### CMPE 273: Enterprise Distributed Systems

Homework 1:

Refresher Assignment

**Links:**

GitHub Link: <https://github.com/geekhardik/CMPE273-HW1.git>

YouTube Link: <https://youtu.be/8nXQBW0jR-A>

Name: Hardik P. Shah

SJSU ID: 010807017

Email ID : Hardik.p.shah@sjsu.edu

Date: 09/17/2016

**Table of Contents:**

I. [**JavaScript**](#JavaScript):

1. [Variables](#variables)
2. [Objects](#objects)
3. [Functions](#functions)
4. [Events](#events)
5. [Arrays](#array)
6. [Inheritance](#inheritance)
7. [Conditions](#conditions)
8. [Regular Expressions](#regexp)
9. [Strict Mode](#strictmode)
10. [Errors](#error)
11. [Type Conversions](#TypeConversions)
12. [JSON](#JSON)

II. [HTML:](#HTML5)

1. [Local Storage](#local_storage)
2. [Media (Video and Audio)](#Media)
3. [Input Types](#Input_type)
4. [Geolocation](#Geolocation)

III.[JAVA:](#JAVA)

1. [Queues](#Queues)
2. [Stacks](#Stacks)
3. [Arrays](#Arrays_java)
4. [Inheritance](#inheritance)
5. [Collections](#Collections)
6. [Generics](#Generics)
7. [Multithreading](#Multithreading)

**JavaScript:**

1) **Variables:**

To store information / data, variables are being used. A variable can be a name or identifier to represent data and its type.

- In JavaScript, the declaration of variable doesn’t require datatype to be mentioned explicitly. Meaning one can simply define a variable without its datatype – string / number / Boolean!

- In Javascript, variables are defined by ‘var’. For Example :

var carname = “Lamborghini”;

var check = true;

var mynumber = 123;

- The reserved words should not be used as javascript variables.

- variable name starts with either by letter or an underscore character. If it begins with number, it’s an invalid.

- Variable names are case-sensitive. ‘here’ and ‘Here’ are different.

**Programming Question:**

Create a program which will define types of variables: Local and static variables. Valid and invalid declarations. Also, create a small HTML entry which will take Height in feet as input from user and convert it in inches.

Code:

<!DOCTYPE html>

<html>

<head>

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

<title>Use of variables!</title>

</head>

<body>

<h1>Examples of variables!</h1>

<p>Below is an example of "global" variable :</p>

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

<p id=*"local"*>click to see an example of "local" variable!</p>

<p id=*"incorrect"*>'\_123' is correct variable name.click to see it's

value.</p>

<p id=*"correct"*>'123here' is not a correct variable name!</p>

<p id=*"undefined"*>click to see the value of undefined variable

'check'!</p>

<p>Click below button to see values for each of above statements!</p>

<button onclick="myfunction()">click me!</button>

<h1>convert height in inches!</h1>

Height in Ft :

<input id=*"height"*>

<p>Result is :</p>

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

<button onclick="convert()">click me!</button>

</body>

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

**var** name = "Hardik Shah"; //global variable

document.getElementById('global').innerHTML = "global varliable value : "

+ name;

**function** myfunction() {

//local variables!

**var** name1 = "Engineer Hardik";

**var** \_123 = 123;

**var** check;

document.getElementById('local').innerHTML = "local varliable value : "

+ name1;

document.getElementById('incorrect').innerHTML = "'\_123' varliable value : "

+ \_123;

document.getElementById('undefined').innerHTML = "'check' varliable value : "

+ check;

}

**function** convert() {

**var** h = Number(document.getElementById('height').value);

**if** (isNaN(h) || h == **null** || h == "") {

alert("incorrect input!");

location.reload();

**return** **false**;

} **else** {

**var** result;

result = h \* 12;

document.getElementById('result').innerHTML = "Converted height in inches is : "

+ result;

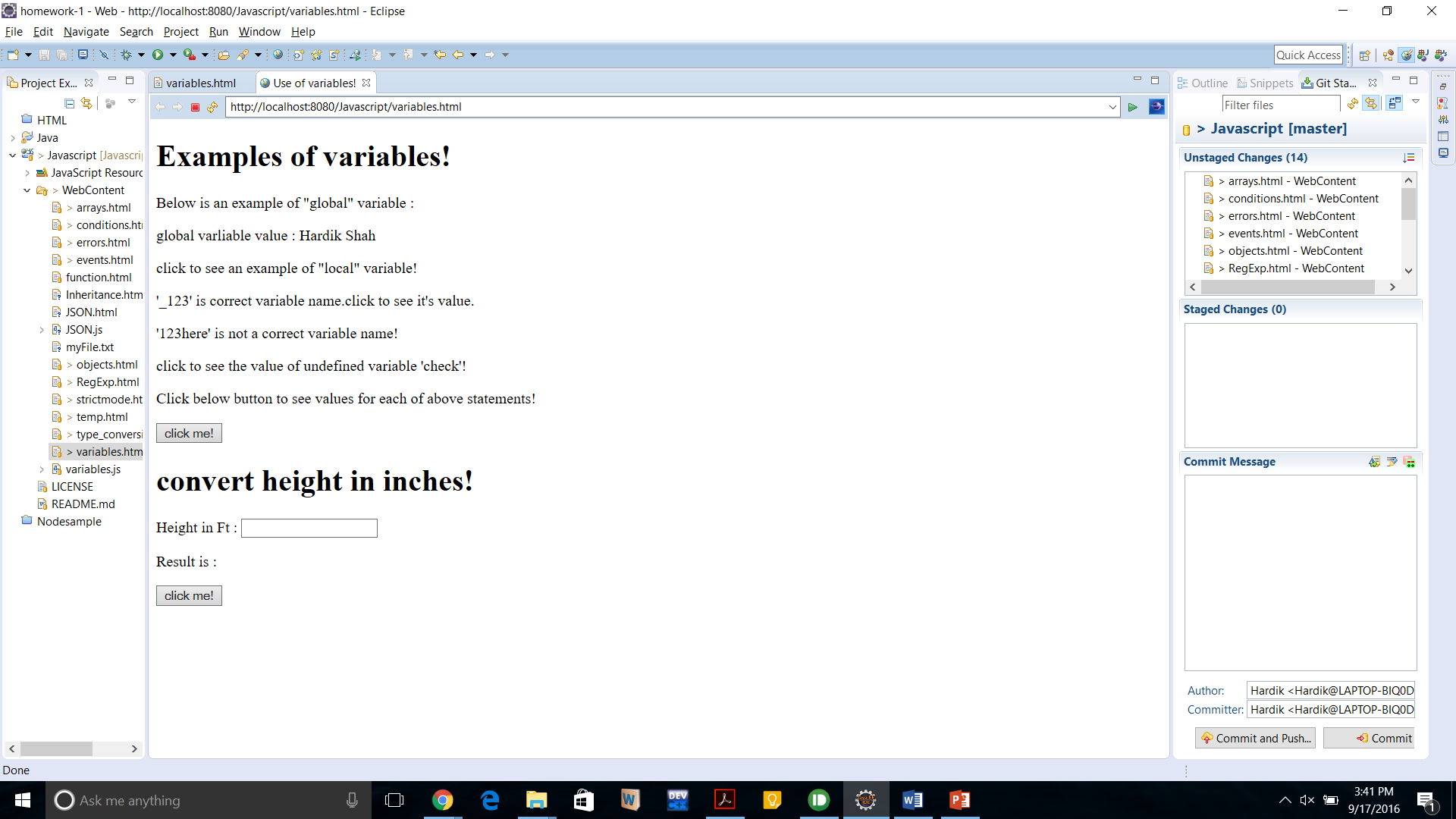
}

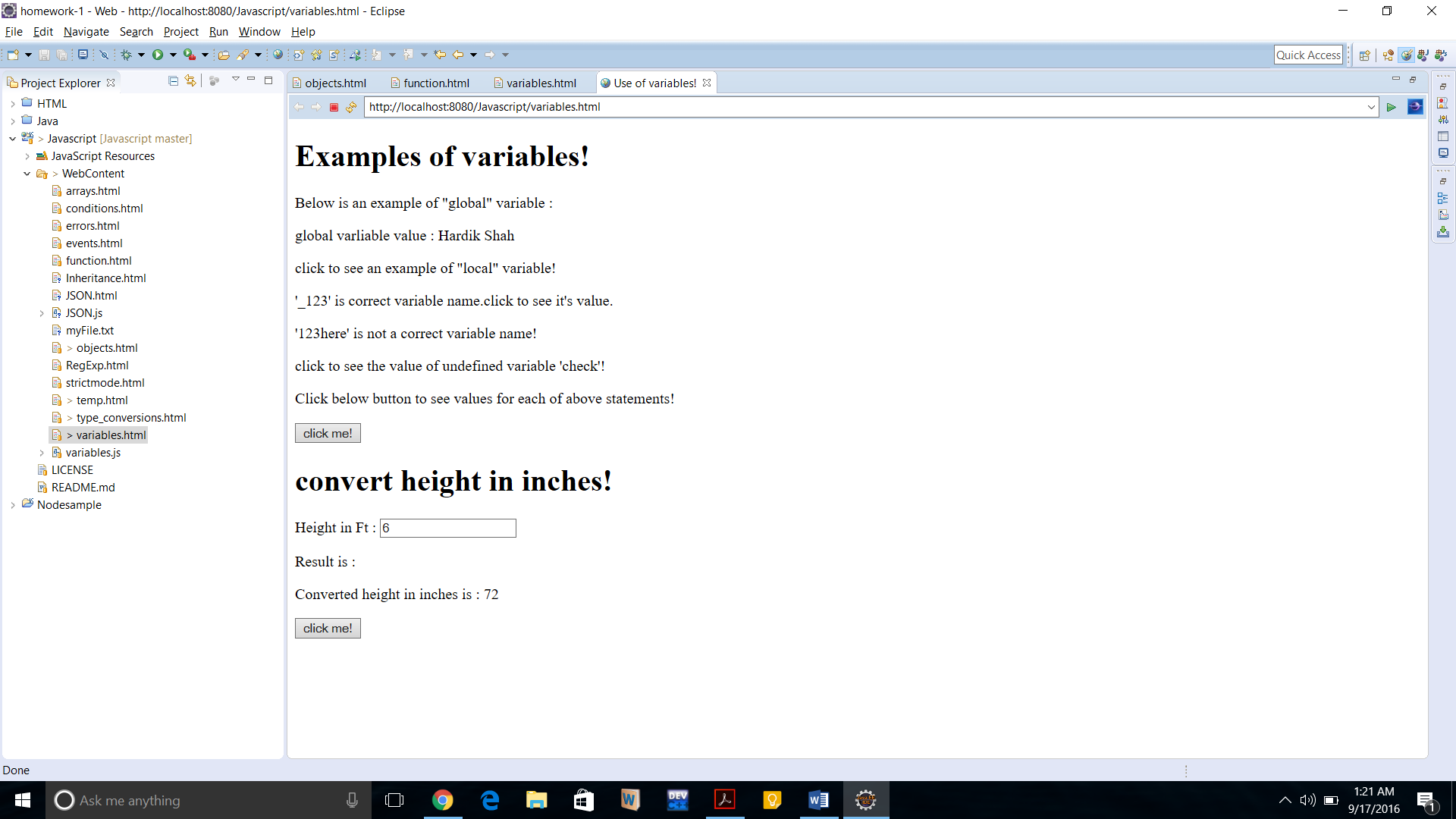
}

</script>

</html>

Output:





2)**Objects:**

Javascript is an object-oriented language. Objects are made up by attributes. If, attribute is a function than it’s called as method or it’s called as property of that object.

The ***new*** operator is used to create object’s instance. String, Array, Date, Math, RegExp all these are typical objects example.

We can also create Boolean, string or number objects using new keyword followed by type of object.

Below are some typical example of objects :

var name = new Object();

var day = new Date();

var SJSU = new Array[“Computer Engg”, “Electrical Engg”,”Industrial Engg”];

We can access the objects passes in parameter of methods by using ‘this’ keyword.

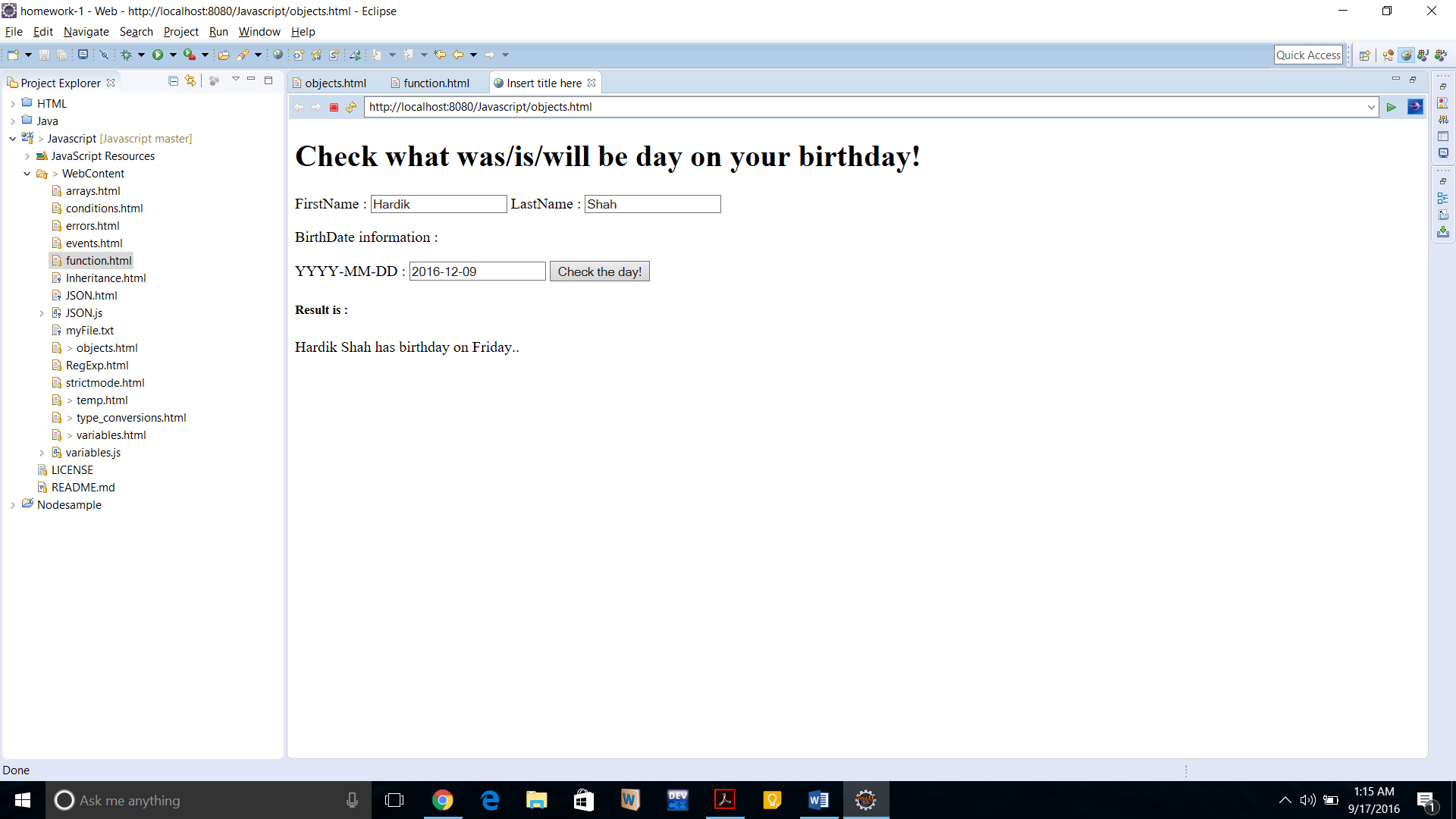
**Programming Question:**

Design a HTML page which will ask user his/her firstname and lastname. Also,his/her birthday which may be upcoming or past! Depending upon inputs, program should respond with the day on which his/her birthday was/is coming.

