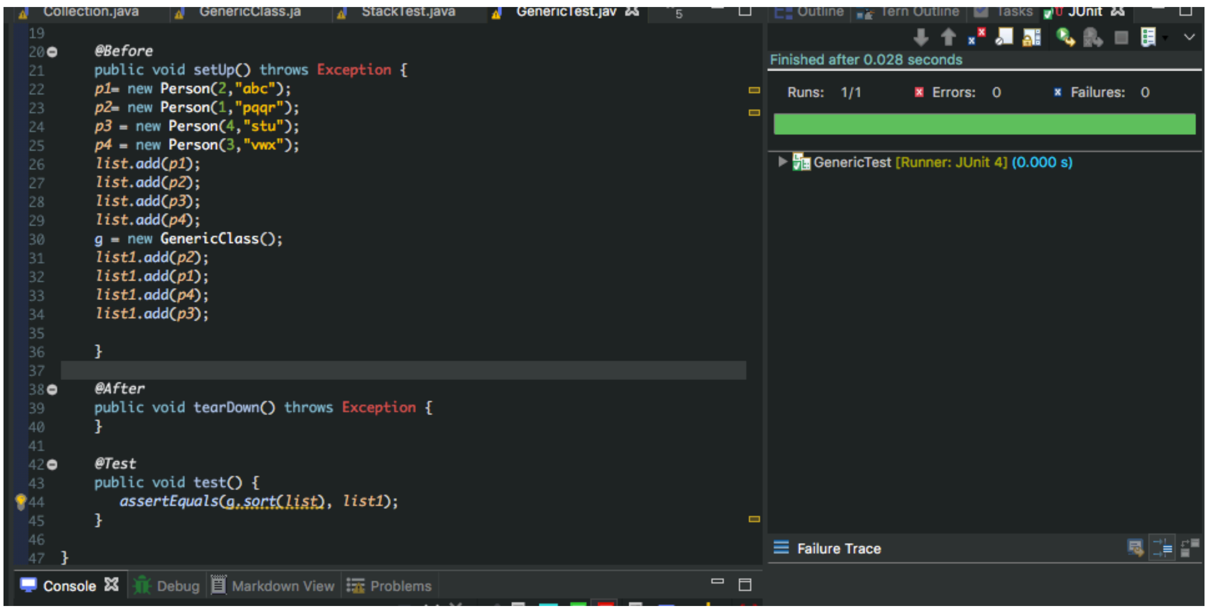
```
// Question  : write a program to sort the person objects inside a list on the
// basis of the account number.


import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
```

```
public class GenericClass {


        public static List sort(List l) {
         Collections.sort(l, new           <Person>() {

                        public int compare(Person p1, Person p2) {
                                return p1.account.         (p2.account);
                        }
        });

                return l;



}
}
```

JUNIT :

```
import static org.junit.Assert.*;

import java.util.ArrayList;
import java.util.List;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

public class GenericTest {

        static List<Person> list=new          <>();
        static List<Person> list1=new          <>();
        static Person p1,p2,p3,p4 ;
        GenericClass g ;



        public void setUp() throws Exception {
        p1= new       (2,"abc");
        p2= new       (1,"pqqr");
        p3 = new       (4,"stu");
        p4 = new       (3,"vwx");
        list.add(p1);
        list.add(p2);
        list.add(p3);
        list.add(p4);
        g = new              ();
        list1.add(p2);
        list1.add(p1);
        list1.add(p4);
        list1.add(p3);

        }
        public void tearDown() throws Exception {
        }


        public void test() {
          assertEquals(g.sort(list), list1);
        }

}
```

Output:



```
	@Before
	public void setUp() throws Exception {
	p1= new Person(2,"abc");
	p2= new Person(1,"pqqr");
	p3 = new Person(4,"stu");
	p4 = new Person(3,"vwx");
	list.add(p1);
	list.add(p2);
	list.add(p3);
	list.add(p4);
	g = new GenericClass();
	list1.add(p2);
	list1.add(p1);
	list1.add(p4);
	list1.add(p3);

	}

	@After
	public void tearDown() throws Exception {
	}

	@Test
	public void test() {
	   assertEquals(g.sort(list), list1);
	}

}
```

```
Finished after 0.028 seconds
Runs: 1/1     Errors: 0     Failures: 0
GenericTest [Runner: JUnit 4] (0.000 s)
```

