**CMPE 273**

**Assignment #1**

**Name: Parth Mukeshkumar Pandya**

**SJSU ID: 010805548**

Video Link: <https://www.youtube.com/watch?v=ZPjaBtuJ3NQ>

GitHub: <https://github.com/parthpandya17/273-Homework1>

**1. JavaScript**

**1.1 Variables** :

*1.1.1:Introduction to the topic-*

Variables in javascript are named containers that can hold value of any data type.

Three types of primitive data types supported by Java Script.

1)Numbers:130,179.5 etc

2)Strings : ”hello world” etc

3)Boolean: true or false

Var keyword is used to declare a variable. Variable can have either of two scope: Global or Local.

* Global variables-Has scope throughout thejavascript code and can be accessed from anywhere in script.
* Local variables: Scope is limited to the function in which it is declared and can be accessed from that function only.

Local variable overshadows global variables i.e declaring a local variable with a same name as that of a global variable, will hide the global variable.

*1.1.2:Programming Question-*

Write a program to convert temperature from Fahrenheit to Celsius demonstrating the use of java script variables.

*1.1.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Variable </title>

</head>

<body>

<script type=*"text/javascript"*>

**function** convert()

{

**var** t = document.getElementById("txtTemp1").value;

t = (t-32)\*5/9;

alert(t + " degree celsius");

}

</script>

</head>

<body>

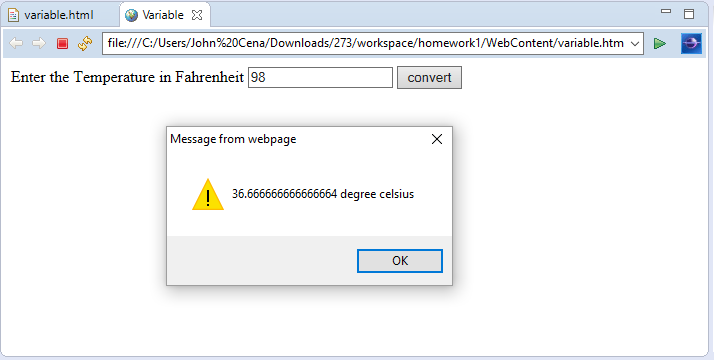
Enter the Temperature in Fahrenheit <input type = *"text"* name = *"temperature"* id = *"txtTemp1"*>

<input type = *"button"* value = *"convert"* onclick = "convert()" id = *"bConvert1"*>

</body>

</html>

*1.1.4:Output of Execution*



**1.2 Objects** :

*1.2.1:Introduction to the topic:*

Objects in Javascript are variables too. But can contain many name value pair. Object consists of methods and properties. Object can be defined as a standalone entity having properties and type. Property of an object ae variables associated with the object. Object in javascript can be created by following ways:

1. Var obj = {type: “person”}
2. Var obj new object();

Obj.type = “person”;

1. Function object(type)

{

this.type = type;

}

Var obj = new object(“Person”);

*1.2.2:Programming Question:*

Write a code demonstrating the use of objects in javascript.

*1.2.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>object</title>

</head>

<body>

<script>

**var** param;

**function** f1() {

**var** t = document.getElementById("t1");

**var** fn = document.getElementById("fn");

**var** ln = document.getElementById("ln");

**var** id = document.getElementById("id");

t.style.display = "";

**var** row = t.insertRow(-1);

row.style.height = "50";

**var** cell = row.insertCell(0);

**var** cell1 = row.insertCell(1);

**var** cell2 = row.insertCell(2);

**var** objaddPerson = **new** person(fn.value,ln.value, id.value);

cell.innerHTML = objaddPerson.firstname;

cell1.innerHTML = objaddPerson.lastname;

cell2.innerHTML = objaddPerson.id;

fn.value = "";

ln.value = "";

id.value = "";

}

**function** person(fname, lname, id)

{

**this**.firstname = fname;

**this**.lastname = lname;

**this**.id = id;

}

</script>

</head>

<body>

First name

<input type=*"text"* name=*"fn"* id=*"fn"*> Last name

<input type=*"text"* name=*"ln"* id=*"ln"*> Student Id

<input type=*"text"* name=*"id"* id=*"id"*>

<input type=*"button"* value=*"add"* onclick="f1()" id=*"b"*>

<table width=*"500"* border=*"1"* align=*"center"* id=*"t1"*

style="display: *none*">

<tr height=*"50"*>

<td width=*"150"* id=*""*>First Name

<td width=*"150"*>Last Name

<td width=*"150"*>Student Id

</td>

</td>

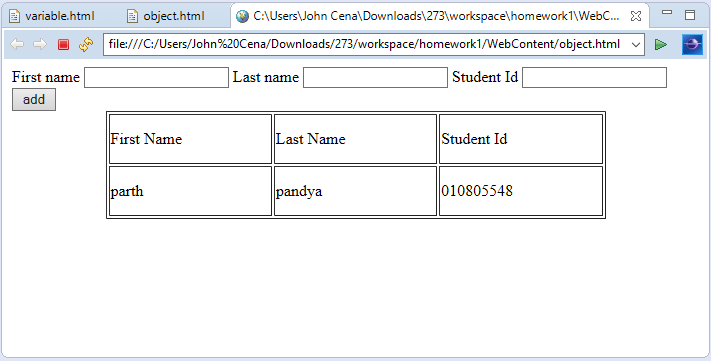
</tr>

</table>

</body>

</html>

*1.2.4:Output*



**1.3 Functions:**

*1.3.1:Introduction*

Functions in javascript is a block of coded which defined once, can be executed or invoked multiple times, to perform a particular task. Javascript function declaration includes function keyword, a name, and arguments that it will take. Function assigned to the property of the object is called method of that object. Functions can be invoked on occurance of event, invoked from code or can be self invokef

*1.3.2:Programming Question:*

Write a program to generate random number demonstrating the use of functions in javascript.

*1.3.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Functions</title>

</head>

<body>

<script type=*"text/javascript"*>

**var** time;

**function** timestamp()

{

**var** timestamp = **new** Date();

alert(timestamp);

}

</script>

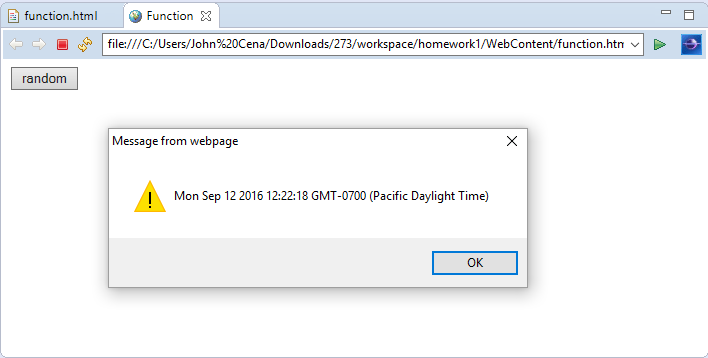
</head>

<input type = *"button"* value = *"random"* id = *"random"* onclick = "timestamp()">

</body>

</html>

*1.3.4:Output*



**1.4 Events:**

*1.4.1:Introduction*

Events represents the act of doing something in javascript on browser. There are a various event related objects in javascript. Event handler is the set of lines executed in response to the event occurred. The object on which event occurs is called the target. Events in javascript

* Onchange
* Onmouseover
* Onmouseout
* Onclick
* Onkeydown
* Onload

Events handlers can be executed directly by event attributes which can call the event handler functions on occurrence of event. Events can be ignored as well.

*1.4.2:Programming Question*

Write a program to demonstrate the use of javascript events using onmouseover event.

*1.4.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Events</title>

<script>

**function** f1(x,y)

{

**var** b11 = document.getElementById("b1");

**var** b21 = document.getElementById("b2");

b11.value=x;

b21.value=y;

}

**function** f2()

{

alert("oh.. u r mad indeed..!! :P");

}

</script>

</head>

<body>

are you mad..?