**Code :**



**Output:**



**3)** **Functions:**

Function is a body in which the block of code is written to reuse it anywhere within the programs at any number of times.

We can define functions in JS by using function keyword followed by unique function name, a range of parameters (if any), and a block surrounded by curly braces.

If you want to return the result executed by that function, then you may want to include return statement followed by that result parameter.

The example of a typical function prototype is as follow:

function myFunction(param1, param2){

var result = param1+param2;

return result;

}

**Programming Question**:

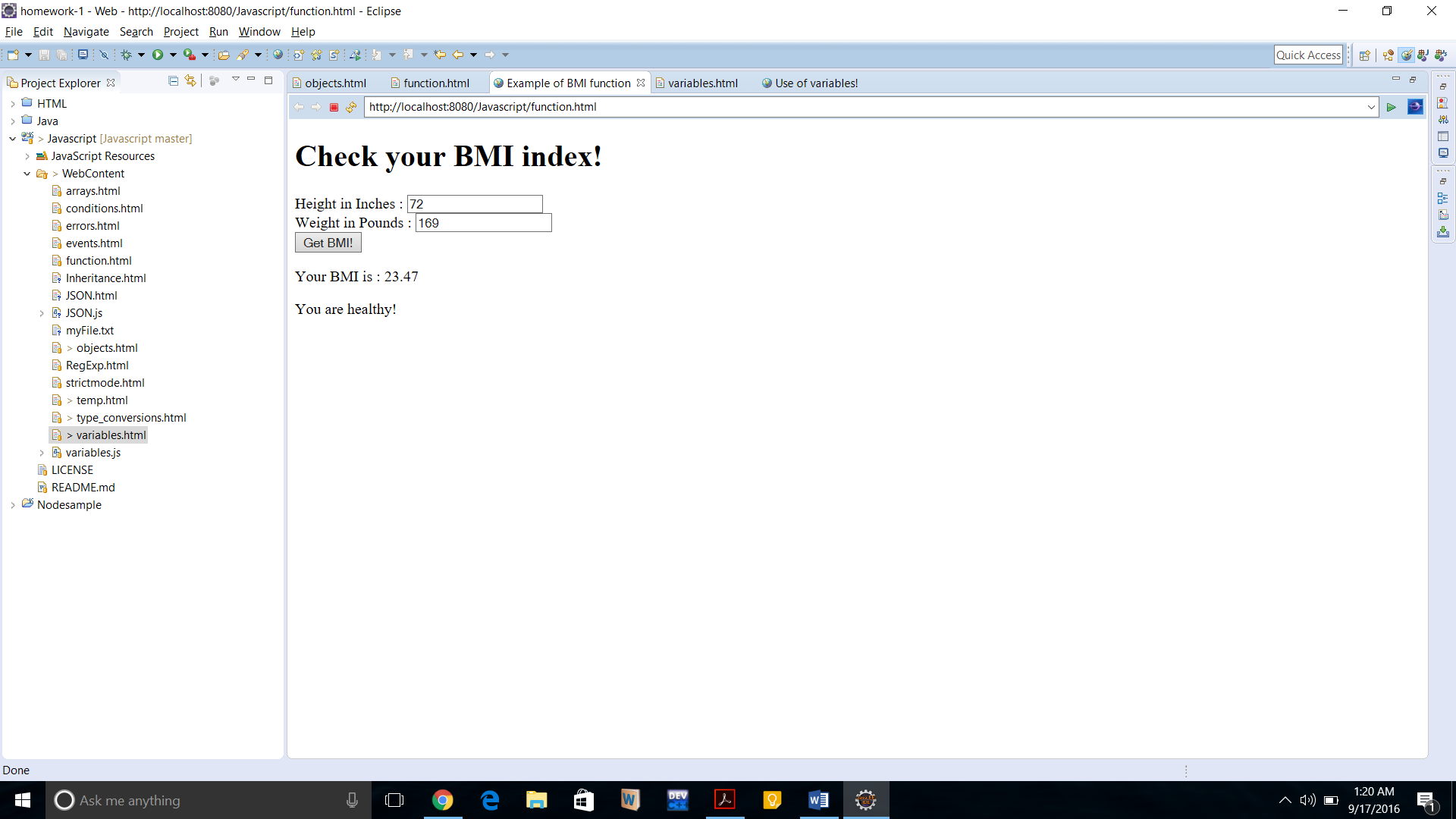
Check user BMI index. User will input height in inches and weight in pounds.

**Code:**





**Output:**



**4)** **Events:**

The event tag in javascript is used to manipulate an HTML content by user defined actions.

This means, it can react on users input at HTML content or an event can be set to display a message or react with user defined actions while loading or navigating between the pages!

Below example demonstrate how events can be declared in HTML entries and later can be invoked by user inputs to do set of actions associated with that event!

<button type=”button” onclick=”myFunction()”></button>

In above example, the onclick is an event invoked when user hits the button and then it will fire the myFunction() to do the things defined inside to that function.

There are many events associated provided by JavaScript which are listed below:

onchange

onclick

onmouseovr

onmouseout

onkeydown

onload

ondbclick

onload

onscroll

onerror

onfocus

onblur

onsubmit

ondrag

And tons more!

**Programming Question:**

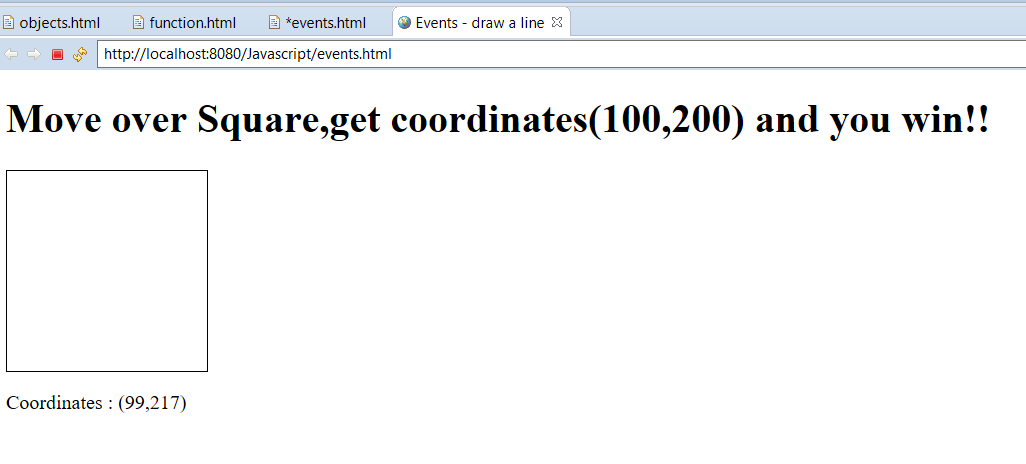
Build a small game in which there is a square box and user will move its screen cursor inside that box. Whenever he will more over the cursor, the coordinates will be updated and shown real time using events. Once user will move the cursor to perfect (100,200) mark, he will win the game and some image should appear to indicate he has won the game.

**Code :**



**Output:**

In process:



Once hit to (100,200) then the output:



**5)** **Arrays:**

Javascript arrays are used to store many values in a single variable.

JS Array is also an object type.

For example :

Var names = [“Hardik”,”Keyur”,”kalgi”];

We may access each element in the above array by their index positions. Meaning names[0] will give output as Hardik and likewise other entries can be accessed.

Arrays have their properties associated with them which are: Constructor, length, index and input.

There are several methods which are inbuilt with an Array declaration and can be used by programmers. Some of them are:

Concat(), forEach(),inderOf(),pop(),push(),reverse(),sort(),toSting() which are been used frequently by programmers to perform specific tasks on arrays.

**Programming Question:**

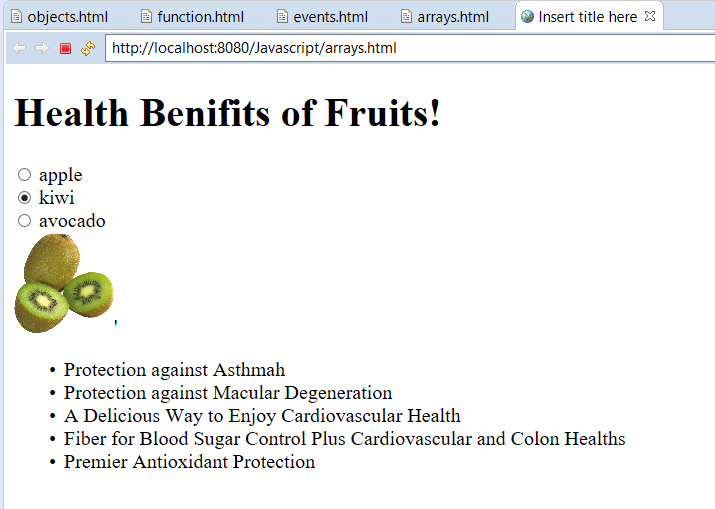
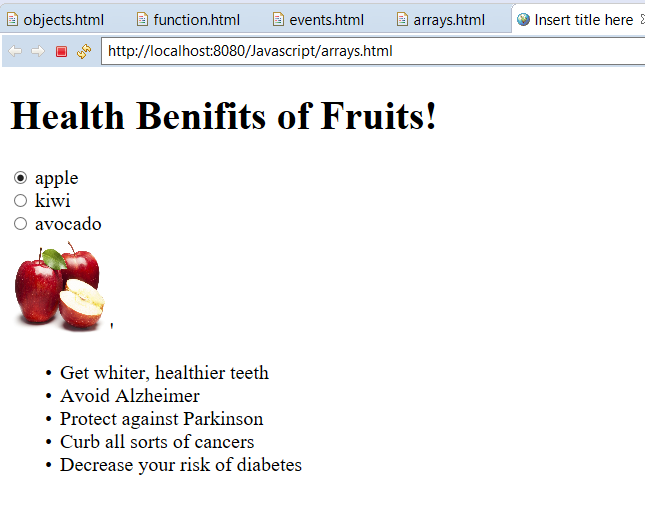
Make a html program in which some fruits are listed and clicking on individual fruit will enlist its health benefits. Use arrays for to achieve this functionality.

**Code:**

********



**Output:**



**6)** **Inheritance:**

Inheritance is to use the block of code of other class / object without writing it again.

Since JavaScript supports object oriented approach, one of the classics approach is inheritance. Since Js doesn’t have classes, it allows an object directly inherits from another object.

We may inherit properties, variables or functions directly from parent objects to child objects!

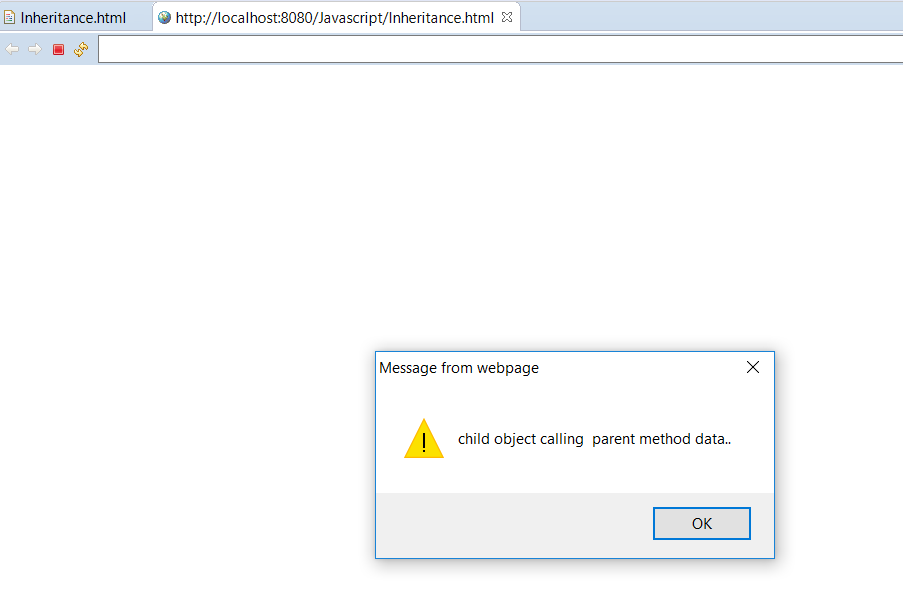
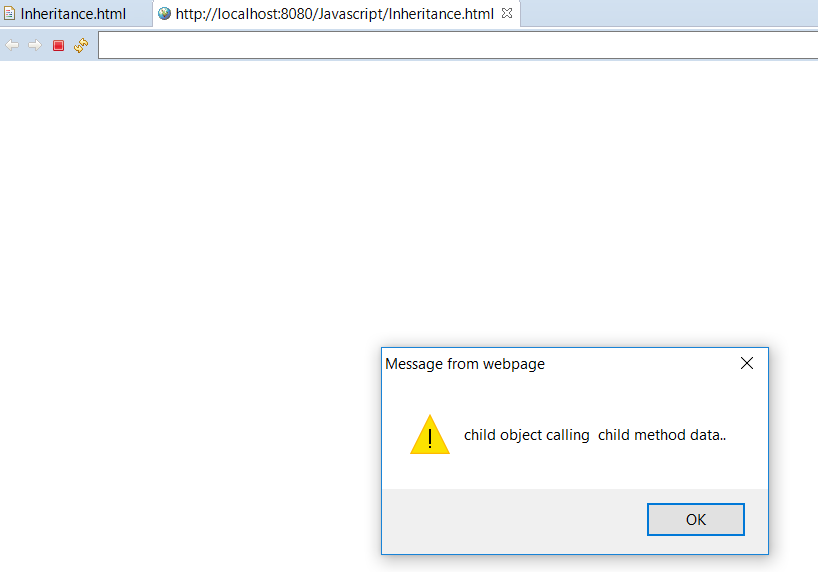
**Programming Question**:

Write a program to send alert to browser whenever function enters in parent class and child classes to understand the flow of inheritance supported by JavaScript.