Topic 7: Multithreading

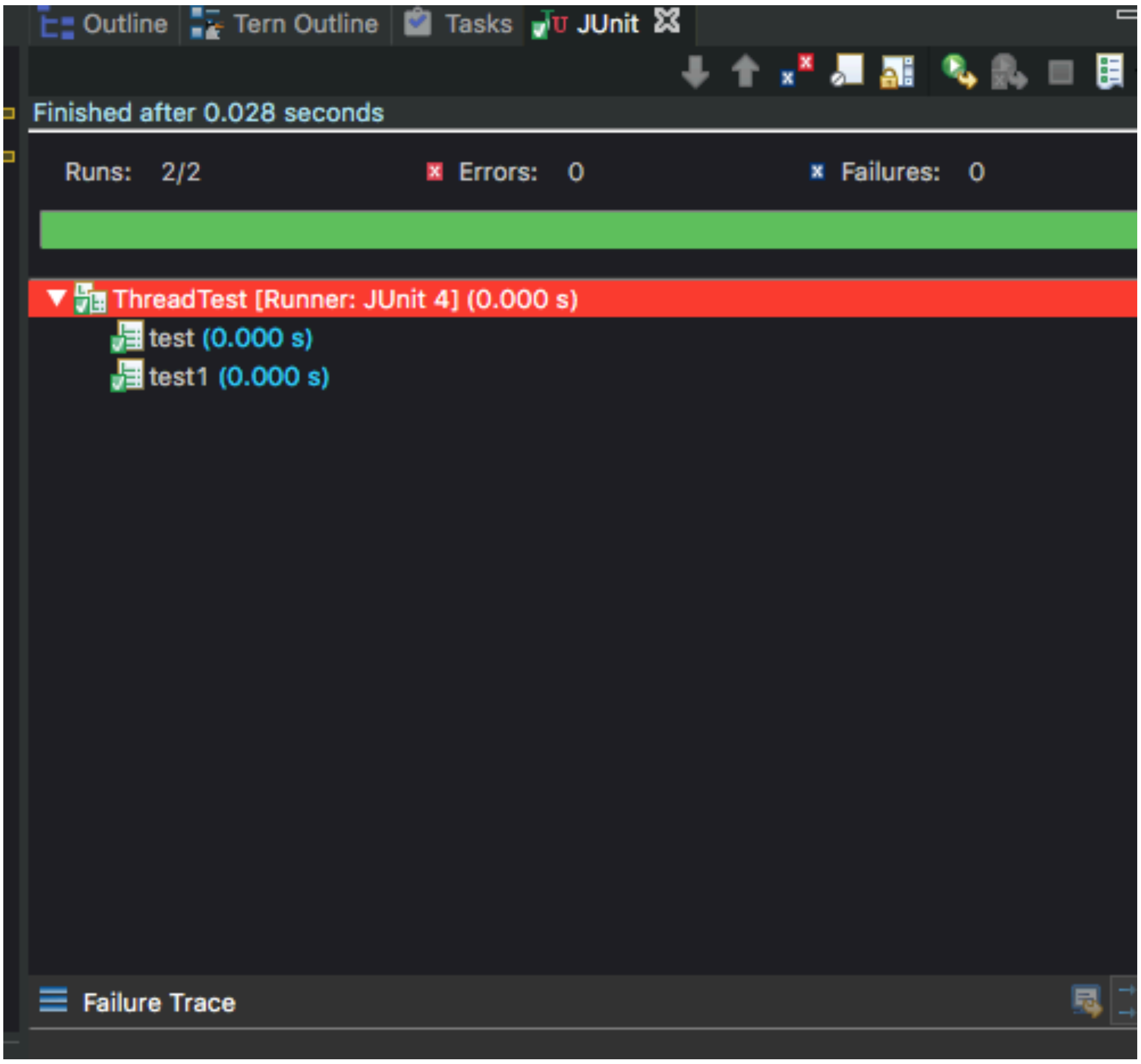
Multithreading is the ablity to run more than one threads , the program can divide the program into multiple segments and run the threads individually that can save a lot of time wasted in the IO operations. It is basically dividing the process into multiple smaller segments called threads and run them simultaneously to achieve greater efficiency.

```
// Question : Write a program to demonstrate the threading , and how it works.

public class thread implements Runnable {
         private Thread t;
         private String threadName;

         thread( String name) {
             threadName = name;
             System.out.        ("Creating " +  threadName );
         }

         public void run() {
             System.out.        ("Running " +  threadName );
             try {
                for(int i = 4; i > 0; i--) {
                    System.out.       (threadName + ", " + i);
                    // Let the thread sleep for a while.
                    Thread.sleep(50);
                }
            }catch (InterruptedException e) {
                System.out.       ("Thread " +  threadName + " interrupted.");
            }
            System.out.       ("Thread " +  threadName + " exiting.");
        }

        public void start () {
            System.out.       ("Starting " +  threadName );
            if (t == null) {
                t = new         (this, threadName);
                t.      ();
            }
        }
    }
```

JUNIT :

```
import static org.junit.Assert.*;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

public class ThreadTest {




       import static org.junit.Assert.assertEquals;
       import org.junit.After;
       import org.junit.Test;
       import org.junit.runner.RunWith;
       import com.anarsoft.vmlens.concurrent.junit.ConcurrentTestRunner;
               (ConcurrentTestRunner.class)
       public class TestCounter {
           private Counter counter = new Counter();

           public void addOne()
           {
               counter.addOne();
           }

           public void testCount()
           {
               assertEquals("4 Threads running addOne in parallel should lead to 4" , 4
, counter.getCount());
           }
       }
}
```

Output :



```
Finished after 0.028 seconds

Runs:   2/2          Errors:   0          Failures:   0

▼ ThreadTest [Runner: JUnit 4] (0.000 s)
      test (0.000 s)
      test1 (0.000 s)
```