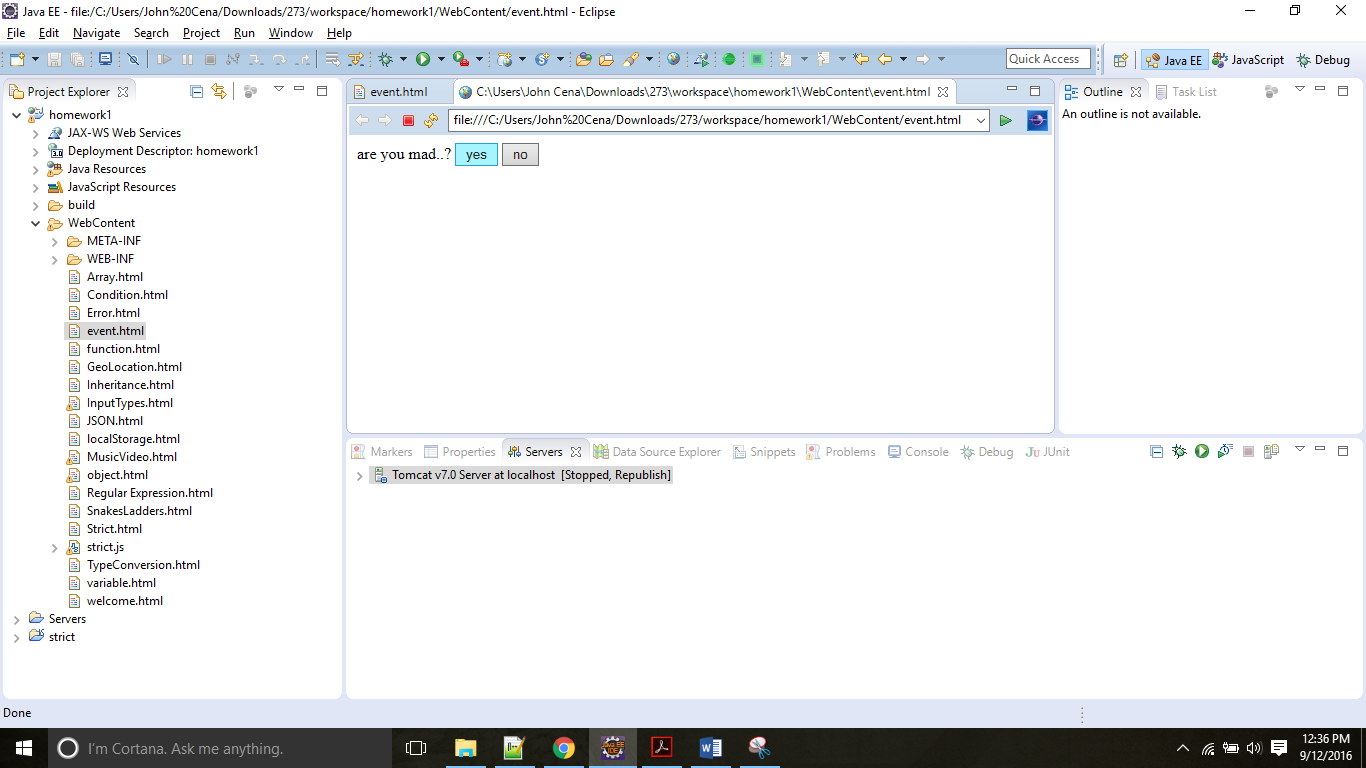
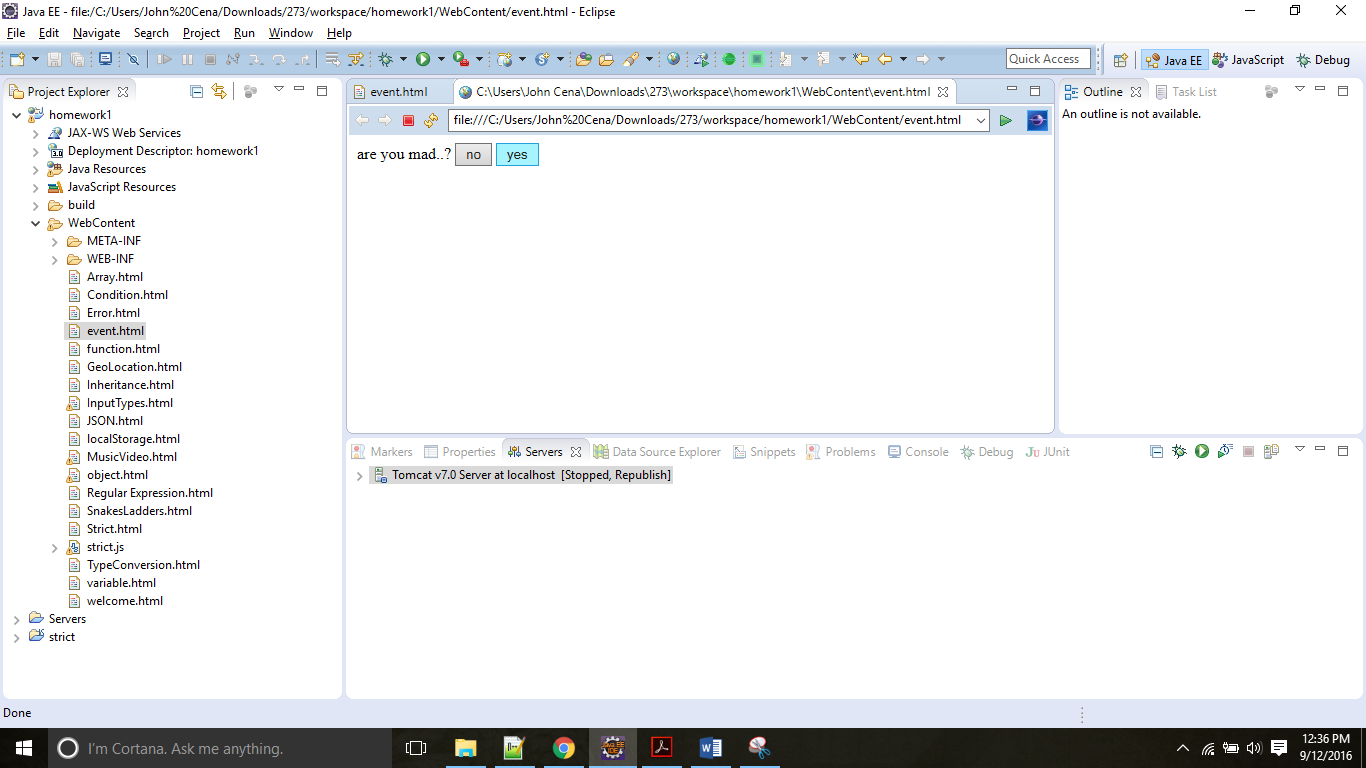
<input type=*"button"* value = *"yes"* id = *"b1"* onmouseover="f1('yes','no')" onclick="f2()">

<input type=*"button"* value = *"no"* id = *"b2"* onmouseover="f1('no','yes')" onclick="f2()">

</body>

</html>

*1.4.4:Output*

**1.5 Arrays**

*1.5.1:Introduction*

An array is an ordered collection of values which can store more than one value at a time. Each value is called an element and each element has a numeric position in the array called index. Elements cab be accessed by index. Javascript arrays can be created using the keyword new. Arays have a length property that show count of elements.

*1.5.2:Programming Question:*Program

Write a program to take readings and add them to an array.