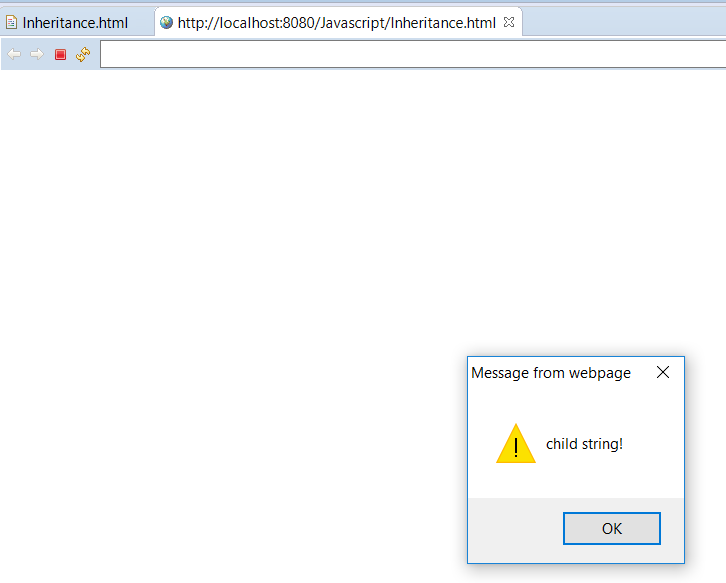
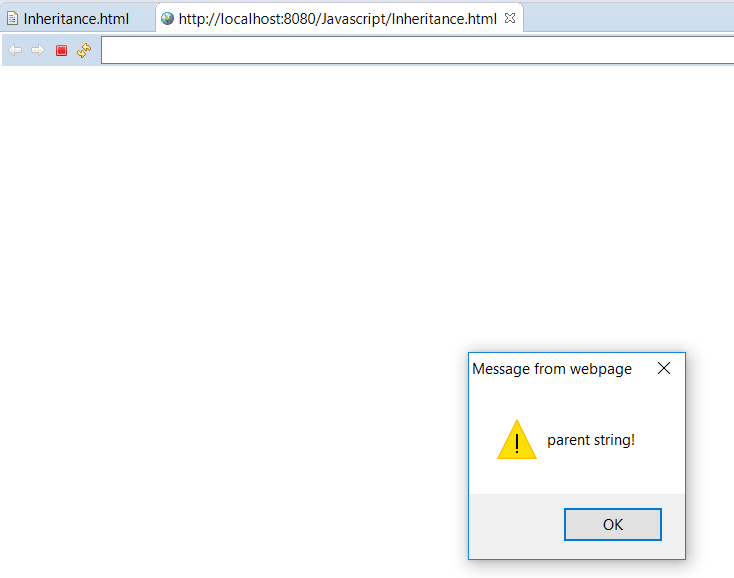
**Code:**



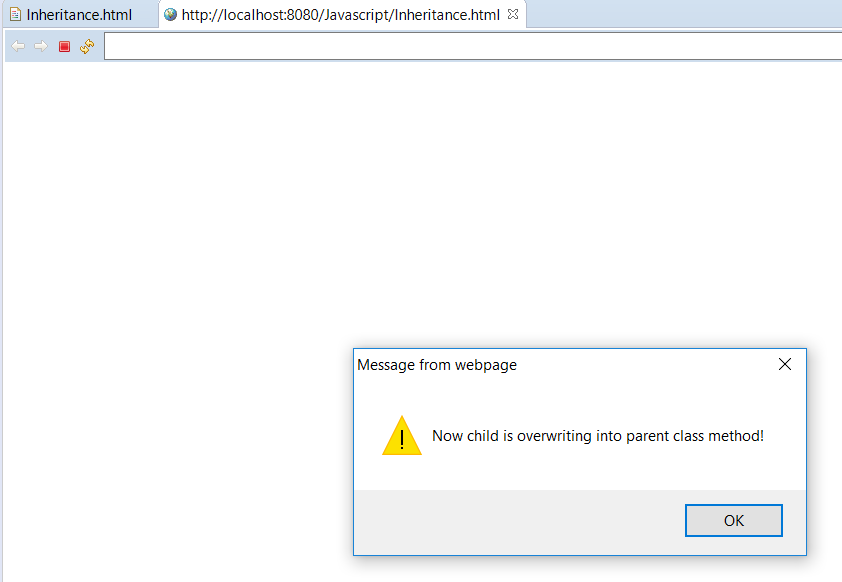
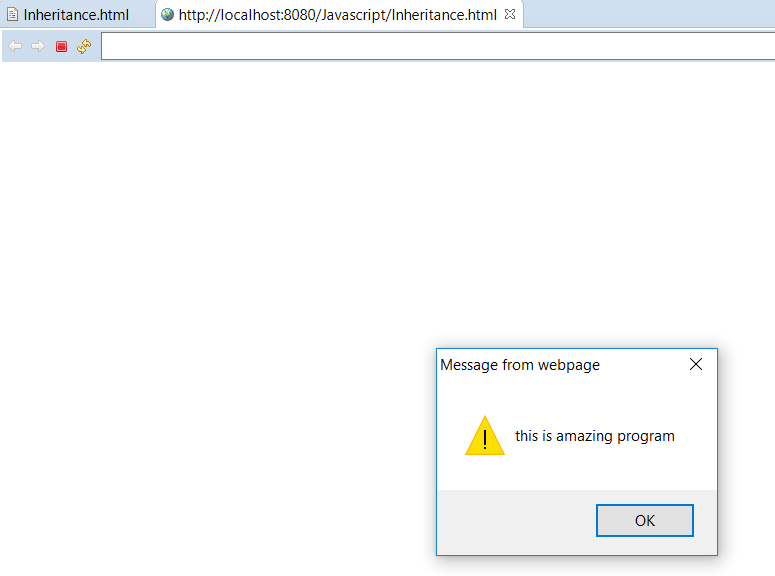
**Output: 1) 2)**

3) 4)



5) 6)

**7)** **Conditions:**

Js conditions are used to perform set of actions depending upon the different decisions/choices.

In JavaScript, we have major four types of conditional statements:

* if
* If..else
* Else if
* Switch..case

For example:

If (condition is true or condition matches with some expression) {

//Perform this task!

} else{

//perform this task if condition is not true!

}

Switch(expression) {

Case n:

Code

Break;

Case n:

Code

Break; …………………….

}

**Programming Question:**

Write a program to check if the user can enter and match the given capcha code or not. Also, the capcha code should be randomly generated each time when user generate the capcha code.

Code: 

**Output:**



**8)** **Regular Expression**:

Regular Expression (RegExp) expressions are useful when programmers want to perform some pattern matching operations or search the block of pattern and then further replace or manipulate it with something else.

The format of RegExp is: /pattern/modifiers

Modifiers can be of three types: I(case-insensitive search) or g(global match – find all matches in the pattern) or m(to perform multiple matching)

Brackets are used to find the desired range of characters :

[123] – find anyone of the letter withing the bracket.

[^123] – find other than letters not in between the brackets.

[1 | 3] – find any of the alternatives

Likewise, we can do the above operations on characters.

Besides above, there is a wide variety of match-makers and Quantifiers can be used by programmer depending upon the exact requirement.

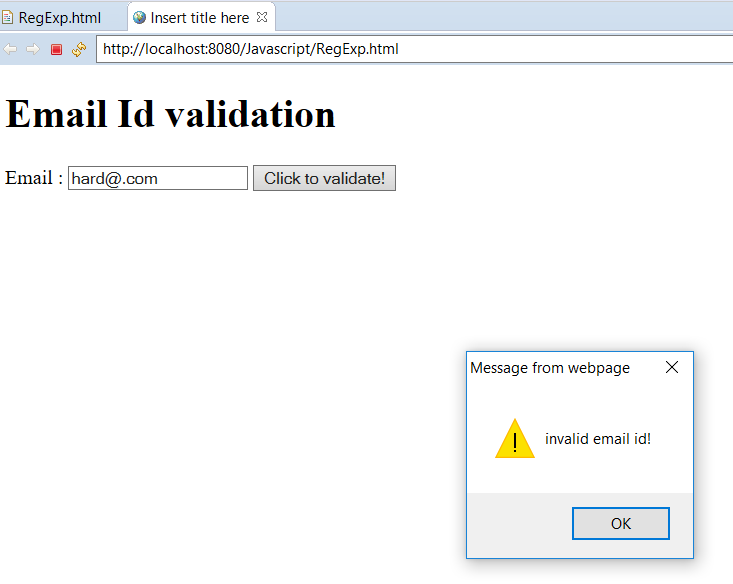
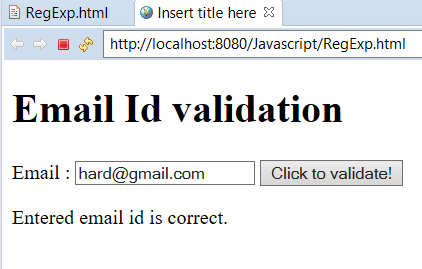
**Programming Question**:

Write a program to validate the email id entered by user.

**Code**:



**Output:**

**9)** **Strict Mode:**

The purpose of strict mode is to indicate that the code should be executed in ***‘strict mode’***

We can declare it by ‘use strict’ in the beginning of the script (global scope) or a function.

If script is defined in strict mode than, programmer can’t omit syntax mistakes in programs like not defining var for variables or duplicating parameter in a function or anything which is against rules of language will cause execution stop and console will throw an error log.

This is a good practice when the code should be free from any type of syntax errors.

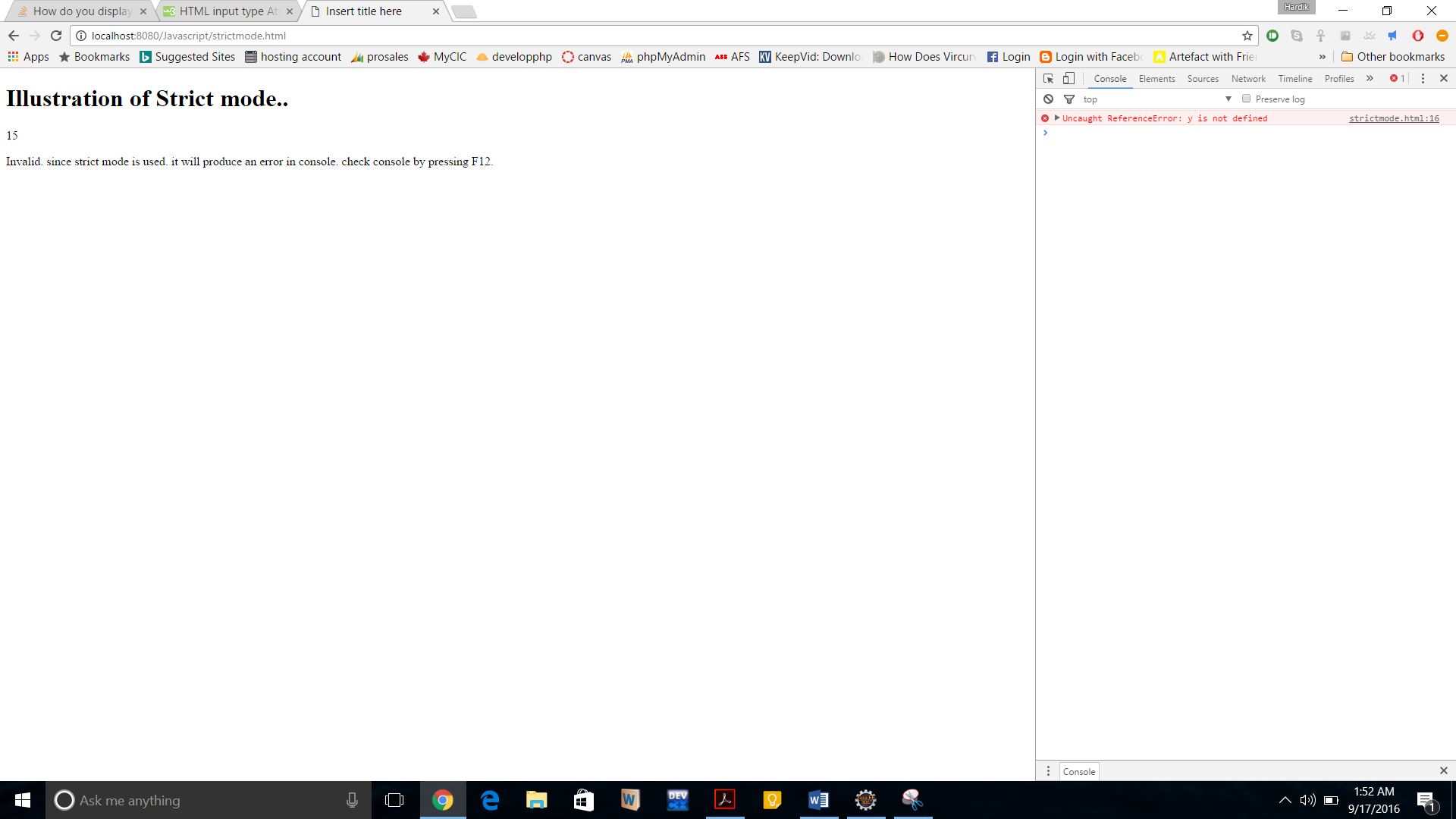
**Programming Question:**

Demonstrate how the use of strict mode produces an error on console!

**Code:**



**Output:**



**10)** **Errors:**

Errors can happen due to syntax or due to wrong input or any other unanticipated flow of program!

To access these errors, Js has provided four major statements:

try: it will allow to test a block of code written inside the try.

catch: this will allow to handle the error.

throw: to create user defined errors.

finally: it will allow to execute the code regardless of if any error in try and catch block. These error handlers are very useful for identifying and handling errors occurs in Js programs.

**Programming Question:**

Take input from user and with the use of Try / Catch, produce different results according to the age user has entered! This is for demonstration purpose of how we can handle errors or code in our desired way.

**Code:**



Output:



**11)** **Type Conversions:**

In JavaScript there are various types of data types and objects.

With the help of type conversion feature, variables can be converted to new variables or datatypes.

Some typical methods are:

toFixed() : returns a string with number being rounded.

toSting() : covert variable to string.

Date().toString() : returns date in a string format.

To convert strings to numbers :

Number(“1.14”) : returns 1.14

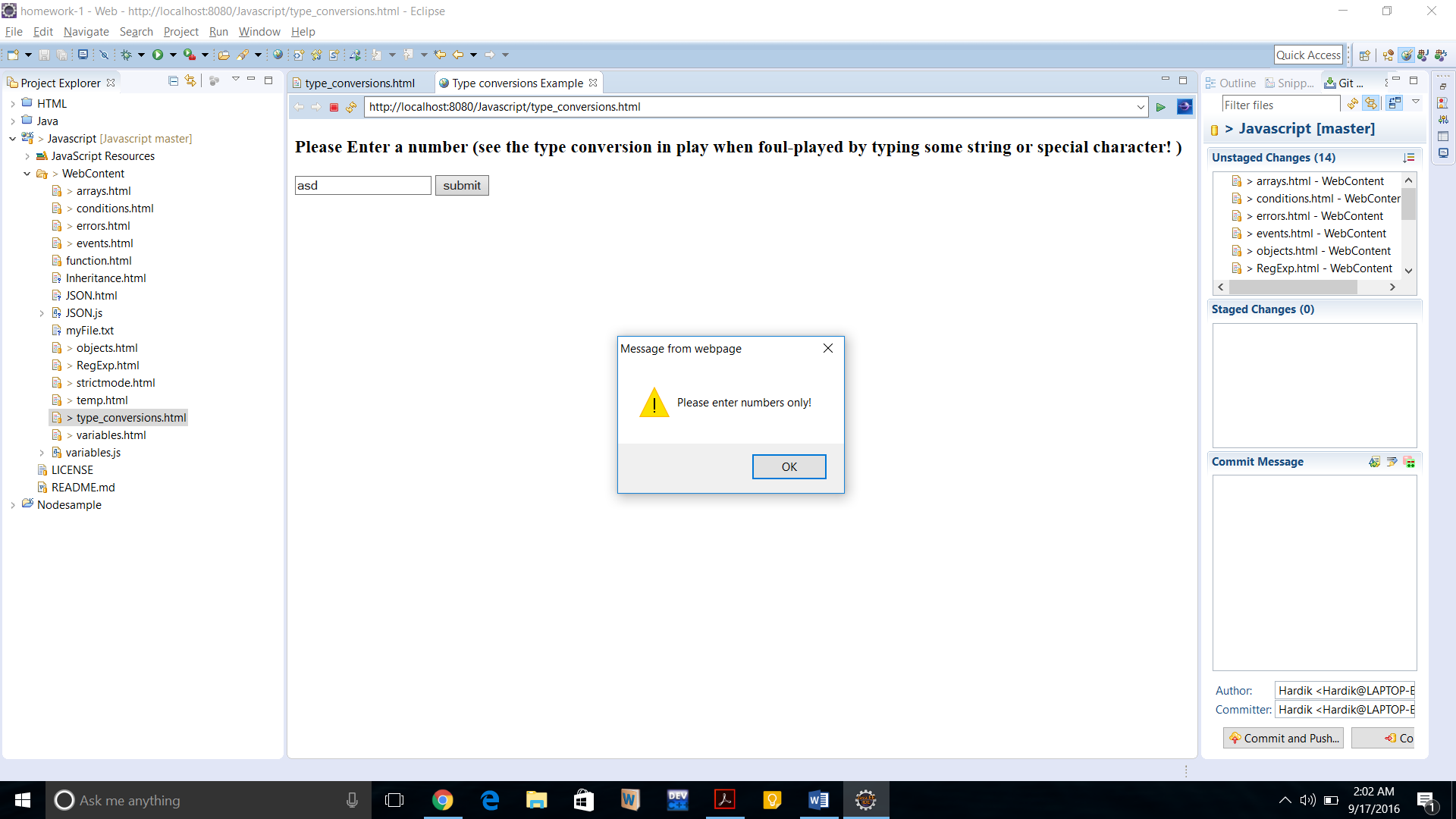
**Programming Question**:

With the use of type conversion, check if the given input is number only! In case of input is not number, alert the user!

**Code:**



Output:



12) **JSON**:

JavaScript Object Notation (JSON) : it’s a syntax for storing and exchanging information (data)

JSON is an alternative to XML. JSON is good since it’s lighter and easier to use and it’s format is identical to the JS objects.

JSON.Parse is used to parse the JSON string.

JSON is always written as name / value pairs.

JSON values can be a number or a string or an array or an object.

JSON files are stored with an extension of “.json”

**Programming Question:**

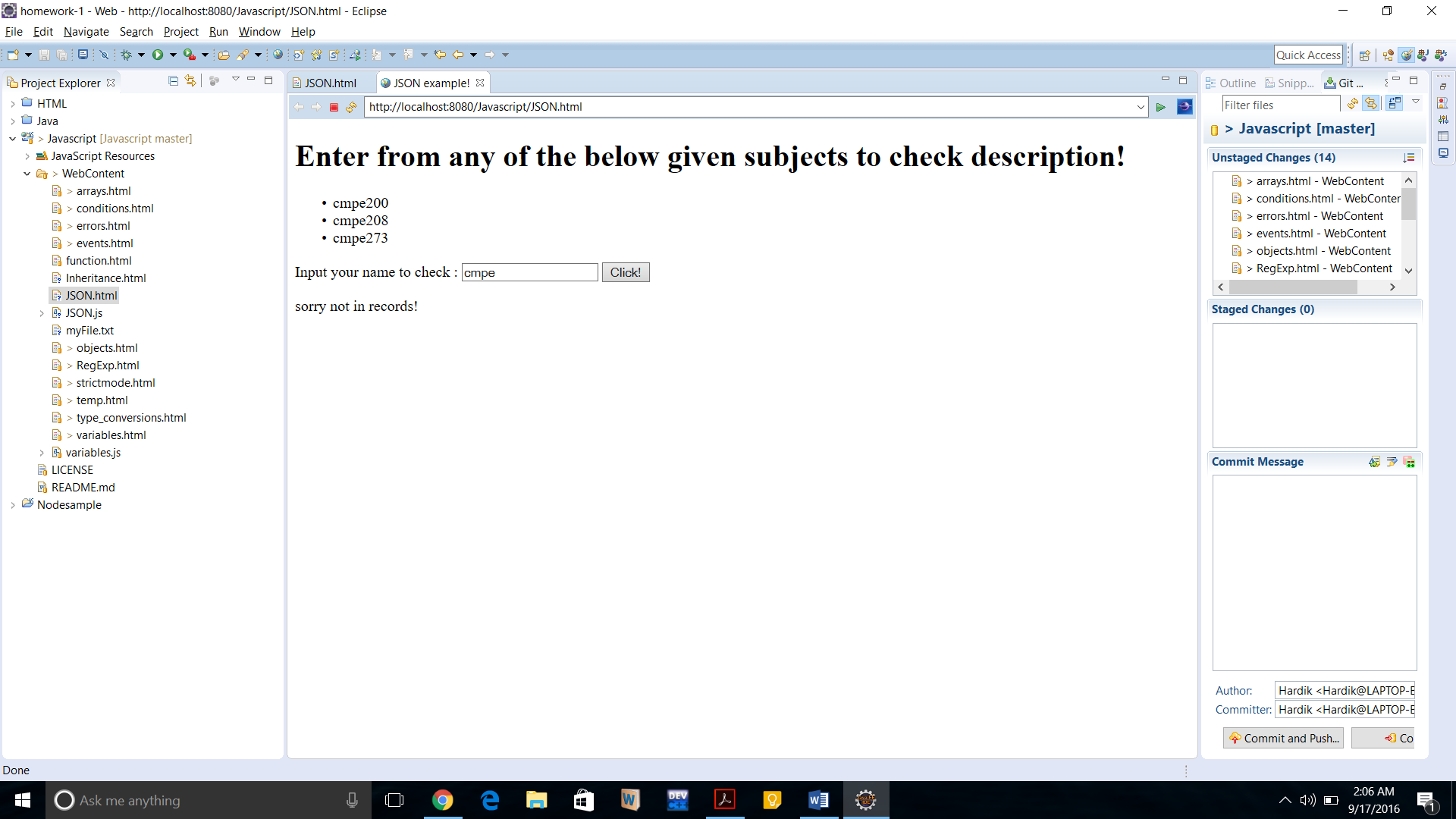
With using JSON, ask user to input one of the three listed subject codes and if it matches he entry stored in program, then print that subject name which correspondence to the subject code else show an error message.

**Code:**



Output:





**HTML5:**

**1)** **Local Storage:**

This HTML5 feature lets user to store data locally within user’s browser.

Earlier (before HTML5) the data was used to be stored in a form of cookies and it had to be included in every server request hence, affecting performance and speed! Local Storage can contain large amount of data locally to speed up the performance for frequently accesses requests.

Local Storage objects can hold data without any expiry, this is in contrast to the session storage which can hold data up to one healthy session and erase the data when browser is terminated.

Syntax :

localstorage.setItem(“this”,”that for this”);

localstorage.getItem(“this”);

localstorage.clickcount; //to track number of times, clicks are being made.

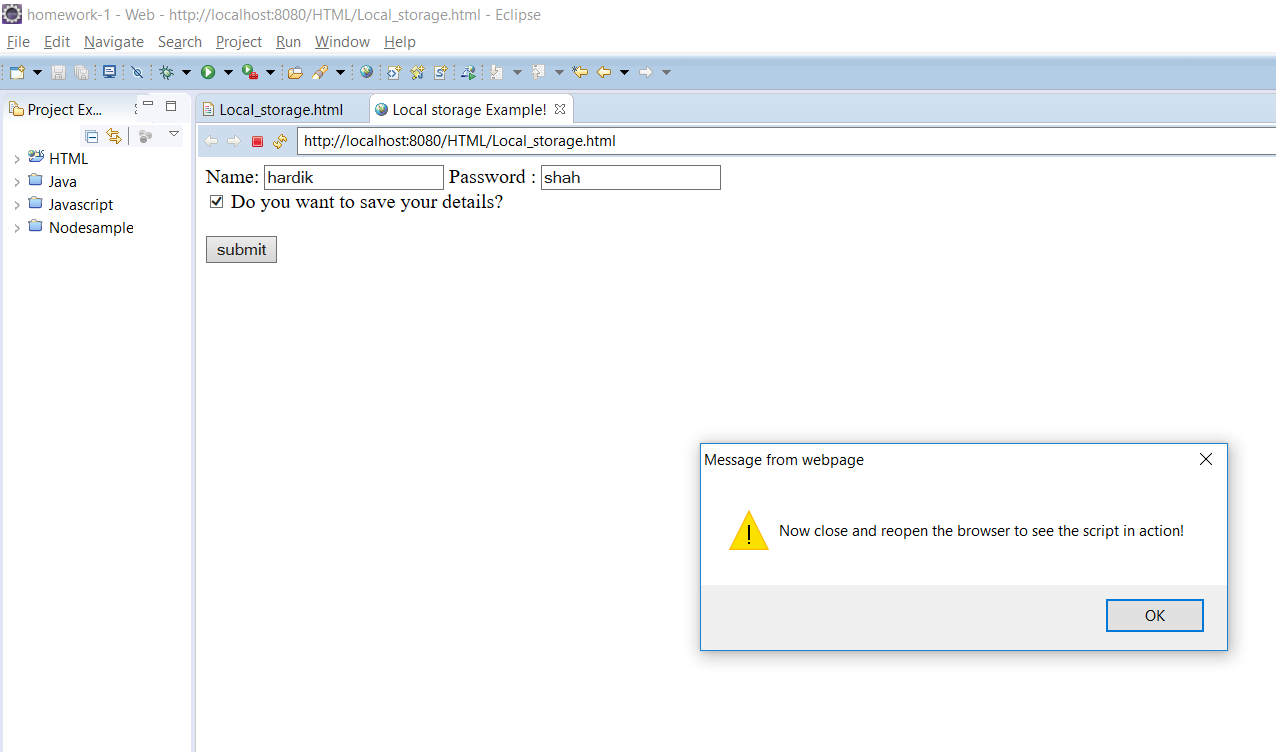
**Programming Question:**

Write a program which has a small HTML firstname and lastname fields and a checkmark box. If checkbox is checked then the user input data will be stored in local storage and if it’s not checked, the user data won’t be saved and earlier saved data should be cleared from cache.