*1.5.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Arrays</title>

<script type=*"text/javascript"*>

**var** array = [];

**function** add()

{

**var** elements = document.getElementById("readings");

**var** display = document.getElementById("display");

array.push(elements.value);

display.value = array;

elements.value = "";

}

</script>

</head>

<body>

Enter the readings <input type = *"text"* name = *"readings"* id = *"readings"*>

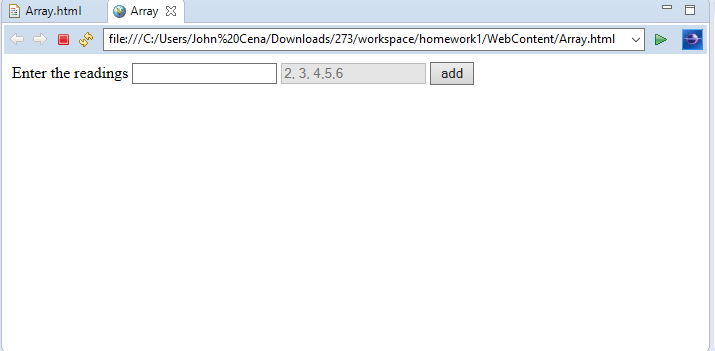
<input type = *"text"* name = *"display"* id = *"display"* disabled>

<input type = *"button"* value = *"add"* onclick = "add()">

</body>

</html>

*1.5.4:Output*



**1.6 Inheritance**

*1.6.1:Introduction*

Inheritances in javascript is prototype based and not classical, since there are no class in javascript. In javascript, object inherits from other objects. By setting the prototype to a particular object, the child object gains all the properties of the prototype object.

*1.6.2:Programming Question:*

White a program demonstrating the use of inheritance in java script.

*1.6.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>inheritance</title>

</head>

<body>

<script type=*"text/javascript"*>

**function** Person(fname, lname, gender)

{

**this**.fname = fname;

**this**.lname = lname;

**this**.gender = gender;

}

**function** Employee(empid)

{

**this**.empid = empid;

}

**function** initialize()

{

Employee.prototype = **new** Person("Parth", "Pandya", "M");

**var** objEmployee1 = **new** Employee(71);

document.getElementById("fname").innerHTML = "First name = " + objEmployee1.fname + " objEmployee1.hasOwnProperty = "+ objEmployee1.hasOwnProperty("fname");

document.getElementById("lname").innerHTML = "Last name = "+ objEmployee1.lname +" objEmployee1.hasOwnProperty = "+ objEmployee1.hasOwnProperty("lname");

document.getElementById("gender").innerHTML = "Gender = " + objEmployee1.gender +" objEmployee1.hasOwnProperty = "+ objEmployee1.hasOwnProperty("gender");

document.getElementById("id").innerHTML = "Employee id " + objEmployee1.empid +" objEmployee1.hasOwnProperty = "+ objEmployee1.hasOwnProperty("empid");

}

</script>

<body onLoad = "initialize()">

<p id = *"fname"*></p>

<p id = *"lname"*></p>

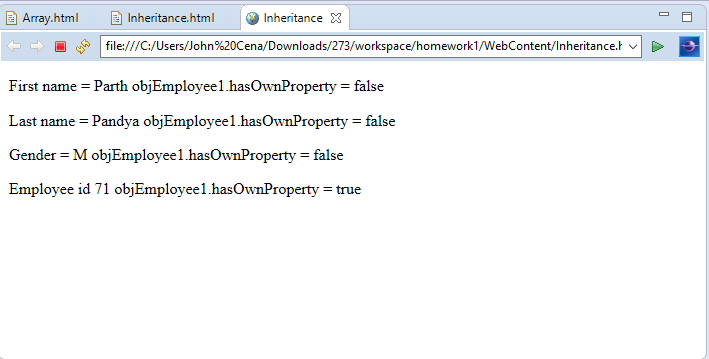
<p id = *"gender"*></p>

<p id = *"id"*></p>

</body>

</html>

*1.6.4:Output*



**1.7 Conditions**

*1.7.1:Introduction*

To make decisions based on different conditions, javascript provides conditional statements. Javascript provides following conditional statements:

* If
* Else if
* Else
* Switch

If is used to test the first condition, else if to test other conditions, and else when none of the condition is found true. Switch case, switches the variable for different value and executes the case accordingly.

*1.7.2:Programming Question:*

Using conditional operators in javascript write a code to diplsay the department head of the department entered by user.

*1.7.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Conditions</title>

</head>

<body>

<script type=*"text/javascript"*>

**function** result()

{

**var** marks = document.getElementById("marks");

**var** result = document.getElementById("result");

**if**(marks.value > 95)

{

result.innerHTML = "Grade: A+";

}

**else** **if**(marks.value<= 95 && marks.value > 89)

{

result.innerHTML = "Grade: A";

}

**else** **if** (marks.value <= 89 && marks.value > 79)

{

result.innerHTML = "Grade: B";

}

**else** **if**(marks.value <=79 && marks.value> 70)

{

result.innerHTML = "Grade: C";

}

**else** **if** (marks.value <=70 && marks.value > 65)

{

result.innerHTML = "Grade: D";

}

**else**

{

result.innerHTML = "Grade: F";

}

}

</script>

Enter marks<input type = *"text"* id = *"marks"*>

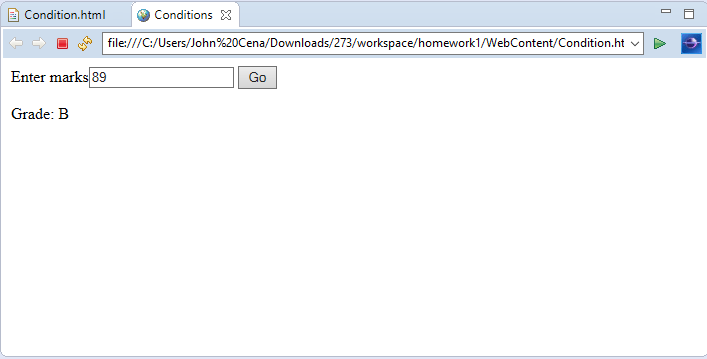
<input type = *"button"* value = *"Go"* onclick = "result()">

<p id = *"result"*></p>

</body>

</html>

*1.7.4:Output*



**1.8 Regular Expressions:**

*1.8.1:Introduction:*

In javascript , Regular Expressions define a pattern of characters which can be used to perform functions such as pattern matching and search-replace.

Syntax : /pattern/modifiers;

Following is a list of few special characters that can be used as regular expressions.

[abc] any of the characters in the brackets

[0-9] any of the digits in the brackets

(x|y) any of the alternatives separated with |

n+ Matches any string that contains at least one n

n\* Matches any string that contains zero or more occurrences of n

n? Matches any string that contains zero or one occurrences of n

*1.8.2:Programming Question:*

*Using regex write a code in javascript to validate the first name provided by user.*

*1.8.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Regular Expression</title>

</head>

<body>

<script type=*"text/javascript"*>

**function** validate()

{

**var** regex = /^[a-zA-Z ]{0,25}$/;

**var** fname = document.getElementById("fname");

**var** result = document.getElementById("result");

**if** (regex.test(fname.value))

{

result.innerHTML = "valid name";

fname.value = "";

**return** **true**;

}

**else**

{

result.innerHTML = "Invalid name";

fname.value = "";