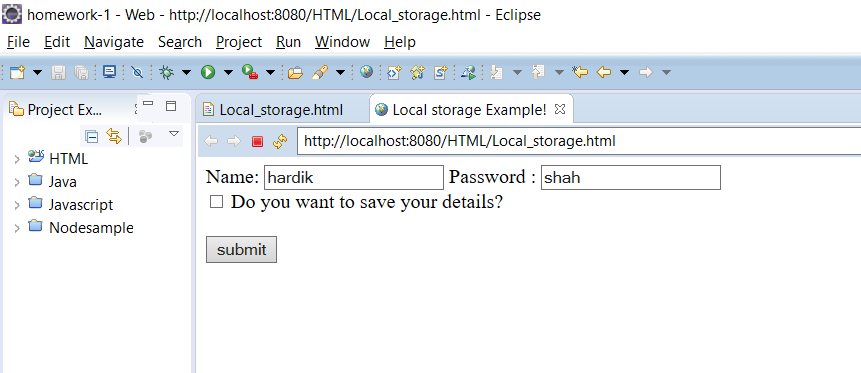
**Code:**

****

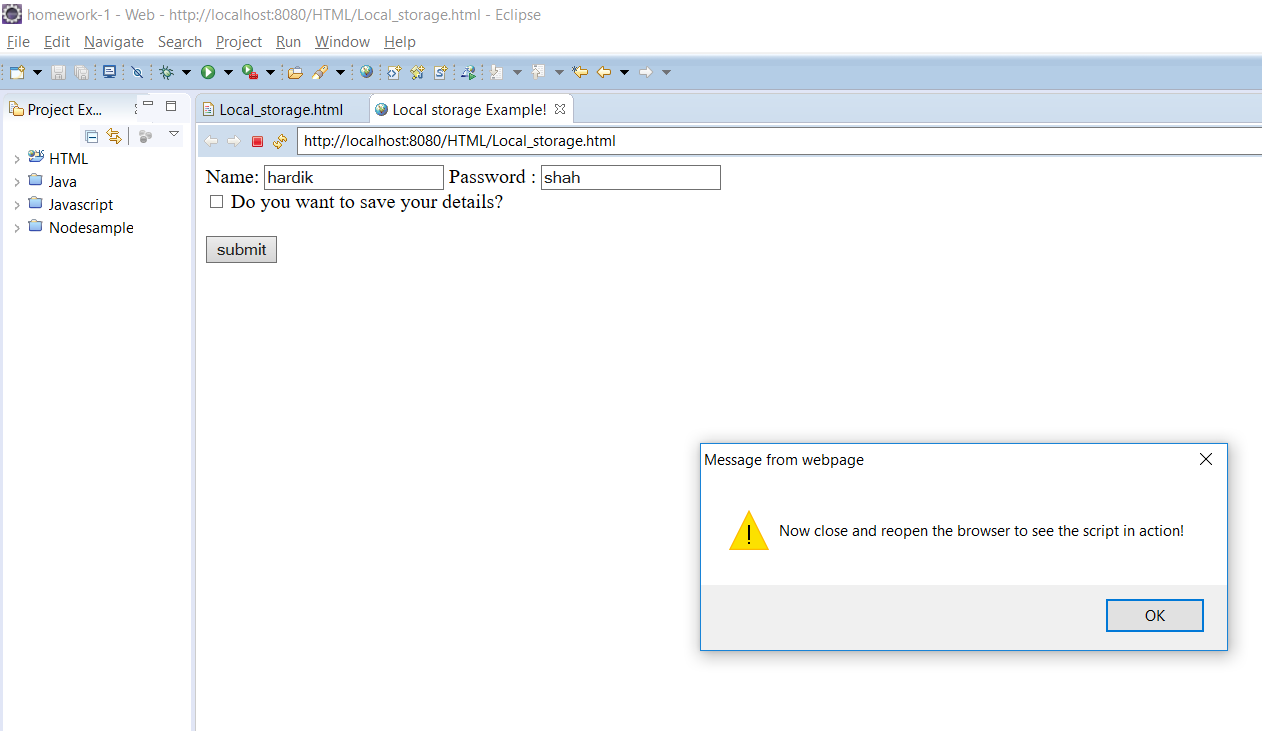
Output:



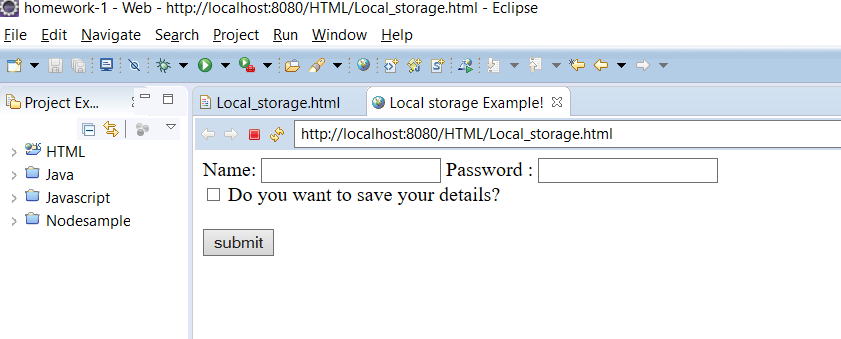
On next reload :



Now, let’s don’t checkmark the box:



After reload:



2) **Media (video and audio):**

In HTML5, the media – Audio and video are supported and HTML5 has methods, events and properties for media content.

Some methods are:

load(), play(),pause() etc.

Some properties are:

Autoplay,controls,duration,loop,played,volume etc.

Some of the events are :

Canplaythrough,abort,play,pause,seeking,volumechange,waiting etc.

<audio ></audio> for playing an audio file.

<video></video> for playing a video file.

<source> element allows you to specify the files which browser may choose from.

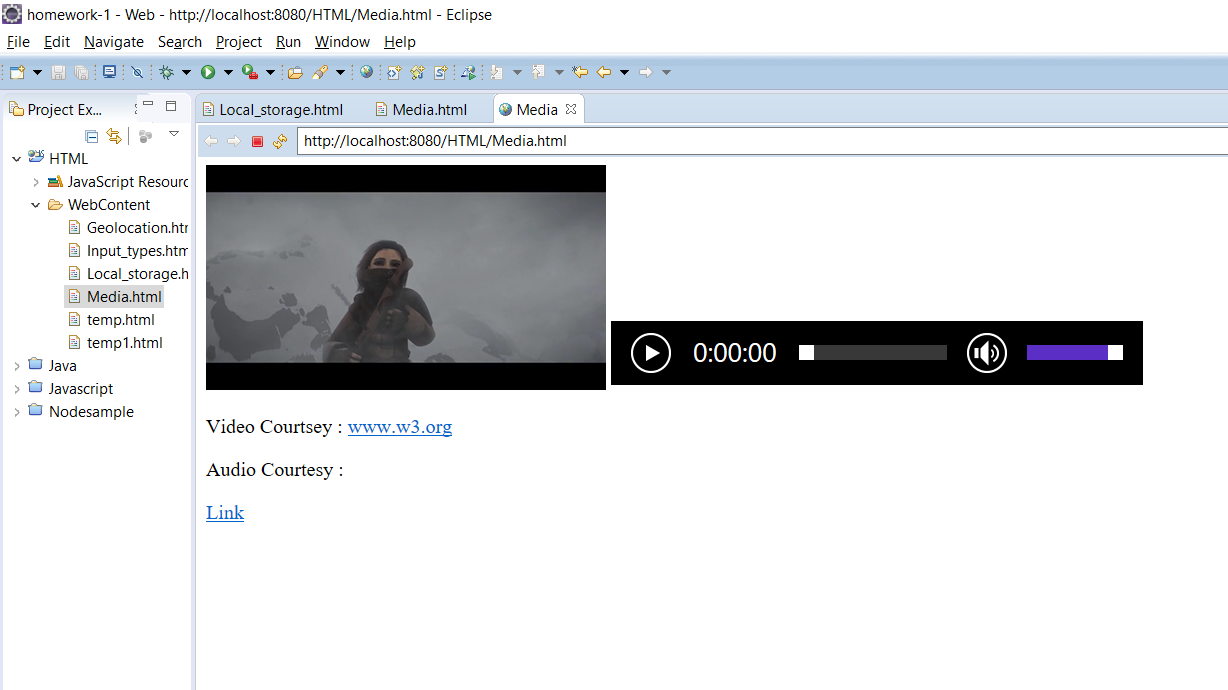
**Programming Question:**

Display a sample video clip and a audio clip with controls using HTML5 media tags. Code:

**Code:**



Output:



3) **Input Type**:

There are various input types being supported by HTML5.

It can be defined by: <input type=”type of input defined by user!”>

Some of the types are highlighted below:

Pattern: specifies a RegExp to check the input value with the expression.

autofocus: defines which input tag should be focused while on page load!

required: with this tag, the input field won’t allow blank input from user

email: with this, the input entry can only take emails format and it will auto validate the email id.

**Programming Question:**

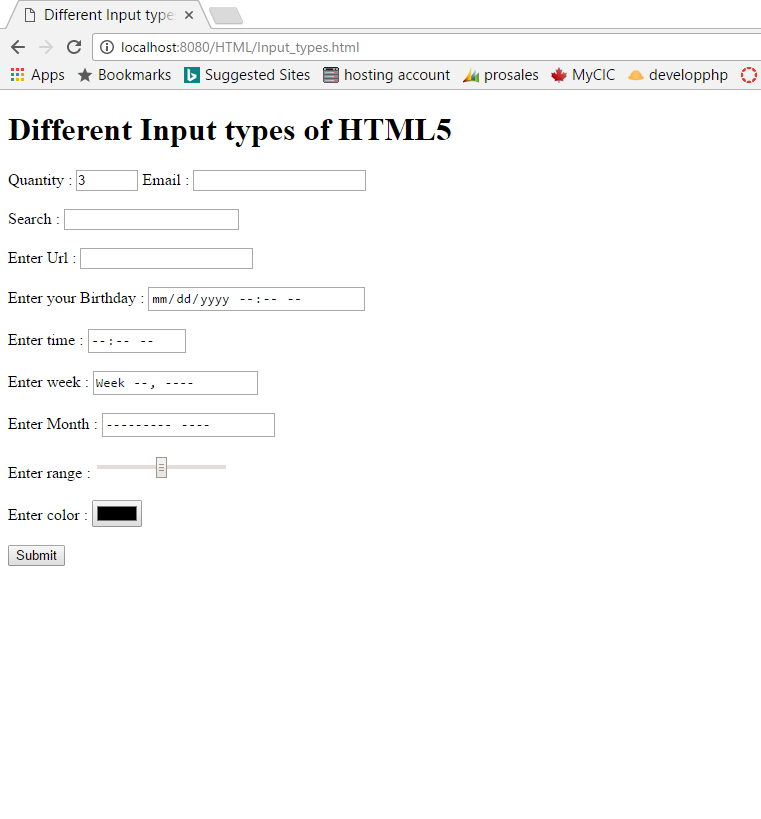
Write a program to render HTML5 various input types to show the functionality of each individual input types written inside the from.

Code:



**Output:**

Since, Output in Eclipse browser doesn’t support input types of HTML5, output snap is taken from Chrome browser.



4) **Geolocation:**

This HTML geolocation API is used to locate user’s coordinates – Longitude and Latitude details.

The function navigator.geolocation.getCurrentPosition will return longitude and latitude data which can be used further to implement user defined logic.

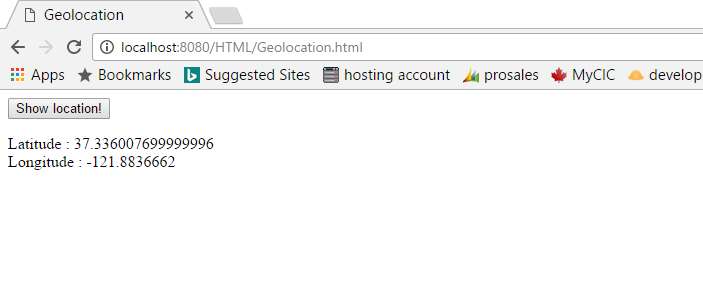
**Programming Question:**

Write a program to show user’s coordinates – Longitude and Latitude.

**Code:**



**Output:** Since, Output in Eclipse browser doesn’t support input types of HTML5, output snap is taken from Chrome browser.



**JAVA:**

**1)** **Queues:**

Just like a queue in real world, the queue types of data structure allow elements to be inserted at one end and let the elements removed from other end.

This means the queue data structure supports FIFO arrangement.

By use of Java.util.Queue interface which is a subtype of java.util.Collection interface, programmer can access the default functions associated with the queues which are,

queue.add : to add element in queue

queue.remove() : to remove/delete an element

poll() : retrieves and removes head of the queue element.

peek() : retrieves but not does not delete head of the queue.

**Programming Question:**

Write a program which will take any integer number and return a sequence of corresponding decimal binary numbers.

**Code:**

//Generate binary numbers!!

package edu.sjsu.cmpe273;

import java.util.LinkedList;

import java.util.Queue;

public class queues {

public String GetBinaryNumbers(int n) {

String result = "";

Queue<String> q = new LinkedList<>();

q.add("1");

while (n > 0) {

String s1 = q.poll();

String s2 = new String();

s2 = s1;

result += s1 + " ";

q.add(s1.concat("0"));

q.add(s2.concat("1"));

n--;

}

// System.out.print(result);

return result;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

queues obj = new queues();

String s = obj.GetBinaryNumbers(10);

System.out.println(s);

}

}

**Junit Program:**

package edu.sjsu.cmpe273.test;

import static org.junit.Assert.\*;

import org.junit.Test;

import edu.sjsu.cmpe273.array;

import edu.sjsu.cmpe273.queues;

public class queuesTest {

queues obj = new queues();

@Test

public void test1() {

String res = "1 10 11 ";

assertEquals(res, obj.GetBinaryNumbers(3));

}

@Test

public void test2() {

String res = "1 10 11 100 101 ";

assertEquals(res, obj.GetBinaryNumbers(5));

}

@Test

public void test3() {

String res = "1 10 11 100 101 110 111 1000 1001 1010 ";

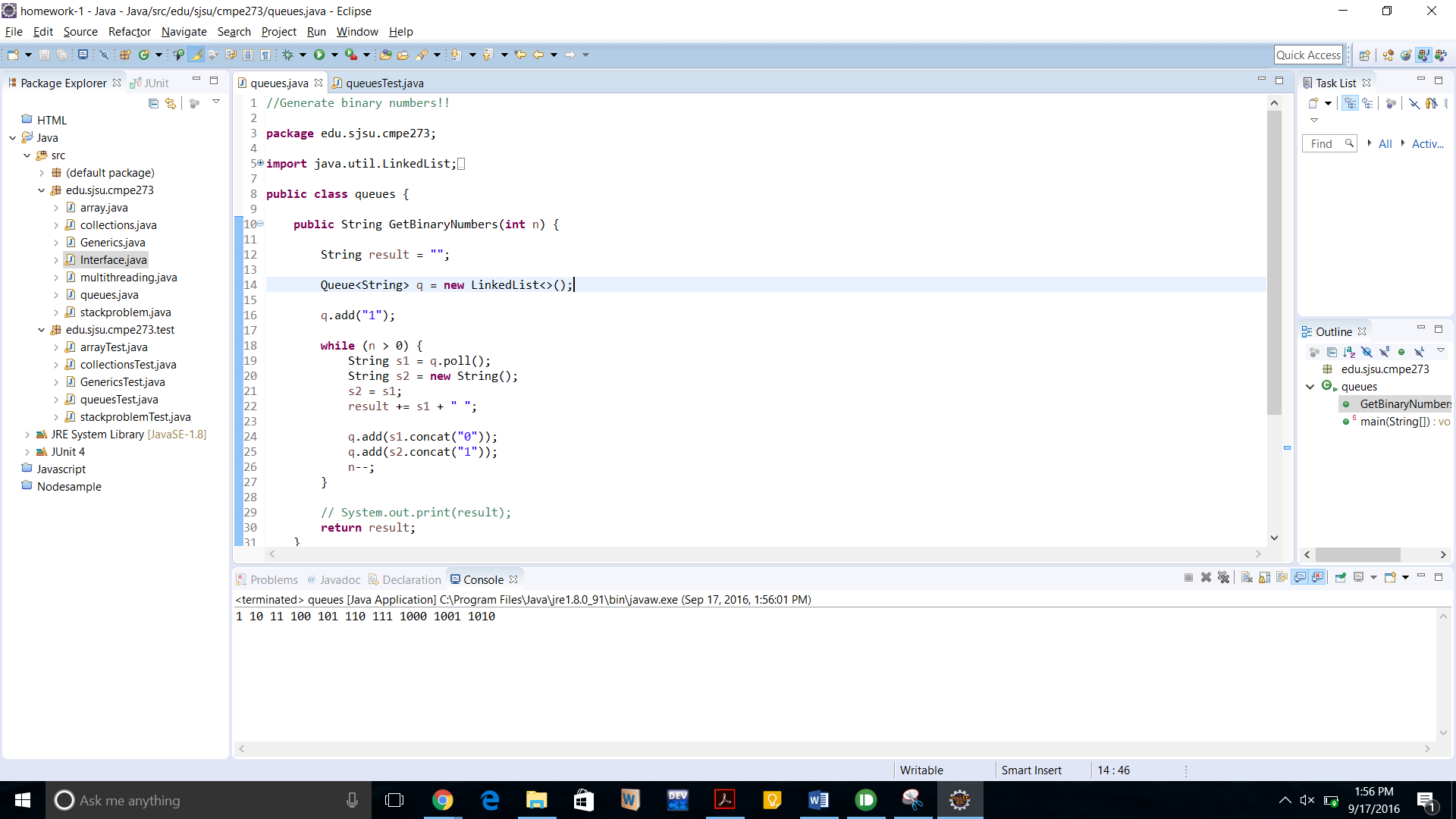
assertEquals(res, obj.GetBinaryNumbers(10));

}

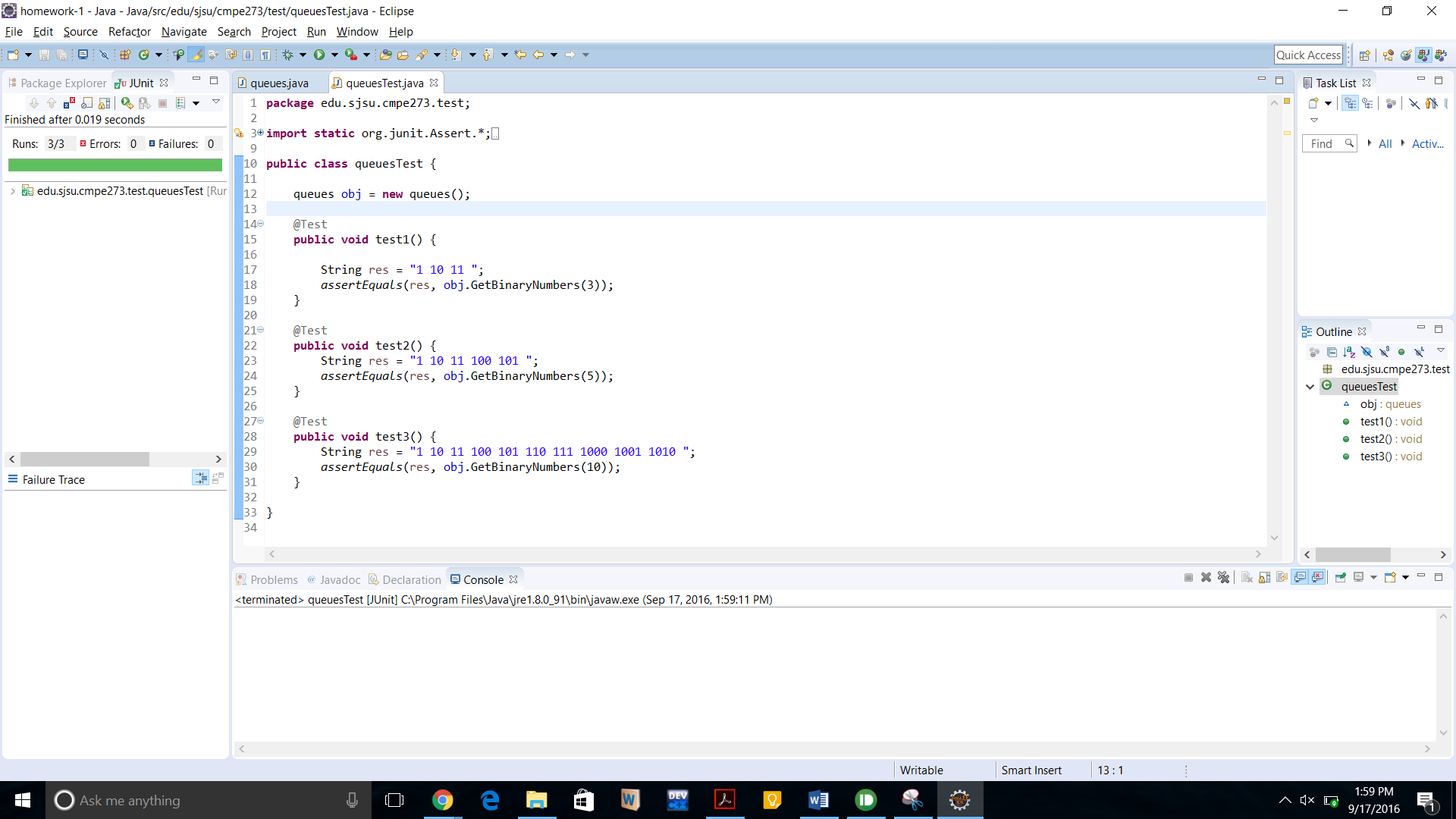
}

**Output:**

Output of a main function in Eclipse browser:



Output from Junit test cases:



**2)** **Stacks:**

Stack is also subset of collection interface which stores and retrieves elements in Last in First out (LIFO) fashion.

One of the typical use of stack is to reverse the contents of array elements.

Some methods associated with the stack are,

empty() : returns true if stack is empty.

Peek() – returns top element of the stack.

Pop() – returns and removes top element of the stack.

Push(object) – add element on stack.

Search(object) – searches the object within stack and if found, returns the offset value from the top of the stack.

**Programming Question**:

Write a program to check if a given string is palindrome or not?

**Code:**

Main Program:

package edu.sjsu.cmpe273;

//Check if a sting is palindrome or not!

import java.util.Scanner;

import java.util.Stack;

import org.junit.Test;

public class stackproblem {

public boolean palindorme(String s){

char[] reverse = new char [s.length()];

int j=0;

Stack st = new Stack();

for (int i = 0; i < s.length(); i++) {

st.push(s.charAt(i));

}

while (!st.isEmpty()) {

reverse[j++] = (char) st.pop();

}

String out = new String(reverse);

if(out.equalsIgnoreCase(s)){

System.out.println("Given string is palindrome!");

return true;

}else{

System.out.println("Given string is not palindrome!");

return false;

}

}

public static void main(String[] args) {

stackproblem sp = new stackproblem();

sp.palindorme("wow");

sp.palindorme("how are you?");

}

}

**Junit Program:**

package edu.sjsu.cmpe273.test;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.Test;

import edu.sjsu.cmpe273.stackproblem;

public class stackproblemTest {

@Test

public void test1() {

String s = "wow";

stackproblem obj = new stackproblem();

assertTrue(obj.palindorme(s));

}

@Test

public void test2() {

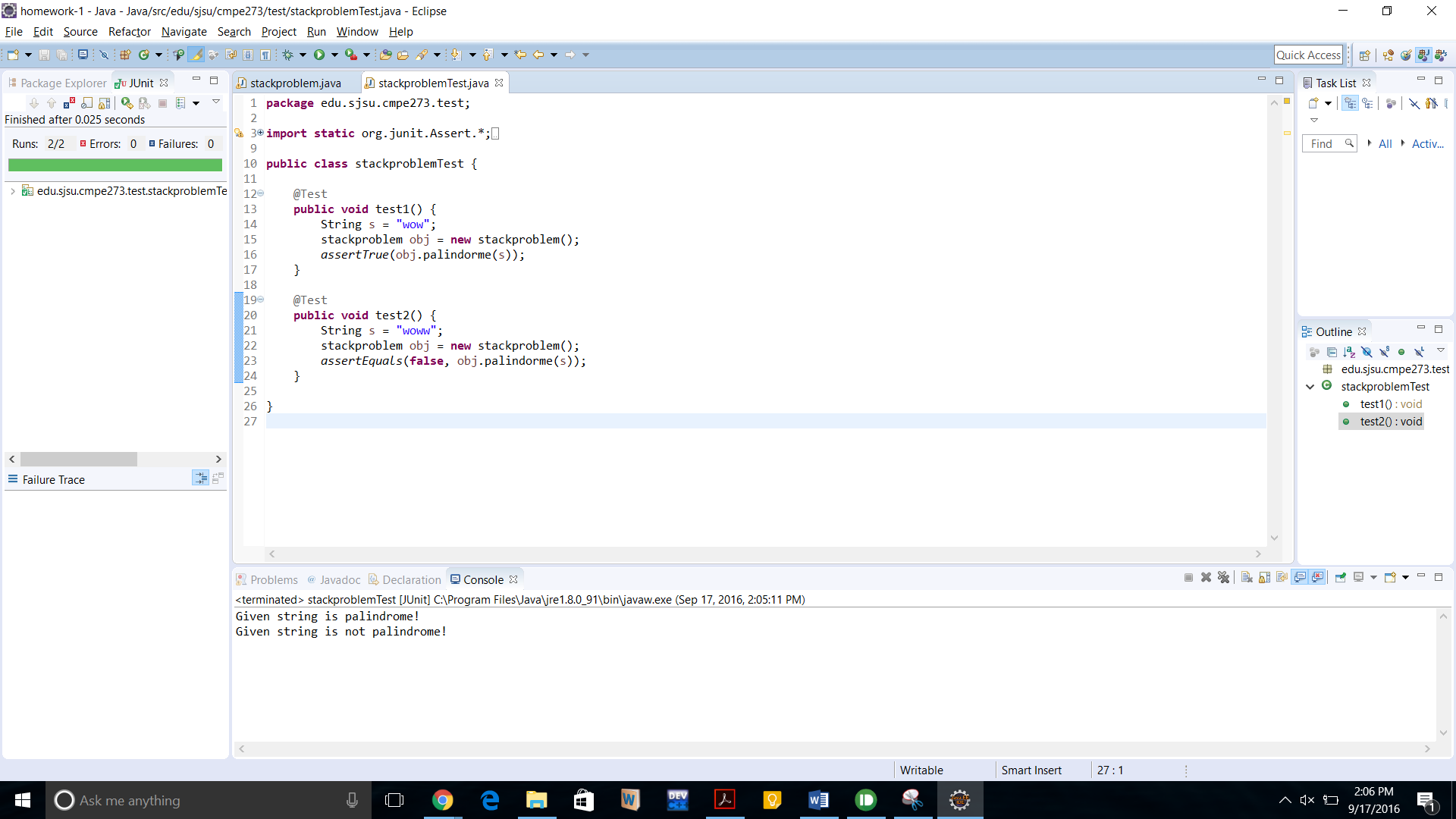
String s = "woww";

stackproblem obj = new stackproblem();

assertEquals(false, obj.palindorme(s));

} }

Output:



**3)** **Arrays:**

It’s a data structure to stores the collection of elements and it will be having the fixed defined size. This will allow to store multiple values under a common variable name rather than creating individual variables for each piece of data.

Declaring variables in java can be done through : int[] myList. We can specify any datatype depending upon the information to be stored in form of an array.

Int[] Samplearray = {1,2,3,4,5,6};

We can iterate through the whole array elements using for loops or iterators.

**Programming Question**:

Reverse an array with the use of java arrays.

**Code:**

**Main Program:**

//Reverse an Array!