**return** **false**;

}

}

</script>

First Name<input type = *"text"* id = *"fname"* >

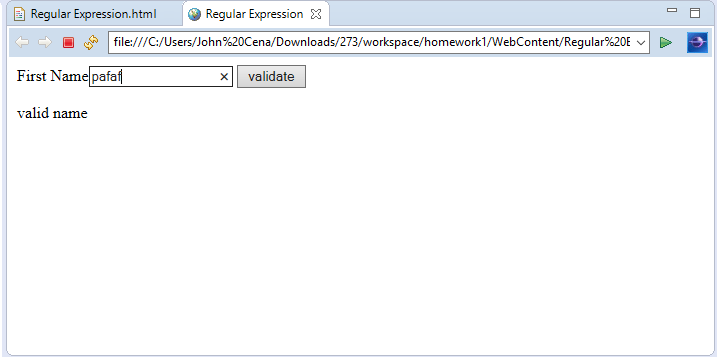
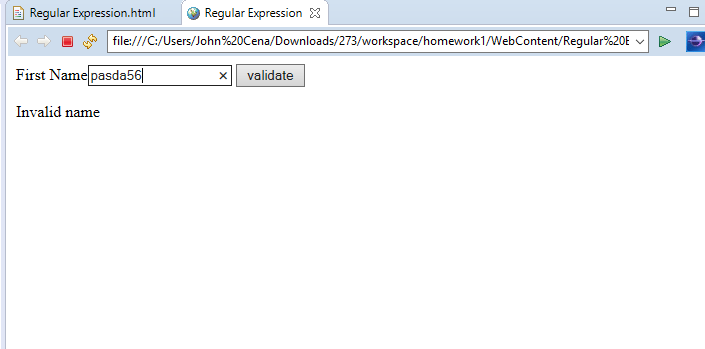
<input type=*"button"* value =*"validate"* onclick = "validate()">

<p id = *"result"*></p>

</body>

</html>

*1.8.4:Output*

*S*

**1.9 Strict Mode:**

*1.9.1: Introduction*

JavaScript provides strict mode to make code error free and prevents unwanted variables initialization. Strict mode converts bad syntax into errors. “use strict” is used as a syntax for declaring strict mode.

*1.9.2: Programming Question:*

Use strict mode and display the error message that is generated while modifying a read only value.

*1.9.3: Code*

/\*\*

\*

\*/

"use strict"

function Employee(empid)

{

this.empid = empid;

}

function initialize()

{

var objEmployee1 = new Employee(71);

Object.defineProperty(objEmployee1, "empid", {writable: false });

alert(objEmployee1.empid);

objEmployee1.empid = 17;

var valid = document.getElementById("valid");

valid.value = objEmployee1.empid;

alert(objEmployee1.empid);

}

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Strict</title>

<script type="text/javascript" src = "strict.js"></script>

</head>

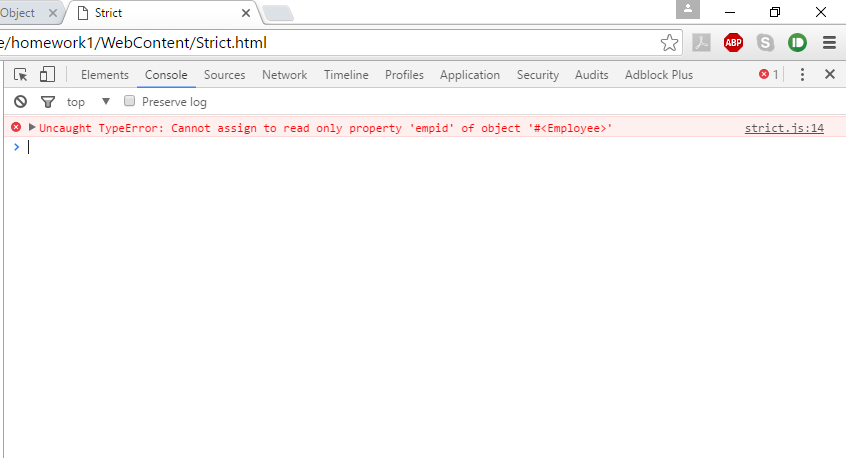
<body onload = "initialize()">

<p id ="valid"></p>

</body>

</html>

*1.9.4:Output*



**1.10 Errors:**

*1.10.1:Introduction*

Javascript provides a mechanism for exception handling in the form of try catch finally throws.

Try-Try block contains the block of code to be tested for errors

Catch-Error handling mechanism if error occurs is to be written inside the catch statement.

Throw-For creating custom errors and throwing them

Finally- The block of code executes irrespective of whether error occurs or not is to be written in finally block. Releasing resources and clean-up is done here.

*1.10.2:Programming Question*

Write a program to validate if input I number or not and Throw error if not.

*1.10.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Error</title>

<script type=*"text/javascript"*>

**function** validate()

{

**var** status = document.getElementById("status");

**var** number = document.getElementById("number");

**try**{

**if**(number.value > 0 && number.value < 1000000000)

{

status.innerHTML = "valid number";

}

**else**

{

**throw** "Not a valid number"

}

}

**catch**(err)

{

status.innerHTML = err;

}

}

</script>

</head>

<body>

Enter number<input type = *"text"* id = *"number"* >

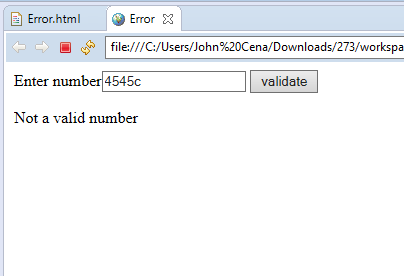
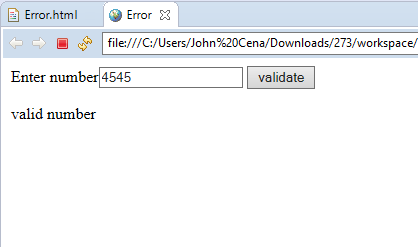
<input type = *"button"* value =*"validate"* onclick="validate()">

<p id = *"status"*></p>

</body>

</html>

*1.10.4:Output*



**1.11 Type Conversion:**

*1.11.1:Introduction*

Javascript has 5 data type that contains value(string, number, Boolean, function, object), 3 type of object(Array, Date, Object), 2 types that cannot contain value(undefined and null). In javascript it is not required to give a particular type of value to a variable. Variable can hold any type of value. Javascript feely allows conversion of values from one type to another.

*1.11.2:Programming Question*

Write a program to test a number using type conversion.

*1.11.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>TypeCoversion</title>

<script type=*"text/javascript"*>

**function** validate()

{

**var** status = document.getElementById("status");

**var** number = document.getElementById("number");

**var** x = Number(number.value);

**if**(isNaN(x))

{

status.innerHTML = "Not a number";

}

**else**

{

status.innerHTML = x + " is a valid Number";

}

}

</script>

</head>

<body>

Enter number<input type = *"text"* id = *"number"* >

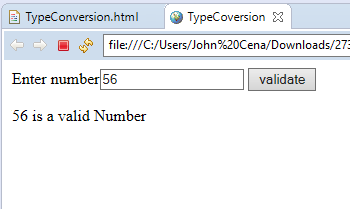
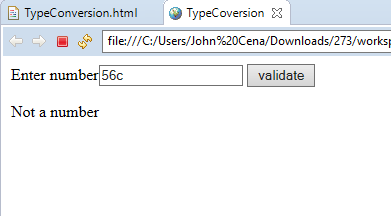
<input type = *"button"* value =*"validate"* onclick="validate()">

<p id = *"status"*></p>

</body>

</html>

*1.11.4:Output*



**1.12 JSON:**

*1.12.1:Introduction*

JSON, JavaScript Object Notation is similar to XML format to represent and transfer data. It is leight weight data interchange format. JSON uses human readable text to transmit data objects, so it is self-describing. JSON is independent of language. It is used in many programming languages. File extension for JSON is .json.

JSON Rules:

* Data is in form name/value pairs, name value separated by :
* Data is separated by commas
* Each Curly braces hold objects
* Square brackets hold arrays of objects

*1.12.2:Programming Question*

Using JSON parse text method extract and display the key pair values in JSON array

*1.12.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>JSON</title>

<script type=*"text/javascript"*>

**function** getCar()

{

**var** status = document.getElementById("status")

**var** json = '{ "car" : [' +

'{ "company":"Audi" , "model":"R8" },' +

'{ "company":"Toyota" , "model":"Corolla" },' +

'{ "compnay":"Honda" , "model":"Civic" } ]}';

**var** array = JSON.parse(json);

**var** index = document.getElementById("number");

status.innerHTML = array.car[index.value].company + " "+array.car[index.value].model;

}

</script>

</head>

<body>

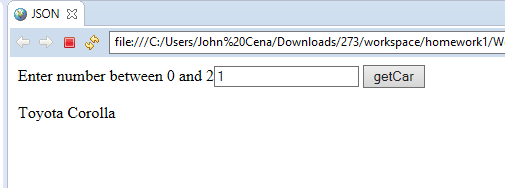
Enter number between 0 and 2<input type = *"text"* id = *"number"* >

<input type=*"button"* value=*"getCar"* onclick="getCar()">

<p id=*"status"*></p>

</body>

</html>*1.12.4:Output:*



**2.HTML5**

**2.1:Local Storage**

*2.1.1:Introduction*

Html5 provides the feature called local storage. Using local storage, web application can store data locally within the user’s browser. Local storage provides more security as compared to cookies and large amount of data can be stored locally. Local storage is origin based, means all the pages from same origin can store and access same data. The storage limit for local storage is atleast 5 mb. Local storage is further of two types: window.localStorage – stores data with unless removed, session.localstorage – stores data for a particular session.

*2.1.2:Programming Question* Using local Storage calculate number of times the page was visited.

*2.1.3:Code* <!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Local Storage</title>

<script type=*"text/javascript"*>

**function** pageVisited()

{

**if**(**typeof**(Storage) !== "undefined") {

**if** (localStorage.pageVisited) {

localStorage.pageVisited = Number(localStorage.pageVisited)+1;

} **else** {

localStorage.pageVisited = 1;

}

document.getElementById("page").innerHTML = "You have visited the page " + localStorage.pageVisited + " time(s).";

} **else** {

document.getElementById("page").innerHTML = "Browser does not support local storage...";

}

}

</script>

</head>

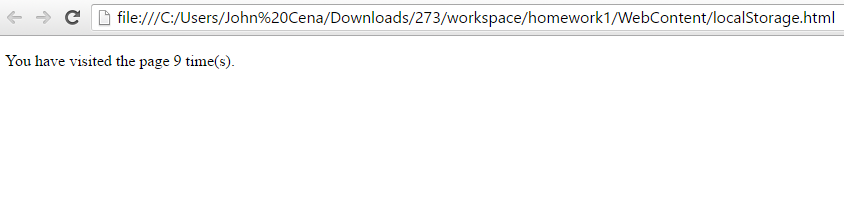
<body onload = "pageVisited()">

<p id = *"page"*></p>

</body>

</html>

*2.1.4:Output*



**2.2:Audio/Video**

*2.2.1:Introduction*

* Audio

Html5 provides <audio> element for embedding audio files in web page. The control attributes of the html5 <audio> add controls like play, pause, volume. The src elements let you specify audio files the brower will choose from. can be embedded in a web page. Duration can also be set.

* Video

Html5 provides video element for playing video files in a web page. Html5 provides <video> element for embedding audio files in web page. In addition the control attributes provides the means of play, pause, adjust volume options. Height and width attributes can be used to avoid flickering. MP4, WebM, ogg are the 3 formats supported in HTML5

*2.2.2:Programming Question*

* Write a code to demonstrate the use of audio tag in html5.
* Write a code to demonstrate the use of video player using html5.

*2.2.3:Code*

*1)Audio*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Audio & Video</title>

</head>

<body>

<video width=*"320"* height=*"240"* controls>

<source src=*"http://techslides.com/demos/sample-videos/small.mp4"* type=*"video/mp4"*>

</video>

<audio controls>

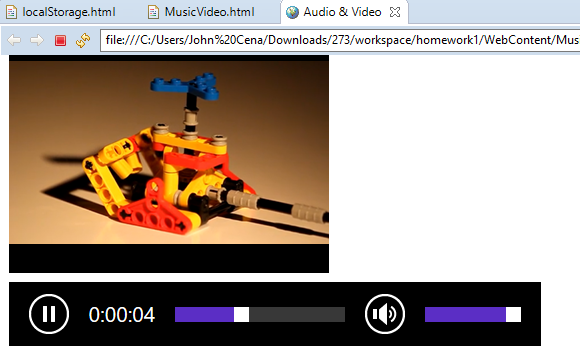
<source src=*"https://ia902508.us.archive.org/5/items/testmp3testfile/mpthreetest.mp3"* type=*"audio/mpeg"*>

</audio>

</body>

</html>

*2.2.4:Output*



**2.3:Input Types**

*2.3.1:Introduction:*

In addition to input types provided by html, html5 provides few extra input tag which specifies the type of input element. Some of them are:

* Color
* Number
* Url
* Email
* Tel
* Date
* password

Restrictions can be provided in the input type. Restrictions introduced in html5 are

* Multiple
* Pattern
* Required
* Autofocus

*2.3.2:Programming Question*

Write a program to demonstrate the use of new input type (Color, Number, Url, Email, tel, date, password) and restrictions(Multiple, Pattern, Required, Autofocus) specified in html 5

*2.3.3:Code*

<html>

<head>

</head>

<body>

<form action=*"welcome.html"*>

Car name

<input type=*"text"* name=*"name"* id=*"name"* autofocus=*"autofocus"* required=*"required"*><br>

Years used<input type = *"number"* name = *"yearsUsed"* id = *"yearsUsed"*><br>

Selling date<input type=*"date"* name=*"sellingDate"* id = *"sellingDate"*><br>

Car Color<input type=*"color"* name=*"carcolor"* id = *"carcolor"*><br>

Email<input type = *"email"* id = *"email"*required=*"required"*><br>

Contact number <input type= *"tel"* name = *"contactnum"* id = *"contactnum"* pattern=*"\d{3}[\-]\d{3}[\-]\d{4}"*><br>

Url for sample image<input type=*"url"* name=*"url"*>

Image<input type = *"file"* name = *"img"* multiple>

Enter username <input type = *"text"* id =*"username"* required=*"required"*><br>

Enter password<input type = *"password"* maxlength=*"16"* id = *"password"* required=*"required"*><br>

<input type=*"submit"* value=*"add"* onclick="f1()" id=*"b"*><br>

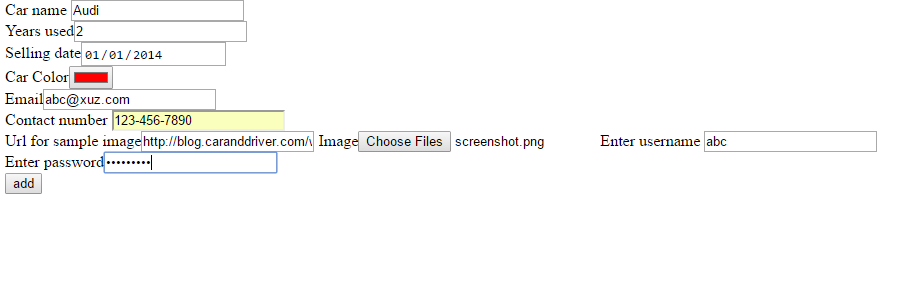
</form>

</table>

</body>

</html>

*2.3.4:Output*



**2.4:Geolocation**

*2.4.1:Introduction*

HTML5 provides with GeoLocation Api that can be used to determine the position of user. GetCurrentPosition() method provided can be used to provide the user position. Position of users cis retrieved in terms of longitude and latitude. But for this access, user permission is required.

*2.4.2:Programming Question*

Write a program to display the current geographical location of user in form of longitude and latitude.