**package** edu.sjsu.cmpe273;

**import** java.util.Arrays;

**public** **class** array {

**public** **int**[] max\_sort(**int**[] a) {

**int**[] max = **new** **int**[a.length];

Arrays.*sort*(a);

**for** (**int** i = 0, j = a.length - 1; i < max.length; i++, j--) {

max[i] = a[j];

}

**for** (**int** x : max) {

System.***out***.print(x + " ");

}

**return** max;

}

**public** **static** **void** main(String[] args) {

array arr = **new** array();

**int**[] a = { 1, 2, 3, 4 };

**int**[] res = **new** **int**[a.length];

res = arr.max\_sort(a);

}

}

**Junit test program:**

package edu.sjsu.cmpe273.test;

import static org.junit.Assert.\*;

import org.junit.Test;

import edu.sjsu.cmpe273.array;

import edu.sjsu.cmpe273.stackproblem;

public class arrayTest {

@Test

public void test1() {

int[] a = { 1, 2, 3, 4, 5 };

array obj = new array();

int[] res = { 5, 4, 3, 2, 1 };

assertArrayEquals(res, obj.max\_sort(a));

}

@Test

public void test2() {

int[] a = { 10, 2, 300, 43, 5000 };

array obj = new array();

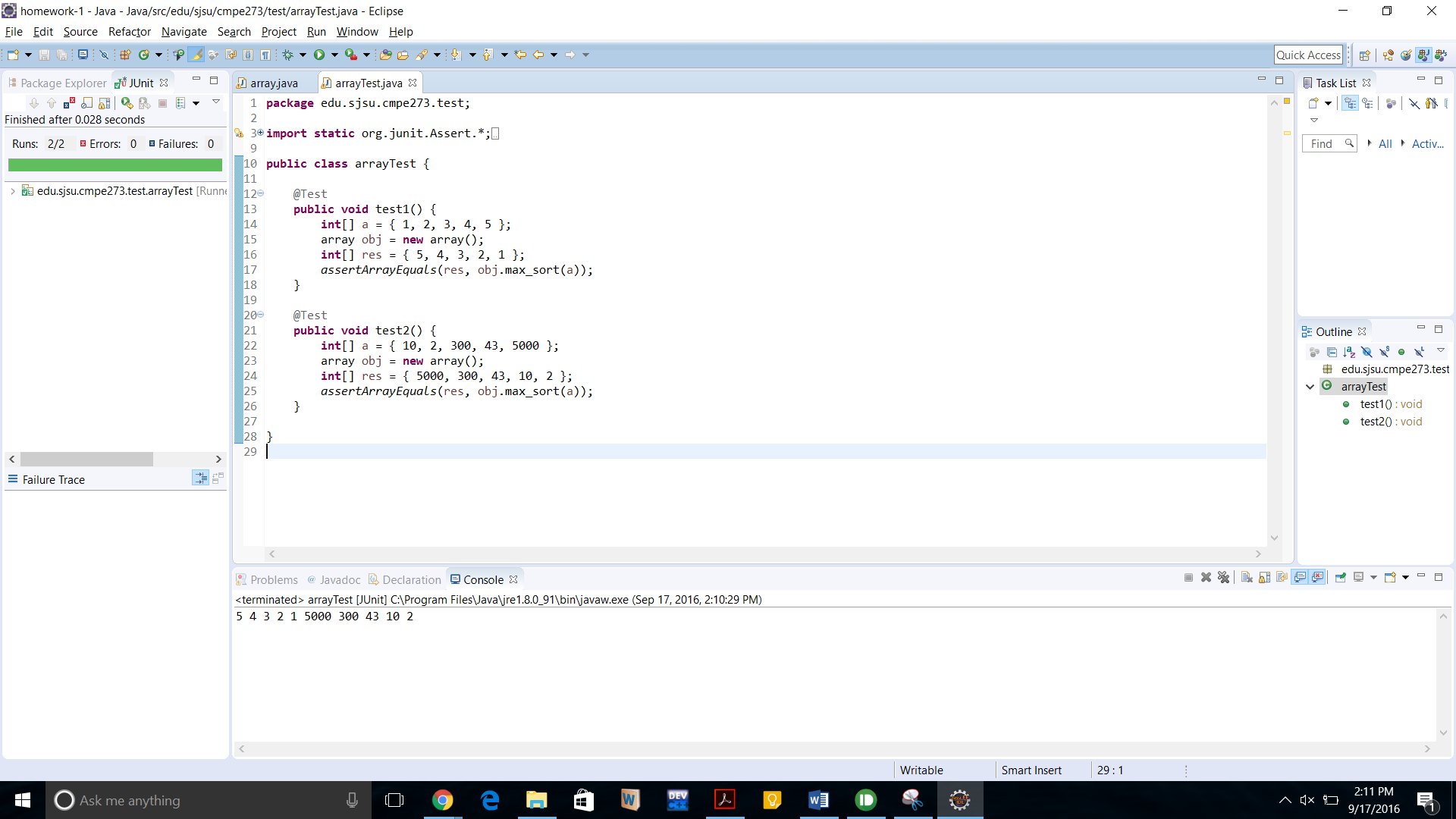
int[] res = { 5000, 300, 43, 10, 2 };

assertArrayEquals(res, obj.max\_sort(a));

}

}

**Output:**



**4)** **Interfaces:**

Interface is a reference type which is similar to class. It has collections of abstract methods which can be inherited by any user-defined class if it implements that particular interface.

Besides the abstract methods, interfaces can also contain constants, static methods and nested types.

An interface is written in .java extension.

We can’t instantiate an interface. It doesn’t contain any constructor, all methods in interface are abstract, interface can’t contain an input field and rather extend, it will be implemented by class.

An interface can also extend multiple interfaces.

**Programming Question**:

Write a program which will make use of interfaces and call the methods written in the interface class to demonstrate how it can be useful in programs.

**Code:**

**package** edu.sjsu.cmpe273;

**import** javax.swing.Spring;

**interface** Enginnering {

**public** String[] diciplines();

**public** **int** subjects();

}

**interface** SJSU {

**public** String[] Building();

}

**public** **class** Interface **implements** Enginnering, SJSU {

**public** String[] diciplines() {

String[] departments = { "Computer Engg", "Software Engg", "Electrical Engg", "Mechanical Engg" };

**return** departments;

}

**public** **int** subjects() {

**int** no\_subjects = 14;

**return** no\_subjects;

}

**public** String[] Building() {

String[] buildings = { "Engg", "Science", "Music" };

**return** buildings;

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Interface i = **new** Interface();

String[] s = i.diciplines();

**for** (String x : s) {

System.***out***.println(x + " ");

}

**int** subj = i.subjects();

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

String[] B = i.Building();

**for** (String x : B) {

System.***out***.println(x + " ");

}

}

}

**Junit Program**:

package edu.sjsu.cmpe273.test;

import static org.junit.Assert.\*;

import org.junit.Test;

import edu.sjsu.cmpe273.Interface;

import edu.sjsu.cmpe273.array;

public class InterfaceTest {

Interface i = new Interface();

@Test

public void test1() {

String[] s = i.diciplines();

String[] res = { "Computer Engg", "Software Engg", "Electrical Engg", "Mechanical Engg" };

assertArrayEquals(res, s);

}

@Test

public void test2() {

int subj = i.subjects();

int res = 14;

assertEquals(res, subj);

}

@Test

public void test3() {

String[] B = i.Building();

String[] res = { "Engg", "Science", "Music" };

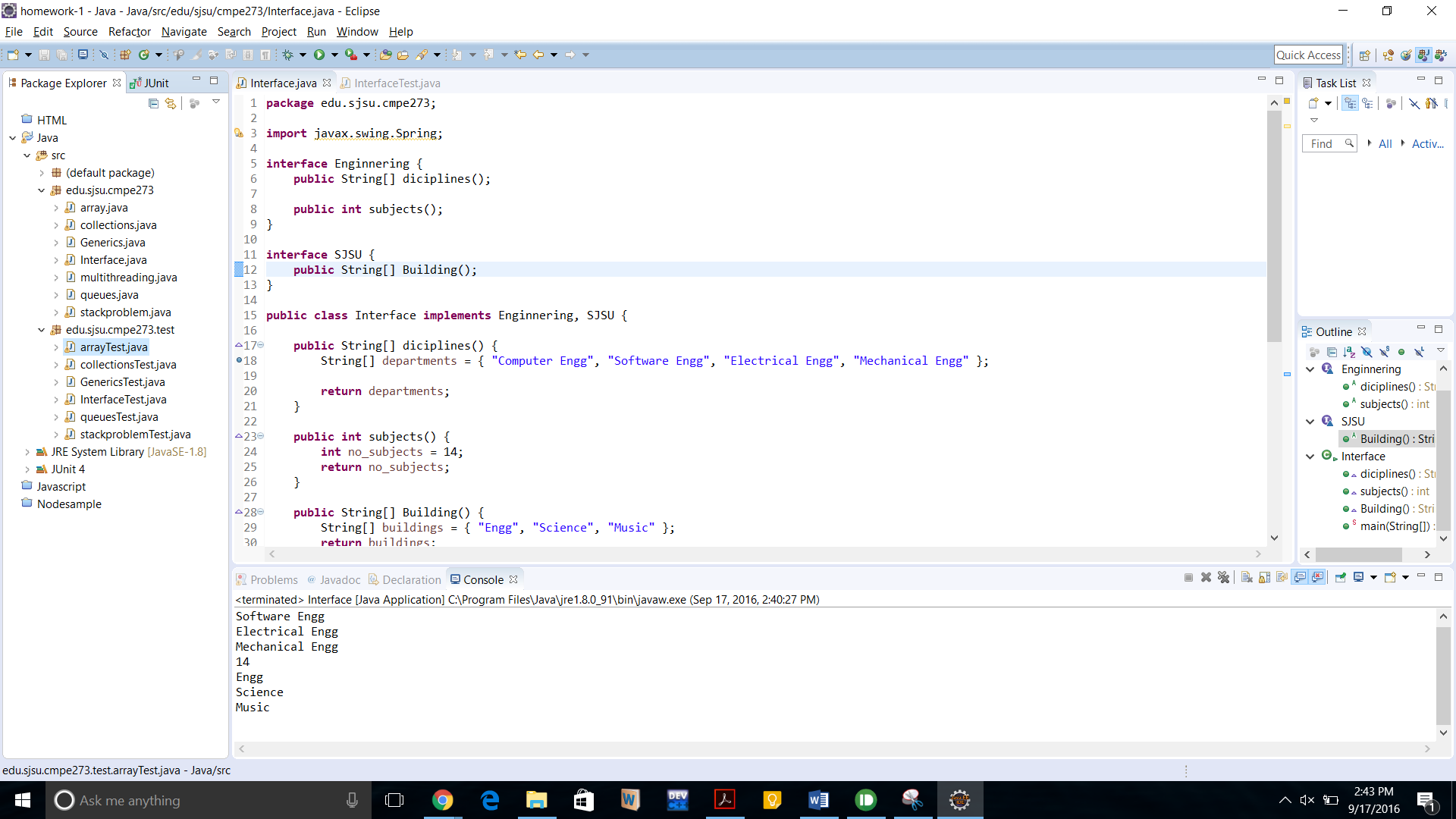
assertArrayEquals(res, B);

}

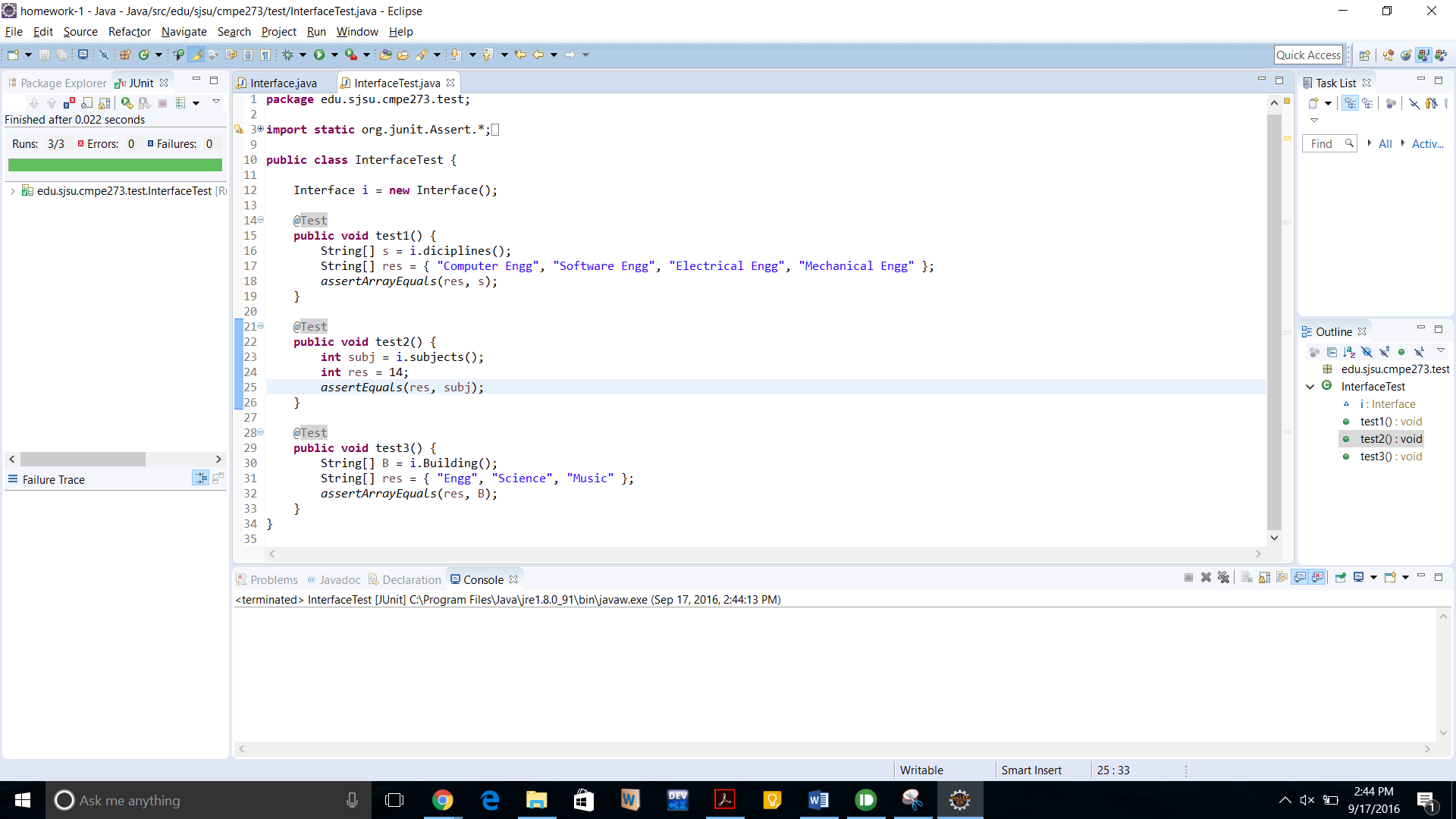
}

**Output:**

Main Program:



Junit output:



**5)** **Collections:**

Collections is a set of standard interfaces such as Linkedlist, Hashset or TreeSet and there are plenty of more!

There are several collection interfaces that can be used by programmers by implementing any of those interfaces to the class:

The List interface: stores the ordered collection of elements.

The Set: this must contain unique values.

The Map: maps unique keys to values and there are some more.

Examples of collection classes which can be extended to the user class:

LinkedList

ArrayList : Dynamic array.

Hashset : for the use of hash tables.

Linkedhashset : to allow insertion order iterations.

TreeSet: implements a set stored in a tree.

**Programming Question:**

Write a program which will make use of any of the Linkedlist collection set and create a function to check if the given entry is stored in linkedlist or not?