*2.4.3:Code*

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>GeoLocation</title>

</head>

<body>

<script type=*"text/javascript"*>

**function** locate()

{

navigator.geolocation.watchPosition(showPosition);

}

**function** showPosition(myposition)

{

document.getElementById("data1").innerHTML="Longitude: "+myposition.coords.latitude+"<br>Latitude: "+myposition.coords.latitude+"<br>Accuracy?"+myposition.coords.accuracy+"<br>speed"+myposition.coords.speed;

**var** coordinates=myposition.coords.latitude+","+myposition.coords.longitude;

alert(coordinates);

**var** map = "http://maps.googleapis.com/maps/api/staticmap?center="

+coordinates+"&zoom=14&size=400x300&sensor=false";

document.getElementById("data2").innerHTML = "<img src='"+map+"'>";

}

</script>

<input type=*"button"* value =*"locate"* onclick="locate()">

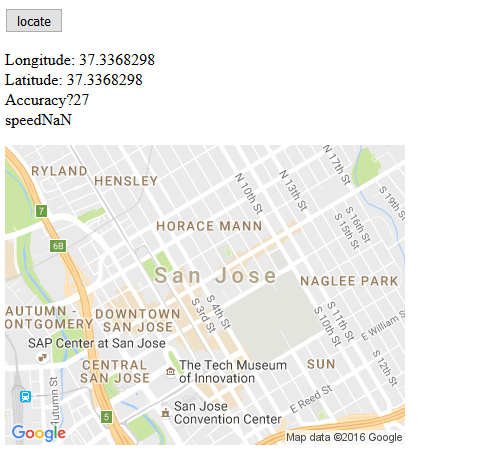
<p id =*"data1"*></p>

<div id =*"data2"*></div>

</body>

</html>

*2.4.4:Output*



**3.JAVA**

**3.1:Queue**

*3.1.1:Introduction*

The Queue interface in java is provided in java.util.\* package, which is extends the collection interface. Typically, Queue uses FIFO(First in First out) strategy to store the data. In addition to Collections operation, it supports addition, inspection and removal of operations.

*3.1.2:Programming Question:*

Write a sample code illustrating the use of Queue to manage movie ticket window.

*3.1.3:Code*

**package** com.main.example.classes;

**import** java.util.Queue;

**public** **class** QueueTheatre

{

**private** Queue<String> line;

**public** **void** addToline(String sName)

{

**if**(**this**.line != **null**)

**this**.line.add(sName);

}

**public** String removeFromLine()

{

String sPerson = **null**;

**if**(**this**.line != **null** && **this**.line.size() > 0)

{

sPerson = **this**.line.remove();

}

**return** sPerson;

}

**public** String getFirst()

{

String sPerson = **null**;

**if**(**this**.line != **null** && **this**.line.size() > 0)

{

sPerson = **this**.line.peek();

}

**return** sPerson;

}

**public** Queue<String> getLine() {

**return** line;

}

**public** **void** setLine(Queue<String> line) {

**this**.line = line;

}

**public** String issueTicket()

{

String sPerson = **null**;

**if**(**this**.line != **null** && **this**.line.size() > 0)

{

sPerson = **this**.line.poll();

}

**return** sPerson;

}

}

<Test Cases>

@Test

**public** **void** testAddToline()

{

QueueTheatre obj = **new** QueueTheatre();

obj.setLine(**new** LinkedList<String>());

*assertEquals*(**null**, obj.getLine().peek());

obj.addToline("parth");

*assertEquals*("parth", obj.getLine().peek());

}

@Test

**public** **void** testRemoveFromLine()

{

QueueTheatre obj = **new** QueueTheatre();

obj.setLine(**new** LinkedList<String>());

*assertEquals*(**null**, obj.removeFromLine());

obj.addToline("parth");

*assertEquals*("parth", obj.removeFromLine());

}

@Test

**public** **void** testGetFirst()

{

QueueTheatre obj = **new** QueueTheatre();

obj.setLine(**new** LinkedList<String>());

*assertEquals*(**null**, obj.getFirst());

obj.addToline("parth");

*assertEquals*("parth", obj.getFirst());

}

@Test

**public** **void** testIssueTicket()

{

QueueTheatre obj = **new** QueueTheatre();

obj.setLine(**new** LinkedList<String>());

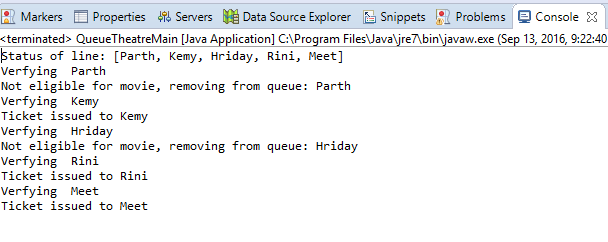
*assertEquals*(**null**, obj.issueTicket());

obj.addToline("parth");

*assertEquals*("parth", obj.issueTicket());

}

*3.1.4:Output*



**3.2:Stack**

*3.2.1:Introduction*

Stack in java is a class that extends the vector class supporting the five operations of vector class. Stack stores data in standard LIFO(Last In First Out) order. In addition, Stack defines peek, poll, push, empty and search method. Push is to add an item to stack, pop removes the first item from stack, peek gives the first item on sack but does not remove it from stack, empty checks whether stack contains any elements or not, search searches for the object in stack and returns the position from top of stack.

*3.2.2:Programming Question:*

Write a program to reverse a string using stack.

*3.2.3:Code*

**package** com.main.example.classes;

**import** java.util.Stack;

**public** **class** ReverseStringStack

{

**private** Stack<String> objStack;

**public** Stack getObjStack()

{

**return** objStack;

}

**public** **void** setObjStack(Stack objStack)

{

**this**.objStack = objStack;

}

**public** String reverseString(String sOriginal)

{

StringBuilder sbReversedString = **new** StringBuilder();

**if**(**this**.objStack != **null** && sOriginal != **null** && sOriginal.length() > 0)

{

**for**(**int** i = 0 ; i < sOriginal.length();i++)

{

**this**.objStack.push(sOriginal.substring(i, i+1));

}

**for**(**int** i = 0 ; i < sOriginal.length();i++)

{

sbReversedString.append(**this**.objStack.pop()) ;

}

**return** sbReversedString.toString();

}

**else**

{

**return** **null**;

}

}

}

**🡨Test Case🡪**

**package com.test.example.classses;**

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

**import java.util.Stack;**

**import org.junit.Test;**

**import com.main.example.classes.ReverseStringStack;**

**public class ReverseStringStackTest**

**{**

**@Test**

**public void testReverseString()**

**{**

**ReverseStringStack objReverseStringStack = new ReverseStringStack();**

**objReverseStringStack.setObjStack(new Stack<String>());**

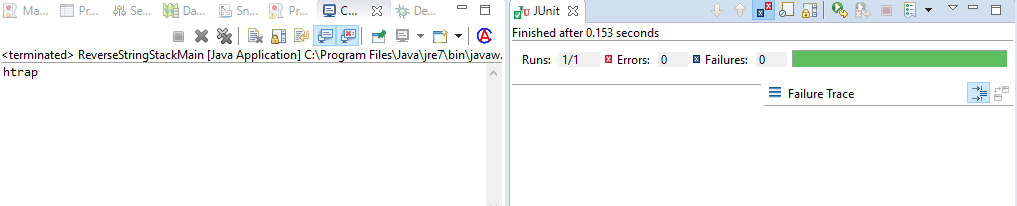
**assertEquals(null, objReverseStringStack.reverseString(null));**

**assertEquals("htrap",objReverseStringStack.reverseString("parth"));**

**}**

**}**

*3.2.4:Output*



**3.3:Array**

*3.3.1:Introduction*

Java provides a data structure called array to store a fixed number of elements accessed by the index. An array can be initialized to hold elements in following ways:

dataType arrayRefVar = new dataType[arraySize];

This creates an array of type datatype with size arraySize and assigns it to variable arrayRefVar.

*3.3.2:Programming Question:*

Write an program to store the marks for a student illustrating use of arrays.

*3.3.3:Code*

**package** com.main.example.classes;

**public** **class** ArrayMarks

{

**private** **int** nMarks[];

**public** **int**[] getnMarks()

{

**return** nMarks;

}

**public** **void** setnMarks(**int**[] nMarks)

{

**this**.nMarks = nMarks;

}

**public** **int** getTotal()

{

**int** nSum = 0;

**if** (nMarks != **null**)

{

**for** (**int** i = 0; i < **this**.nMarks.length; i++)

{

nSum += **this**.nMarks[i];

}

}

**return** nSum;

}

**public** **int** getHighest()

{

**int** nHighest=0;

**if**(nMarks != **null**)

{

nHighest=nMarks[0];

**for**(**int** i = 0 ; i < **this**.nMarks.length-1; i++)

{

nHighest = (nMarks[i] > nMarks[i+1])? ( nMarks[i]):(nMarks[i+1]);

}

**return** nHighest;

}

**return** nHighest;

}

}

🡨Test Code🡪

package com.test.example.classses;

import static org.junit.Assert.\*;

import org.junit.Test;

import com.main.example.classes.ArrayMarks;

public class ArrayMarksTest

{

@Test

public void testHighest()

{

ArrayMarks objArrayMarks = new ArrayMarks();

int[] nArray = {89,75,45,68,98};

assertEquals(0,objArrayMarks.getHighest());

objArrayMarks.setnMarks(nArray);

assertEquals(98, objArrayMarks.getHighest());

}

@Test

public void testTotal()

{

ArrayMarks objArrayMarks = new ArrayMarks();

int[] nArray = {89,75,45,68,98};

assertEquals(0,objArrayMarks.getTotal());

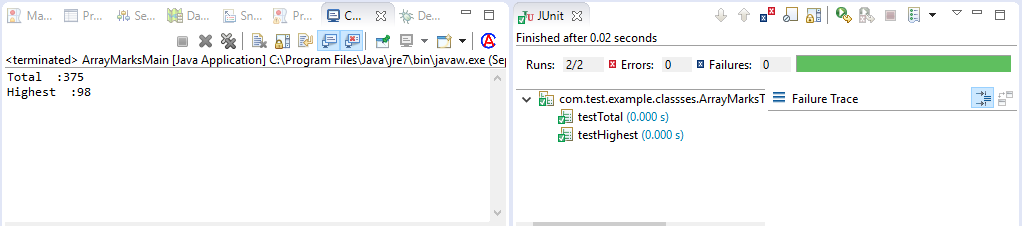
objArrayMarks.setnMarks(nArray);

assertEquals(375, objArrayMarks.getTotal());

}

}

*3.3.4Output*



**3.4:Interfaces**

*3.4.1:Introduction*

Interfaces in java are collection of abstract methods. A Class while implementing an interface signs a contract that it must provide the definition of all the methods that are declared in the interface.  
All the methods defined in the interface are by default public abstract means they have no definition. All the variables declared in an interface are by default public static final that means they are constant. Interface can also be used as a data type , to reference to object of any class that implements it.

*3.4.2:Programming Question*

Write a Service interface and a class implementing it.

*3.4.3:Code*

**package** com.main.example.interfaces;

**public** **interface** Service

{

**void** parseInput(String[] sData);

**void** executeService();

**void** generateOutput();

}

**package com.main.example.classes;**

**import java.util.LinkedList;**

**import java.util.List;**

**import com.main.example.interfaces.Service;**

**public class CSVServiceImpl implements Service**

**{**

**private List<String[]> lsCSVData ;**

**private List<String> lsOutput ;**

**public List<String[]> getLsCSVData()**

**{**

**return lsCSVData;**

**}**

**public List<String> getLsOutput()**

**{**

**return lsOutput;**

**}**

**public CSVServiceImpl()**

**{**

**this.lsCSVData = new LinkedList<String[]>();**

**this.lsOutput = new LinkedList<String>();**

**}**

**public synchronized void executeService()**

**{**

**if(!lsCSVData.isEmpty() && lsCSVData != null && lsOutput !=null)**

**{**

**for(String[] sLine : lsCSVData)**

**{**

**StringBuffer sModLine = new StringBuffer();**

**for(String sElement : sLine)**

**{**

**sElement = sElement.toUpperCase();**

**sModLine.append(sElement);sModLine.append('-');**

**}**

**lsOutput.add(sModLine.toString().substring(0, sModLine.length() -1));**

**}**

**}**

**}**

**public synchronized void parseInput(String[] sData)**

**{**

**if (sData != null && lsCSVData != null)**

**{**

**for (String sline : sData)**

**{**

**this.lsCSVData.add(sline.split(","));**

**}**

**}**

**}**

**public synchronized void generateOutput()**

**{**

**if(lsOutput != null && !lsOutput.isEmpty())**

**{**

**for(String sLine : lsOutput)**

**{**

**System.out.println(sLine);**

**}**

**}**

**}**

**}**

***<test code>***

**package com.test.example.classses;**

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

**import java.util.Iterator;**

**import org.junit.BeforeClass;**

**import org.junit.Test;**

**import com.main.example.classes.CSVServiceImpl;**

**public class CSVServiceImplTest**

**{**

**static CSVServiceImpl objCSVServiceImpl;**

**static String[] sData = { "parth,pandya" };**

**@BeforeClass**

**public static void Load()**

**{**

**objCSVServiceImpl = new CSVServiceImpl();**

**}**

**@Test**

**public void testParseInput()**

**{**

**String[] sTemp = {"parth","pandya"};**

**objCSVServiceImpl.parseInput(sData);**

**Iterator<String[]> i = objCSVServiceImpl.getLsCSVData().iterator();**

**String[] sResult = i.next();**

**assertEquals(sTemp, sResult);**

**}**

**@Test**

**public void testExecuteService()**

**{**

**objCSVServiceImpl.executeService();**

**Iterator<String> i = objCSVServiceImpl.getLsOutput().iterator();**

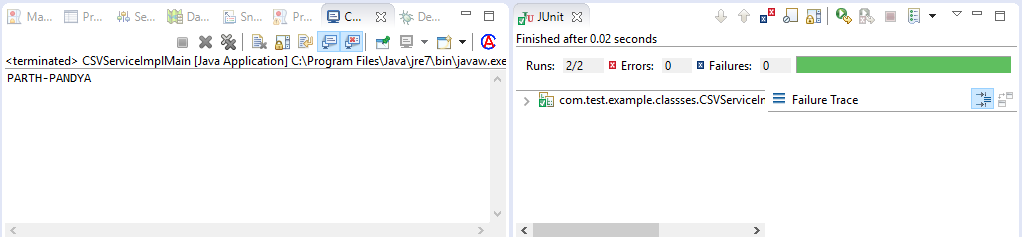
**String sResult = i.next();**

**assertEquals("PARTH-PANDYA", sResult);**

**}**

**}**

*3.4.4:Output*



**3.5 Collections:**

*3.5.1: Introduction*

Collections is a class provided in java under java.util package. This class provides a set of static methods to operate on collections.

*3.6.2:Programming Question:*

Write a program demonstrating the use of collections in java.

*3.6.3:Code*

**package** com.main.example.classes;

**public** **class** Employee

{

**public** String sFirstName;

**public** Integer nEmployeeId;

**public** Employee(String sFirstName, Integer nId)

{

**this**.nEmployeeId = nId;

**this**.sFirstName = sFirstName;

}

}

**package** com.main.example.classes;

**import** java.util.Comparator;

**public** **class** EmployeeSortById **implements** Comparator<Employee>

{

@Override

**public** **int** compare(Employee emp1, Employee emp2)

{

**return** emp1.nEmployeeId.compareTo(emp2.nEmployeeId);

}

}

**package** com.main.example.classes;

**import** java.util.Comparator;

**public** **class** EmployeeSortByName **implements** Comparator<Employee>

{

@Override

**public** **int** compare(Employee emp1, Employee emp2)

{

**return** emp1.sFirstName.compareTo(emp2.sFirstName);

}

}

🡨Test Class🡪

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

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

**import** java.util.Scanner;

**import** java.util.Set;

**import** org.junit.Test;

**public** **class** CollectionDemoTest {

CollectionDemo cd=**new** CollectionDemo();

HashMap<Integer,String> mapp=cd.CDemo();

@Test

**public** **void** testcollection()

{

Scanner sc= **new** Scanner(System.***in***);

System.***out***.println("Please enter the id number to know the name of employee");

**int** a=sc.nextInt();

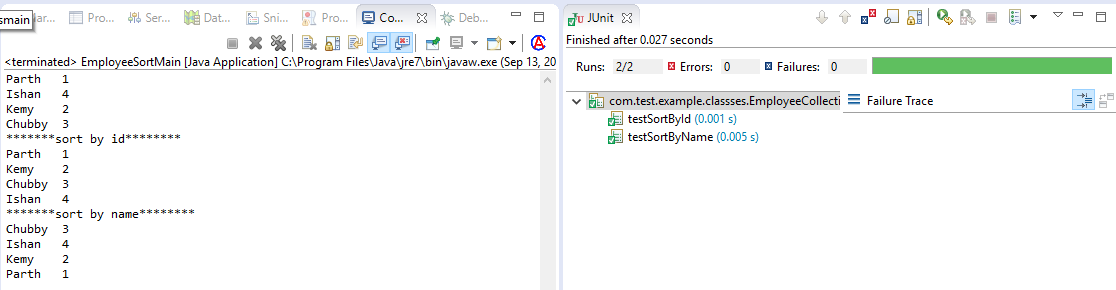
String name=(String)mapp.get(a);

*assertEquals*("Marry",name);

}

}

*3.6.4:Output*



**3.6Generics**

*3.6.1:Introduction*

Generics is the compile time safety feature provided by java to make sure that the random different types of objects are not added to a collection. It enables to defines a set of defined related data types to work on set of class or methods.

*3.6.2:Programming Question:*

Using generics write a code to dynamically pass the data type of a variable.

*3.6.3:Code*

**package** com.main.example.classes;

**public** **class** Generics <T>

{

T tFlagValue;

**public** T gettFlagValue()

{

**return** tFlagValue;

}

**public** **void** settFlagValue(T tFlagValue)

{

**this**.tFlagValue = tFlagValue;

}

}

🡨Test Case🡪

package com.test.example.classses;

import static org.junit.Assert.\*;

import java.awt.Window.Type;

import org.junit.BeforeClass;

import org.junit.Test;

import com.main.example.classes.Generics;

public class GenericsTest

{

static Generics<String> g1;

static Generics<Integer> g2;

@BeforeClass

public static void load()

{

g1 = new Generics<String>();

g1.settFlagValue("NA");

g2 = new Generics<Integer>();

g2.settFlagValue(1);

}

@Test

public void testString()

{

assertTrue(g1.gettFlagValue().equals("NA"));

}

@Test

public void testInt()

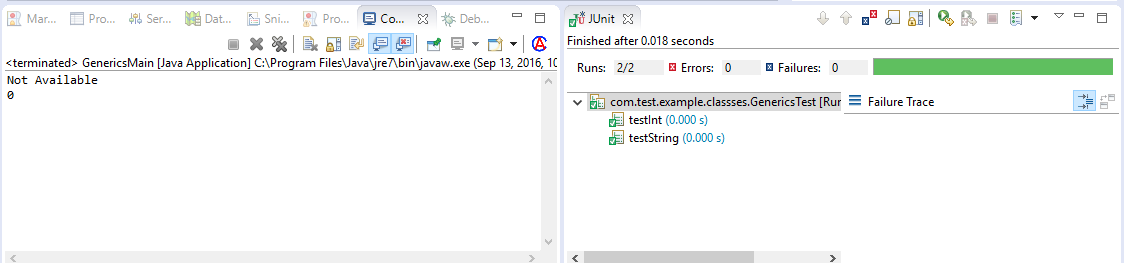
{

assertFalse(g2.gettFlagValue().equals(0));

}

}

*3.6.4:Output*



**3.7:Multithreading**

*3.7.1:Introduction*

Java language supports multithreading and hence can provide concurreny. Multithreading can be used to run different parts of program on different threads simultaneously. A new thread is created, it runs, waits for event, and then dies. A thread in java can be created by two ways:

1. Extending the Thread Class.
2. Implementing the runnable interface.

*3.7.2:Programming Question:*

Design a multithreaded datetime server using socket programming.

*3.7.3:Code*

*Thread.java*

**package com.main.example.classes;**

**import java.io.\*;**

**import java.net.\*;**

**import java.util.Date;**

**public class MultithreadedDateTimeServer implements Runnable**

**{**

**public static ServerSocket ssocMasterSocket;**

**private Socket socChildsocket;**

**private DataOutputStream dos;**

**private static int nRequestCount = 0;**

**public static int getnRequestCount()**

**{**

**return MultithreadedDateTimeServer.nRequestCount;**

**}**

**public MultithreadedDateTimeServer(Socket childSocket)**

**{**

**this.socChildsocket = childSocket;**

**}**

**public void run()**

**{**

**System.out.println("connected to " + this.socChildsocket.getInetAddress());**

**try**

**{**

**String sDateTme = new Date().toString();**

**this.dos = new** **DataOutputStream(socChildsocket.getOutputStream());**

**this.dos.writeBytes(sDateTme);**

**System.out.println("Time sent: " +sDateTme);**

**this.doIncrementRequestCount();**

**this.socChildsocket.close();**

**} catch (Exception e)**

**{**

**e.printStackTrace();**

**}**

**}**

**private synchronized void doIncrementRequestCount()**

**{**

**MultithreadedDateTimeServer.nRequestCount++;**

**System.out.println("Number of request serviced: " + MultithreadedDateTimeServer.getnRequestCount());**

**}**

**}**

*ThreadTest.java*

**package com.test.example.classses;**

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

**import java.io.DataInputStream;**

**import java.io.IOException;**

**import java.net.Socket;**

**import java.net.UnknownHostException;**

**import java.util.ArrayList;**

**import java.util.Date;**

**import java.util.List;**

**import org.junit.Test;**

**import com.main.example.classes.MultithreadedDateTimeServer;**

**public class MultithreadedDateTimeServerTest extends Thread**

**{**

**static Date objDate = new Date();**

**@Test**

**public void testdoIncrementRequestCount()**

**{**

**MultithreadedDateTimeServerTest objClients = null;**

**List<MultithreadedDateTimeServerTest> threads = new ArrayList<MultithreadedDateTimeServerTest>();**

**for(int i = 0 ; i < 9; i++)**

**{**

**objClients = new MultithreadedDateTimeServerTest();**

**objClients.start();**

**threads.add(objClients);**

**}**

**try**

**{ for(MultithreadedDateTimeServerTest t : threads)**

**t.join();**

**System.out.println("Finished");**

**} catch (InterruptedException e)**

**{**

**e.printStackTrace();**

**}**

**assertEquals(new Date().getDate(), objDate.getDate());**

**}**

**public void run(){**

**try**

**{**

**Socket socket = new Socket("localhost", 1025);**

**DataInputStream dis = new DataInputStream(socket.getInputStream());**

**StringBuffer s = new StringBuffer();**

**String tmp;**

**while((tmp = dis.readLine()) != null)**

**{**

**s.append(tmp);**

**}**

**System.out.println(s.toString());**

**objDate = new Date(s.toString());**

**} catch (UnknownHostException e)**

**{**

**e.printStackTrace();**

**} catch (IOException e)**

**{**

**e.printStackTrace();**

**}**

**}**

**}**

*3.7.4:Output*