**Code:**





**JUNIT program**:



Output:



**6)** **Generics:**

With the use of generics, it allows programmers to be independent of data types of function parameters or classes data types. So, this will result in efficient wring code independent of user input datatype!

The syntax to define generic method is :

Public <G> void myFunction(G[] inputArray){

System.out.print(E Array[0]);

}

Generics classes can be defined as :

Public class myClass<A>{

Public T myvariable;

Public T add(T elem1, T elem2){

………..

};

}

**Programming Question:**

Write a function which is free from any datatypes. Aka the parameters which you are passing can be of any type – string, int, double etc. the function should also return the same contents which are passed as parameters to it.

Code:

//using java generics, make a function general and independent of datatype.

**package** edu.sjsu.cmpe273;

**public** **class** Generics {

**public** **static** <G> G[] Show(G[] myData) {

// display

**for** (G element : myData) {

System.***out***.print(element + " ");

}

**return** myData;

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Integer[] IntArr = { 1, 2, 3, 4, 5 };

String[] StrArr = { "This is string array!" };

Double[] DoubleArr = { 55.0, 4.67, 3.32, 7.54 };

System.***out***.println("printing int array : ");

*Show*(IntArr);

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

System.***out***.println("printing string : ");

*Show*(StrArr);

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

System.***out***.println("printing double array : ");

*Show*(DoubleArr);

}

}

**JUNIT code:**

package edu.sjsu.cmpe273.test;

import static org.junit.Assert.\*;

import org.junit.Test;

public class GenericsTest {

@Test

public void test1() {

Integer[] IntArr = { 1, 2, 3, 4, 5 };

Integer[] Res = { 1, 2, 3, 4, 5 };

assertArrayEquals(Res, IntArr);

}

@Test

public void test2() {

String[] StrArr = { "This is string array!" };

String[] Res = { "This is string array!" };

assertArrayEquals(StrArr, Res);

}

@Test

public void test3() {

Double[] DoubleArr = { 55.0, 4.67, 3.32, 7.54 };

Double[] Res = { 55.0, 4.67, 3.32, 7.54 };

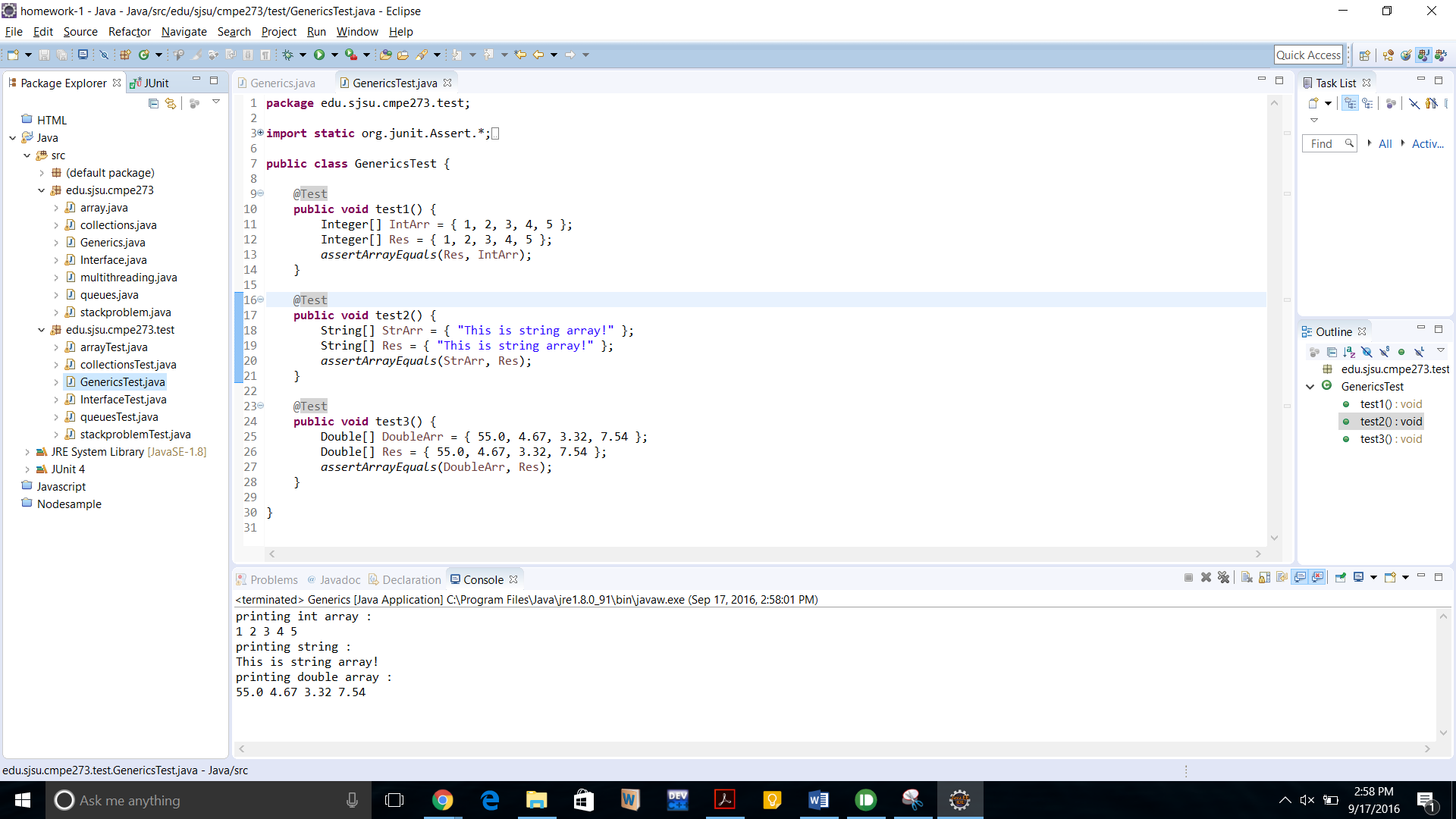
assertArrayEquals(DoubleArr, Res);

}

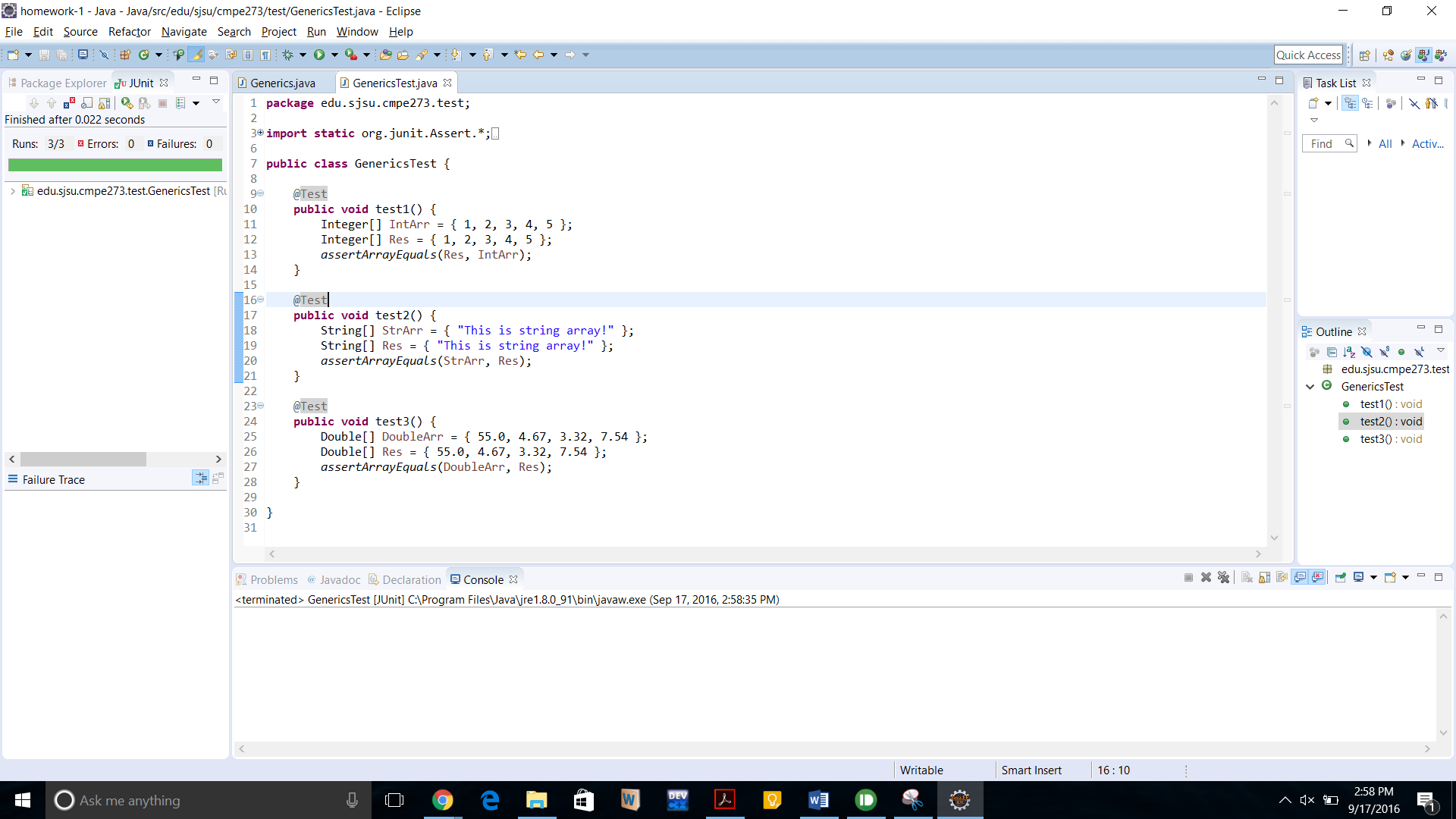
}

Output:

Output of a main program:



Output of JUNIT test case:



7) **Multithreading:**

A multi-threaded program means that two or more thread can run concurrently and each of them can handle different tasks to achieve the result, which is much speedy and efficient.

We can create a multi-threaded program in two ways:

I) by implementing runnable interface

Or

II) By extending the thread class

Both of the above, there are 3 process steps to follow:

1) Instantiate the Thread object

2) User defined functional code to be performed by an individual thread should be written in run() method

3) once a thread object is created, we can start thread by calling start() method, which executes the call to run() method.

Example of Implementing runnable interface:

Class myClass implements Runnable {

Public Thread t = new Thread (this, threadname);

Public void run(){}

Public void start(){}

}

Example of extending Thread class:

Class myClass extends Thread {

Public Thread t = new Thread (this, threadname);

Public void run(){}

Public void start(){}

}

Since, each thread is running concurrently there are chances that the program output will not be come as a desired output so, we can use the thread synchronization method to sync between threads accessing the common block of code.

We can also handle the thread deadlocks and inter-thread communications using wait(),notify() methods.

**Programming Question**:

Write a multithreaded program which will simply call a function. This function is having counter of 1 to 10. Develop the program in such a way that you can trace out the sequence of threads entering in this function. Also, write a program with use of synchronization and w/o sync.

**Code with Sync**:





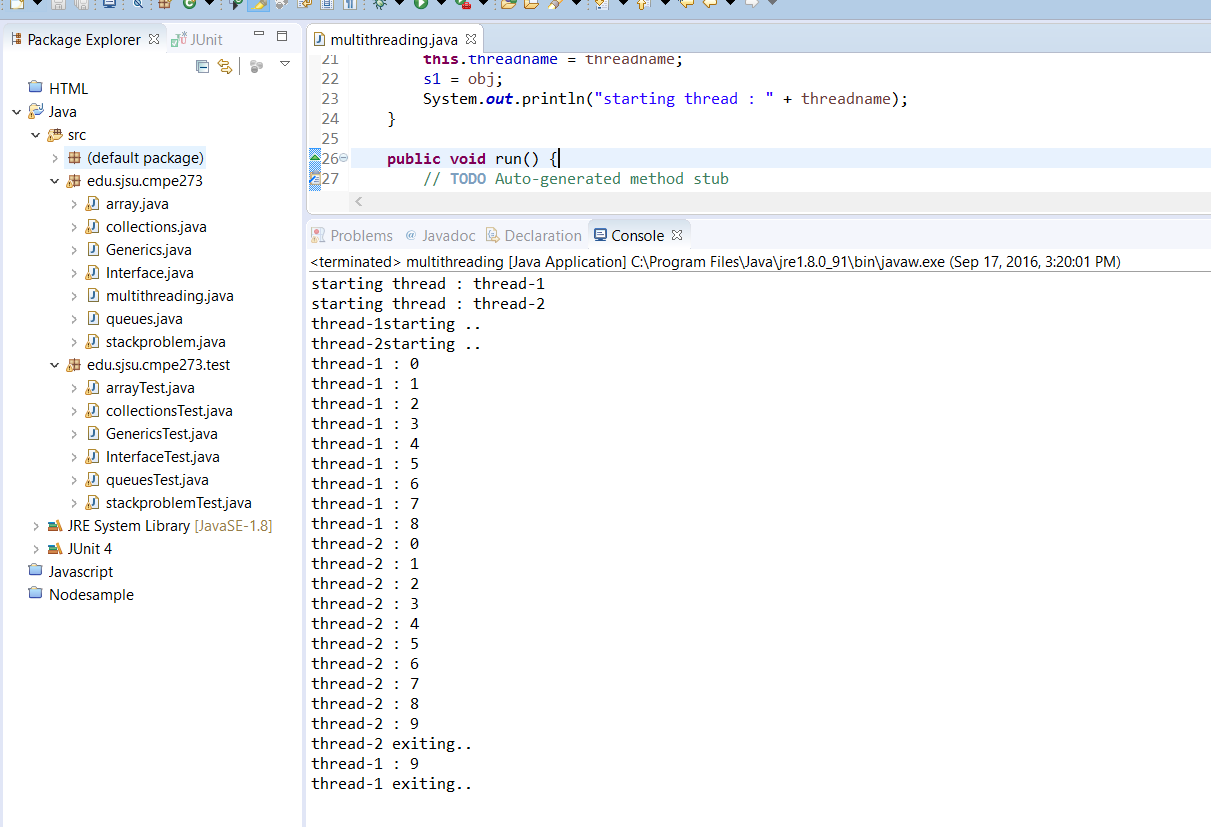
**Without sync:**





Output:

Without Sync:



With sync:

