

Assignment01_Refresher

by Jiongfeng Chen, 02/22/2016

Javascript

Variable

Introduction of topic

Defined as containers for storing data values.

Question

Make a speech or introduction of myself by using variable (output to screen)

Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="01var.js"></script>
<title>Insert title here</title>
</head>
<body>

<h1>JavaScript Variables</h1>

<p id="speechLine"></p>

<script type="text/javascript">
var speech1 = speech();
document.getElementById("speechLine").innerHTML = speech1;
alert(speech());
</script>
</body>
```

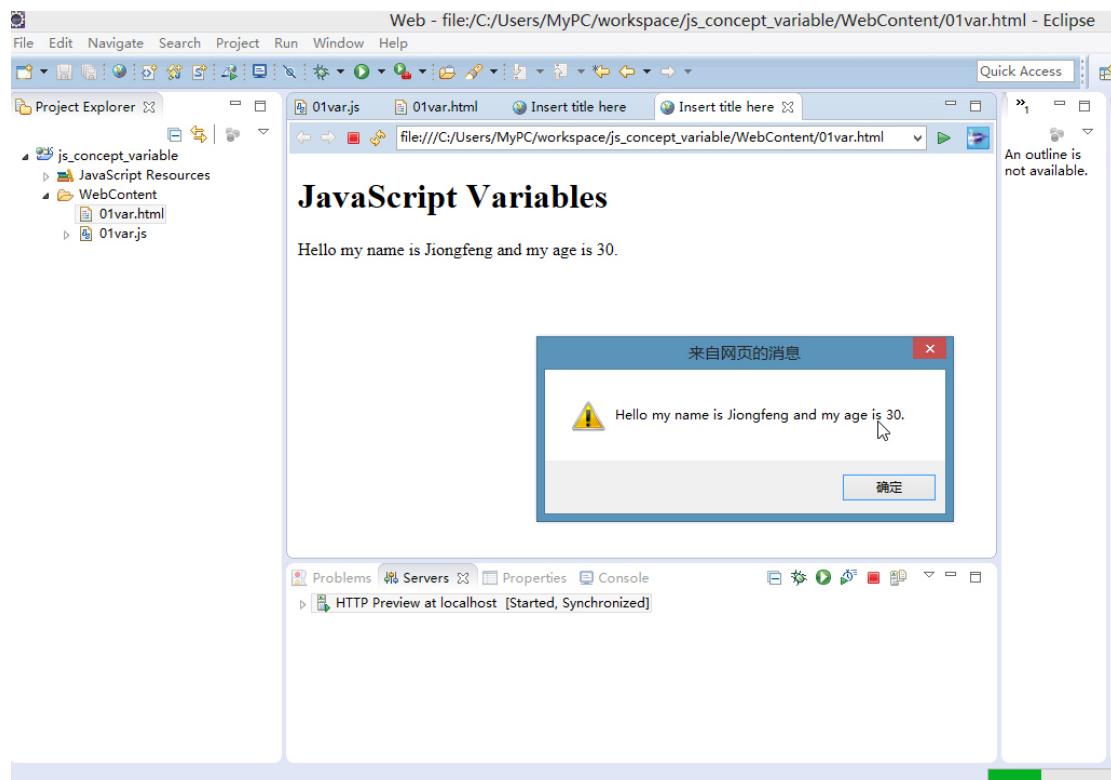
```

</html>

"01var.js"
/**
 * question: give a small speech by using variables
 */
function speech()
{
    var age_a = 15;
    var age_b = 2;
    var age_c = age_a * age_b;
    var intro = "Hello Vedang, my name is";
    var name = "Jiongfeng";
    var intro_ful = intro +" "+ name;
    var speech = intro_ful + " and my age is " + age_c +".";
    return speech;
}

```

Output



Objects

Introduction of topic

Used to describe physical instance in life by giving some properties. For example, a television is an object. It has properties like brand, model, weight and color.

Question

Create an object of my cellphone

Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="02objects.js"></script>
<title>Insert title here</title>
</head>
<body>

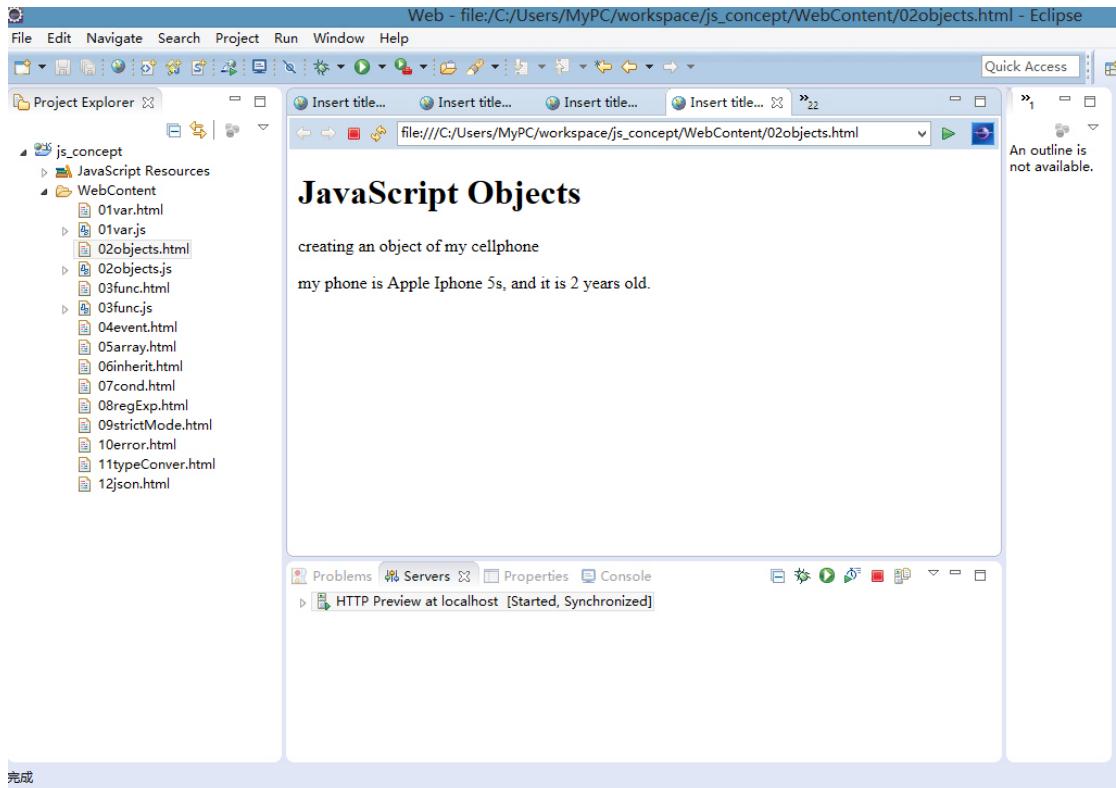
<h1>JavaScript Objects</h1>

<p>creating an object of my cellphone</p>
<p id="phone"></p>

<script type="text/javascript">
var cellp = {
    brandName : "Apple Iphone",
    model   : "5s",
    color   : "black",
    years   : 2,
};
document.getElementById("phone").innerHTML = " my phone is " +
    cellp.brandName + " " + cellp.model + ", and it is " +
    cellp.years + " years old.";
</script>

</body>
</html>
```

Output



Functions

Introduction of topic

Defined as a block of code designed to perform a particular task. It is executed when something called.

Question

Create a function which can multiply two numbers

Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="03func.js"></script>
<title>Insert title here</title>
</head>
<body>
```

<h1>JavaScript Functions</h1>

<p>creating a function which can multiply two numbers</p>

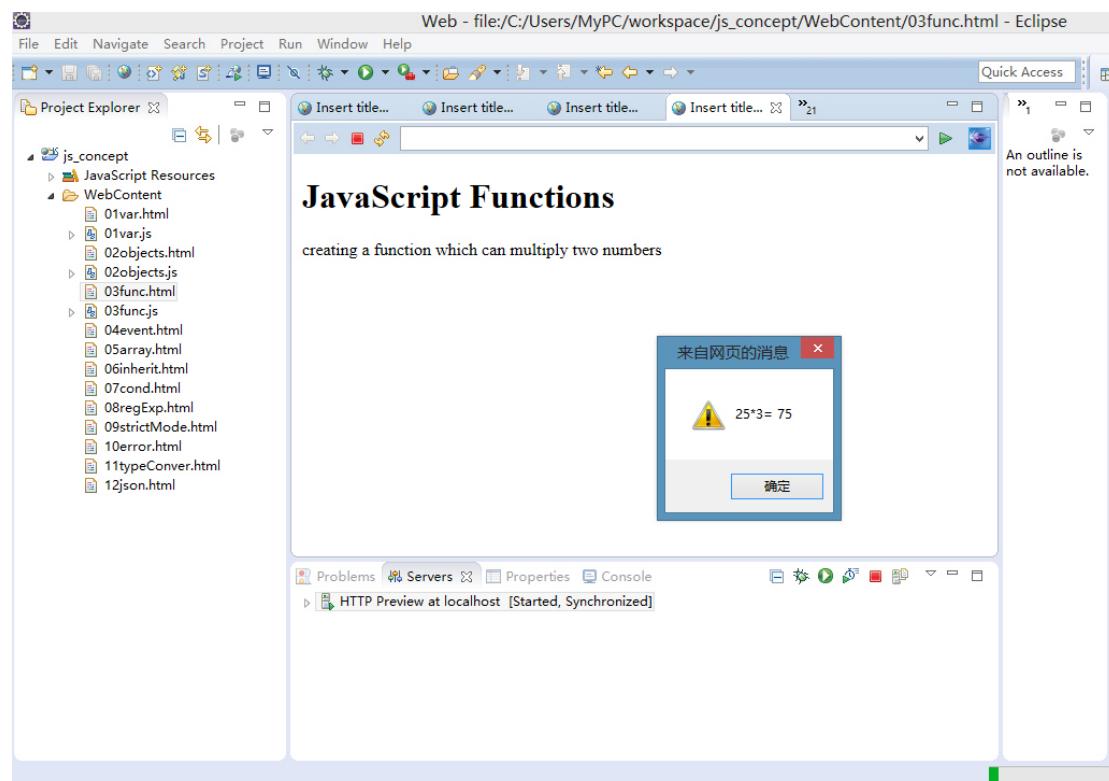
```
<script type="text/javascript">

alert("25*3= " + calc(25, 3));

</script>
</body>
</html>
```

```
//03func.js
function calc(a, b)
{
    return a*b;
}
```

Output



Events

Introduction of topic

Defined to things that happen to HTML elements, while JS can "react" on these events.

Question

Create a button which can tell what you are doing by using event

Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript"></script>
<title>Insert title here</title>
</head>
<body>

<h1>JavaScript Events</h1>

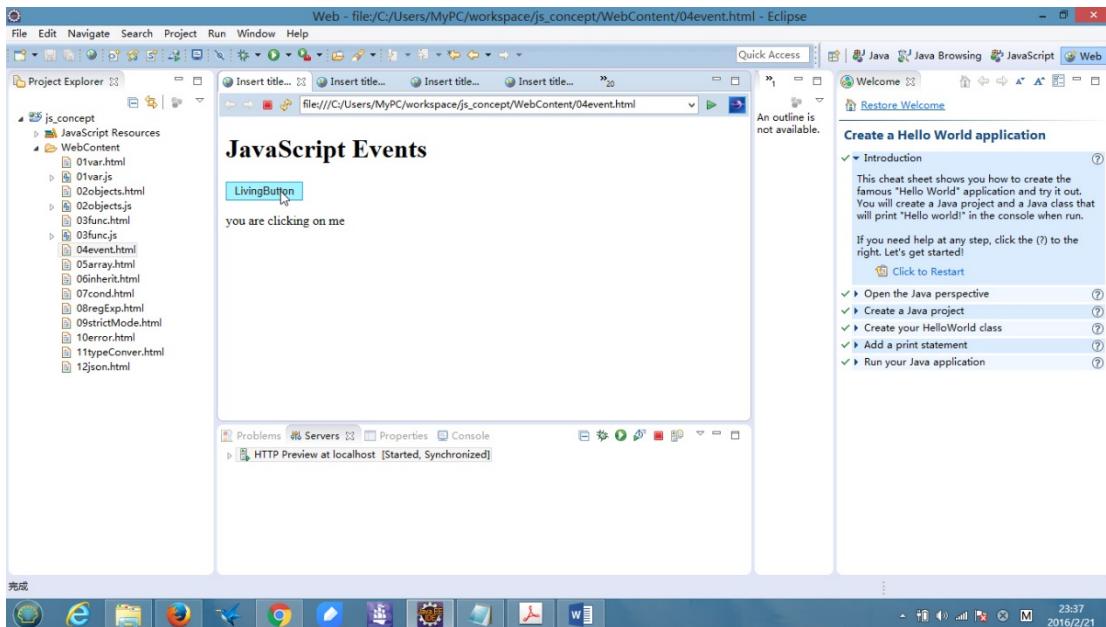
<button onclick="click1()"
onmouseover="mouseover1()">LivingButton</button>

<script type="text/javascript">
function click1(){
    document.getElementById("text").innerHTML = "you are clicking on
me";
}
function mouseover1(){
    document.getElementById("text").innerHTML = "you are moving over
my head";
}
</script>
<p id="text"> </p>

</body>
```

```
</html>
```

Output



Arrays

Introduction of topic

Defined to store multiple values in a single variable.

Question

Add and detail some info of my phone by using array and tell how many things you know about my phone

Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript"></script>
<title>Insert title here</title>
</head>
<body>

<h1>JavaScript Arrays</h1>
```

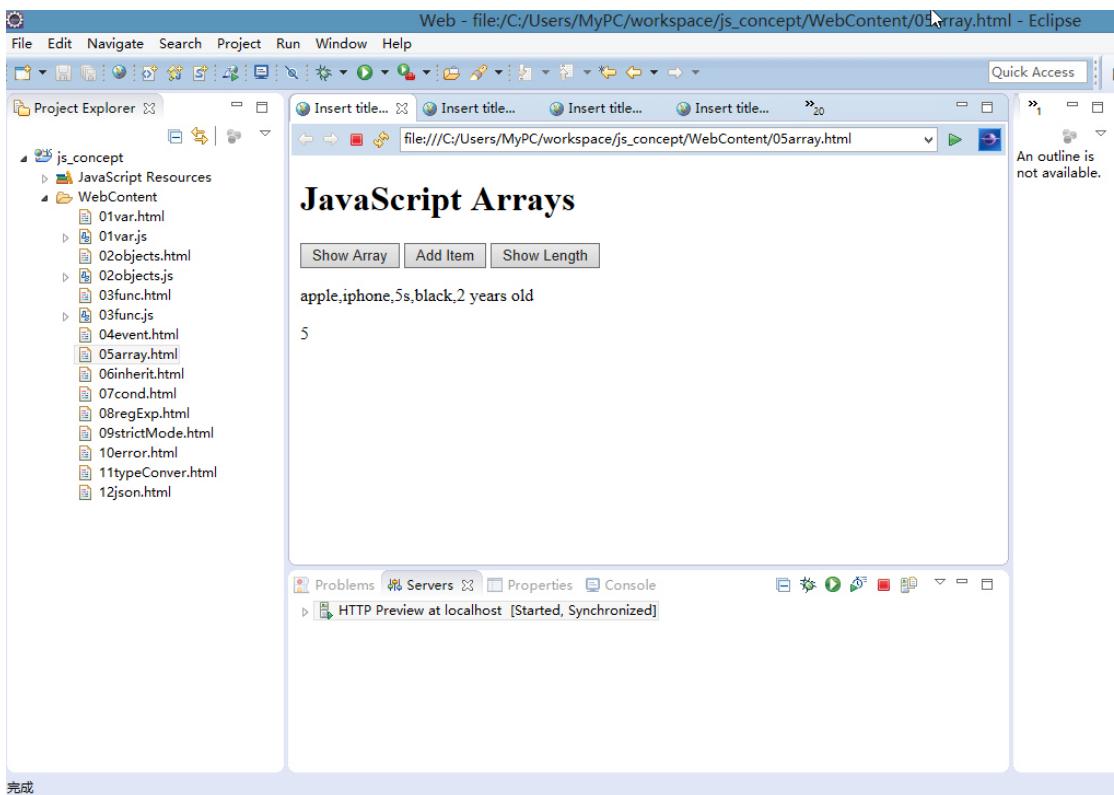
```
<button onclick="click1()">Show Array</button>
<button onclick="click2()">Add Item</button>
<button onclick="click3()">Show Length</button>

<p id="text"> </p>
<p id="textNum"> </p>

<script type="text/javascript">
var phone = ["apple", "iphone", "5s", "black"];
function click1(){
    document.getElementById("text").innerHTML = phone;
}
function click2(){
    phone.push("2 years old");
    document.getElementById("text").innerHTML = phone;
}
function click3(){
    document.getElementById("textNum").innerHTML = phone.length;
}
</script>

</body>
</html>
```

Output



Inheritance

Introduction of topic

In object-oriented programming, defined as inheritance is when an object or class is based on another object or class, using the same implementation specifying implementation to maintain the same behavior.

Question

Create an object of Pad product and create an Ipad product which is the inheritance from Pad

Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
```

```

<script type="text/javascript"></script>
<title>Insert title here</title>
</head>
<body>

<h1>JavaScript Inheritance</h1>

<p id="text1"> </p>
<p id="text2"> </p>

<script type="text/javascript">
//Define the Pad constructor
var Pad = function(Name) {
    this.Name = Name;
};

Pad.prototype.sayHello = function(){
    document.getElementById("text1").innerHTML = "Hello, I'm " +
this.Name;
};

Pad.prototype.sayBye = function(){
    document.getElementById("text2").innerHTML ="BYE";
};

function Ipad(Name, subject) {
    Pad.call(this, Name);
    this.subject = subject;
}

// Create a Ipad object that inherits from Pad prototype.
Ipad.prototype = Object.create(Pad.prototype);
Ipad.prototype.constructor = Ipad;
Ipad.prototype.sayHello = function(){
    document.getElementById("text1").innerHTML ="Hello, I'm " +
this.Name + ". I'm attually "
        + this.subject + ".";
};

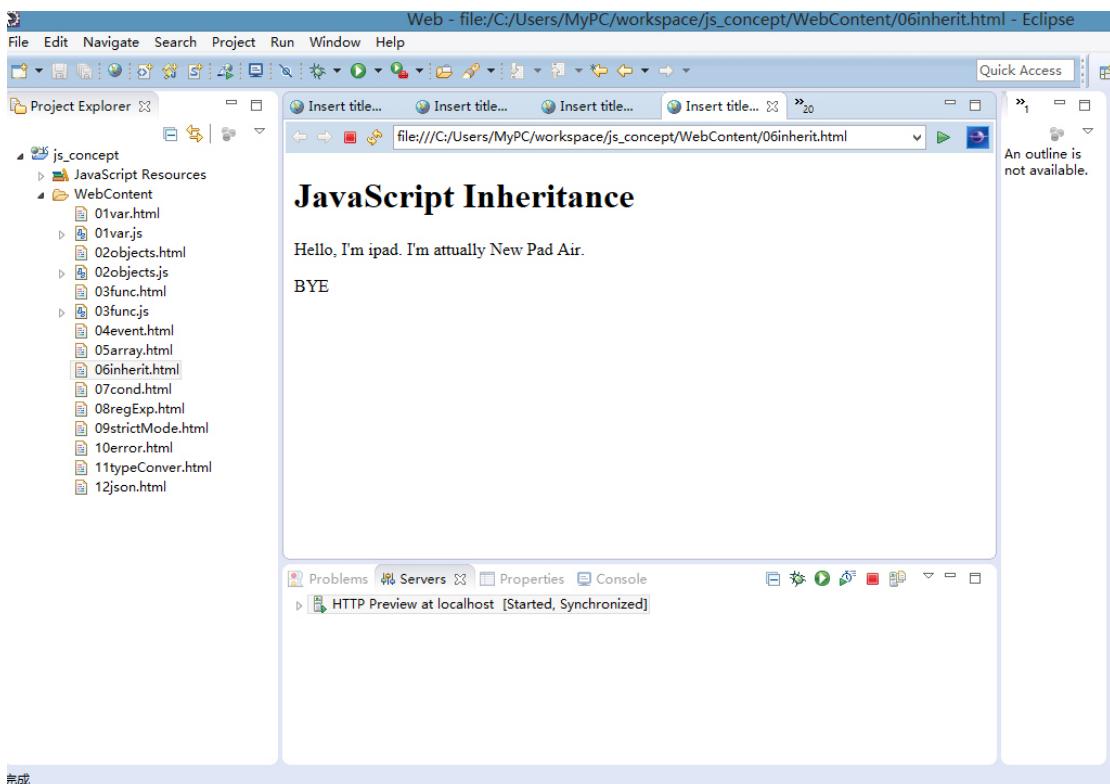
var Ipad1 = new Ipad("ipad", "New Pad Air");
Ipad1.sayHello();
Ipad1.sayBye();

```

```
</script>
```

```
</body>  
</html>
```

Output



Conditions

Introduction of topic

Conditional statements will take different actions according to different situations by using if ... else et al.

Question

Tell me I am old or not, say not younger than 30 is old

Code

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="UTF-8">
```

```

<script type="text/javascript"></script>
<title>Insert title here</title>
</head>
<body>

<h1>JavaScript Conditions</h1>

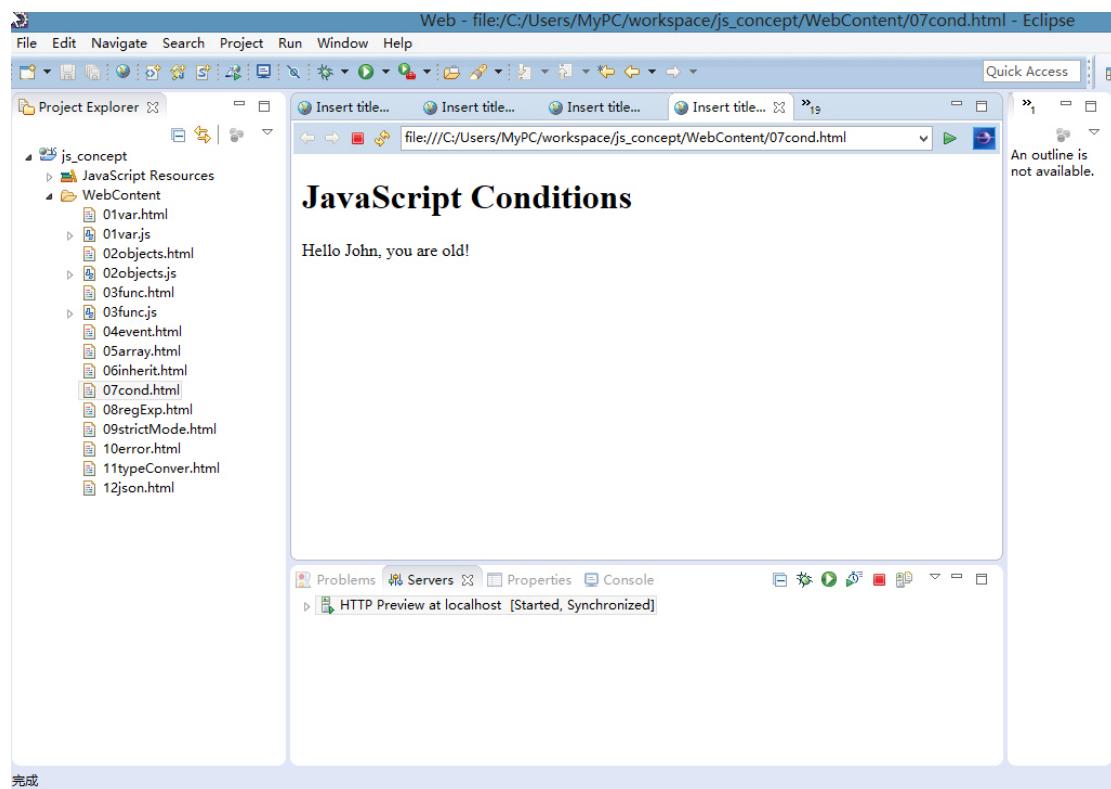
<p id="text1"> </p>
<script type="text/javascript">
var age = 30;

if (age >= 30) {
    document.getElementById("text1").innerHTML = "Hello John, you are
old!";
}
</script>

</body>
</html>

```

Output



Regular Expressions

Introduction of topic

It is a sequence of characters that forms a search pattern, which can be used for text search and text replace operations.

Question

Change the assignment due date in the text from Feb.23 to Feb.22 using RegExp

Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript"></script>
<title>Insert title here</title>
</head>
<body>

<h1>JavaScript Regular Express</h1>

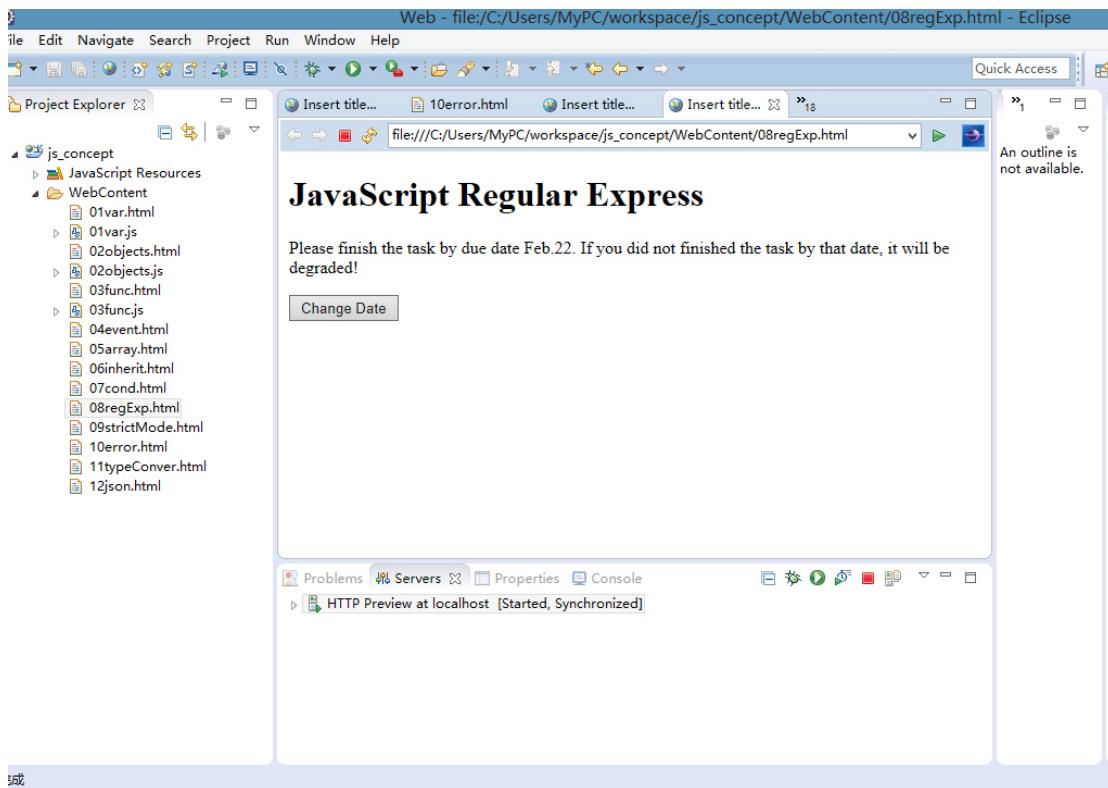
<p id="text1">Please finish the task by due date Feb.23. If you did not finished the task by that date, it will be degraded! </p>

<button onclick="changeDate()">Change Date</button>

<script type="text/javascript">
function changeDate() {
    var str1 = document.getElementById("text1").innerHTML;
    var str2 = str1.replace(/Feb.23/, "Feb.22");
    document.getElementById("text1").innerHTML = str2;
}
</script>

</body>
</html>
```

Output



Strict mode

Introduction of topic

It is new in JavaScript 1.8.5 (ECMAScript version 5). JavaScript code should be executed strictly under this programming mode. With strict mode, we cannot, i.e., use undeclared variables.

Question

Check the difference of variable outputs when using the javascript under strict mode (as under strict mode, “not defined” will show error and blank/no output)

Code

```
<!DOCTYPE html>
<html>
<head>
```

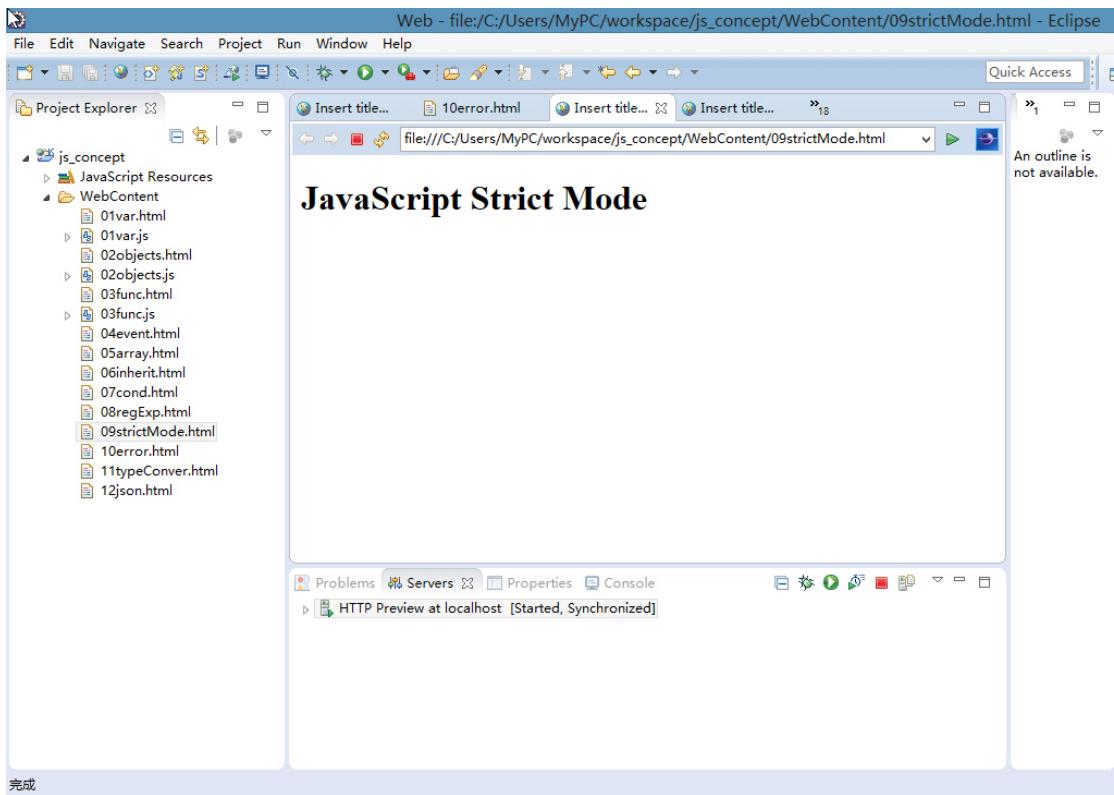
```
<meta charset="UTF-8">
<script type="text/javascript"></script>
<title>Insert title here</title>
</head>
<body>

<h1>JavaScript Strict Mode</h1>

<script type="text/javascript">
"use strict";
var a = 5.3;                      // a is not defined, will cause an error
alert(a);
var arguments = 8.82;      // arguments not allowed, will cause an
error
alert(arguments);
var a = 010;                      // Octal numeric literals are not allowed
alert(a);
</script>

</body>
</html>
```

Output



Errors

Introduction of topic

Errors are equal to the try to catch and throw grammar. The try statement test a block of code for errors. The catch statement handle the error. The throw statement create custom errors.

Question

Input the age to apply for the driver license, too old or too young will throw error

Code

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

```

</head>
<body>

<h1>JavaScript Error</h1>

<input id="age" type="text">
<button type="button" onclick="click1()">input age</button>
<p id="text1"></p>

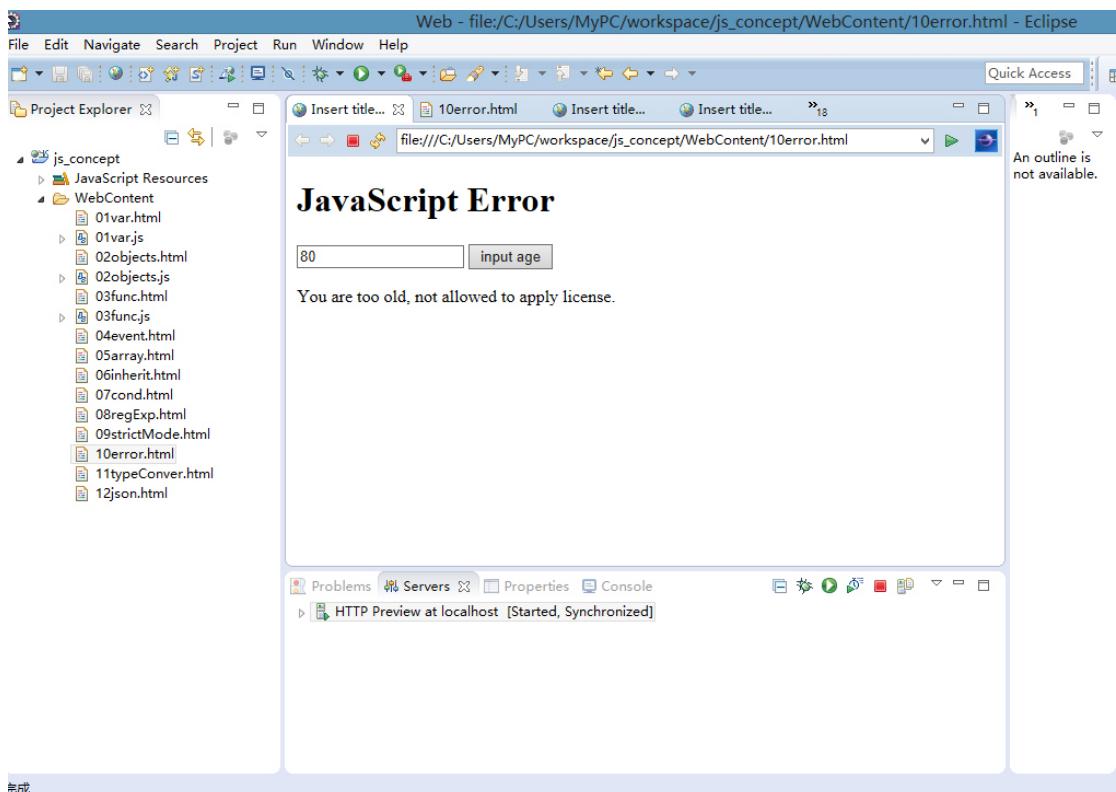
<script type="text/javascript">
function click1() {
    var showText, x;
    showText = document.getElementById("text1");
    showText.innerHTML = "";
    x = document.getElementById("age").value;
    try {
        x = Number(x);
        if(x < 18)    throw "too young, not allowed to apply
license.";
        if(x > 70)   throw "too old, not allowed to apply license.";
    }
    catch(err) {
        showText.innerHTML = "You are " + err;
    }
}

</script>

</body>
</html>

```

Output



Type Conversions

Introduction of topic

Javascript convert types among Number, String, Boolean, et al.

Question

Create some test cases of conversion from string to number, number to string and print it, to see how type conversion work

Code

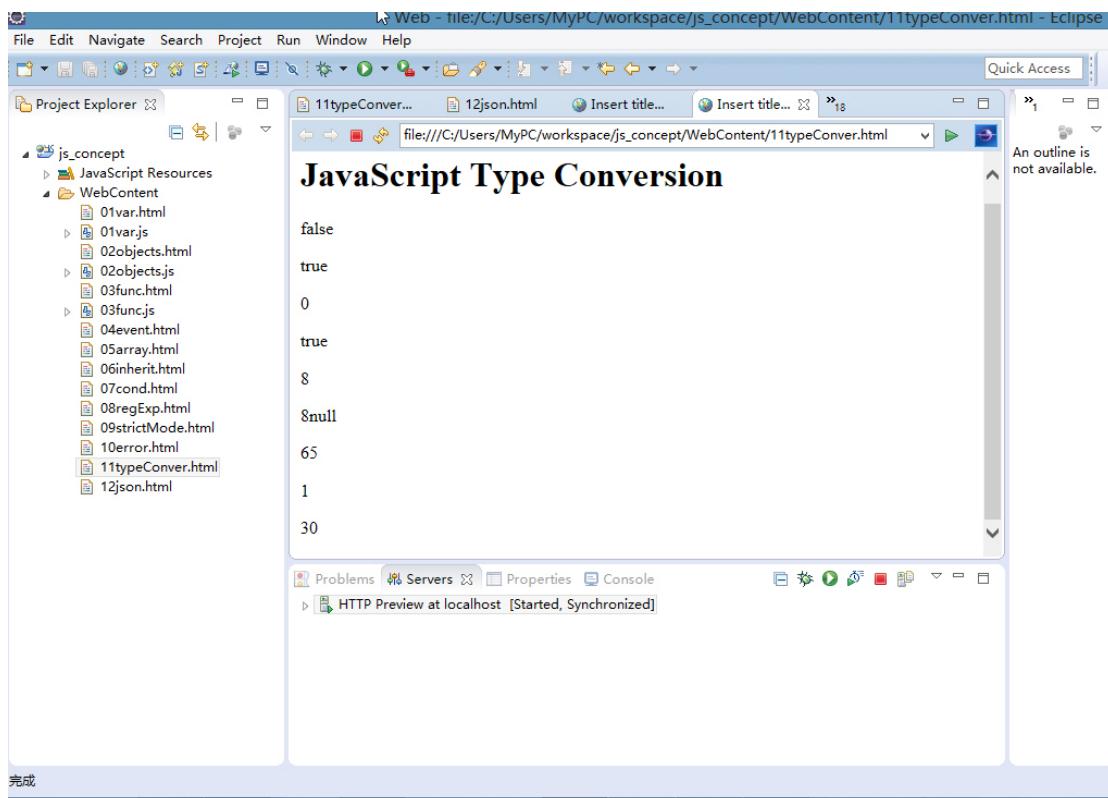
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript"></script>
<title>Insert title here</title>
</head>
<body>
```

```
<h1>JavaScript Type Conversion</h1>
<p id="text1"> </p>
<p id="text2"> </p>
<p id="text3"> </p>
<p id="text4"> </p>
<p id="text5"> </p>
<p id="text6"> </p>
<p id="text7"> </p>
<p id="text8"> </p>
<p id="text9"> </p>

<script type="text/javascript">
document.getElementById("text1").innerHTML=false.toString()
document.getElementById("text2").innerHTML=true.toString()
document.getElementById("text3").innerHTML=Number(false)
document.getElementById("text4").innerHTML=true
document.getElementById("text5").innerHTML=8 + null
document.getElementById("text6").innerHTML="8" + null
document.getElementById("text7").innerHTML="6" + 5
document.getElementById("text8").innerHTML="6" - 5
document.getElementById("text9").innerHTML="6" * "5"
</script>

</body>
</html>
```

Output



JSON

Introduction of topic

JSON stands for JavaScript Object Notation, it is a format for storing and transporting data, and often used when data is sent from a server to a web page.

Question

Create a JASON array of my favorite sports list, and print my most favorite one on the screen

Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript"></script>
```

```

<title>Insert title here</title>
</head>
<body>

<h1>JavaScript JSON</h1>

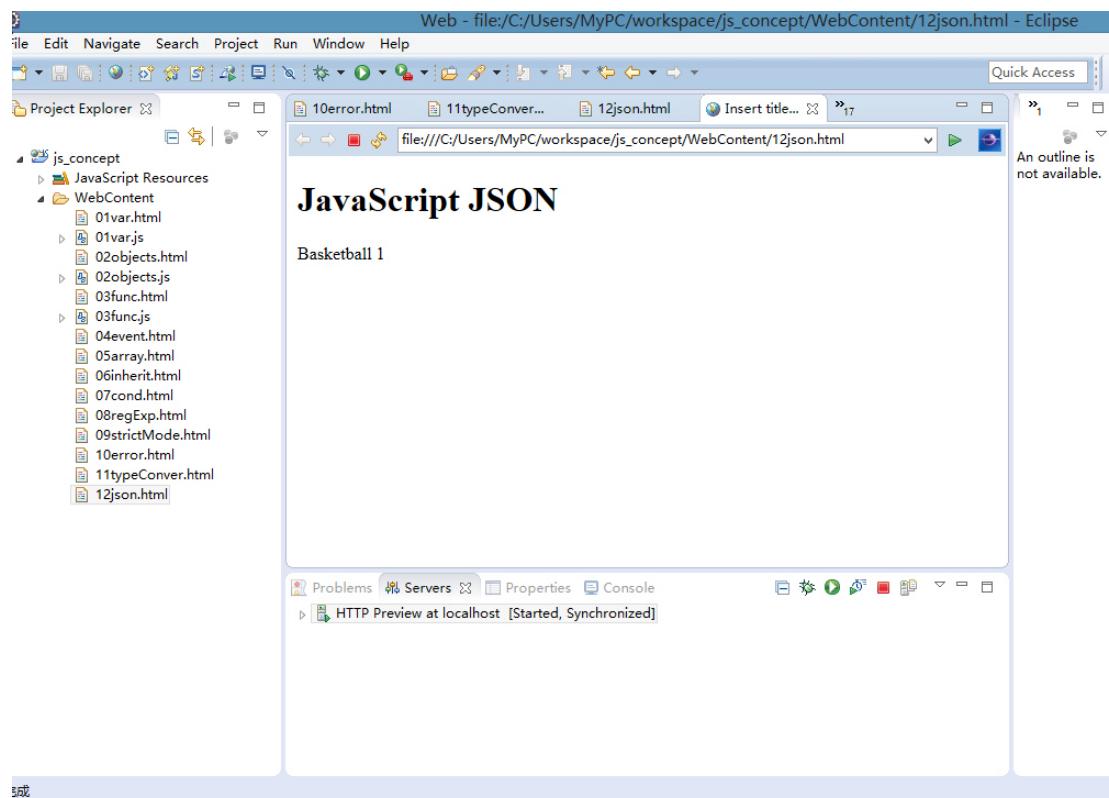
<p id="text1"> </p>

<script type="text/javascript">
var sports = '{"sports":[' +
'{"Name":"Basketball","Rank":"1" },' +
'{"Name":"Football","Rank":"2" },' +
'{"Name":"Swimming","Rank":"3" }]';
obj = JSON.parse(sports);
document.getElementById("text1").innerHTML = obj.sports[0].Name + " " +
" + obj.sports[0].Rank;
</script>

</body>
</html>

```

Output



HTML5

Local Storage

Introduction of topic

It stores data with no expiration date, unlike session storage, which stores data for one session. The data will be available even when the browser or browsing tab is closed or reopened. It is a simple client side database that allows the users to persist data in the form of key pairs or value pairs. It has a fairly simple API to write data into the local storage, it can store up to 10MB of data per domain. Unlike cookies, the data stored are not included with every HTTP request.

Question

Add number and save it to the local storage, and output to the screen. And set the color and save it to local storage, when the webpage is opened again, try to load the color as background color.

Code

//Open two tabs and click the button alternatively to see how the local saved value changes.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="01LocStor.js"></script>
</head>
<body onload="changeColor()">

<h1>HTML5 Local Storage</h1>
```

```

<p><button onclick="counter1()" type="button">Add 10!</button></p>
<p><button onclick="setColor()" type="button">Save color yellow to
LocalStorage for loading!</button></p>

<div id="showNum"></div>
<script type="text/javascript">

</script>

</body>
</html>

//01LocStor.js
/**
 * question: save total number users give by clicking the button, and
reload the number when the page reload
*/
function counter1() {
    if(typeof(Storage) !== "undefined") {
        try {
            if (localStorage.numberTotal) {
                localStorage.numberTotal =
Number(localStorage.numberTotal)+10;
            } else {
                localStorage.numberTotal = 10;
            }
            document.getElementById("showNum").innerHTML = "Saved Total
Number: " + localStorage.numberTotal + ".";
        }
        catch (e)
        {
            if (e == QUOTA_EXCEEDED_ERR) {
                alert('Quota exceeded!');
            }
        }
    } else {
        document.getElementById("showNum").innerHTML = "does not support web
storage";
    }
}
function setColor() {
    if(typeof(Storage) !== "undefined") {

```

```

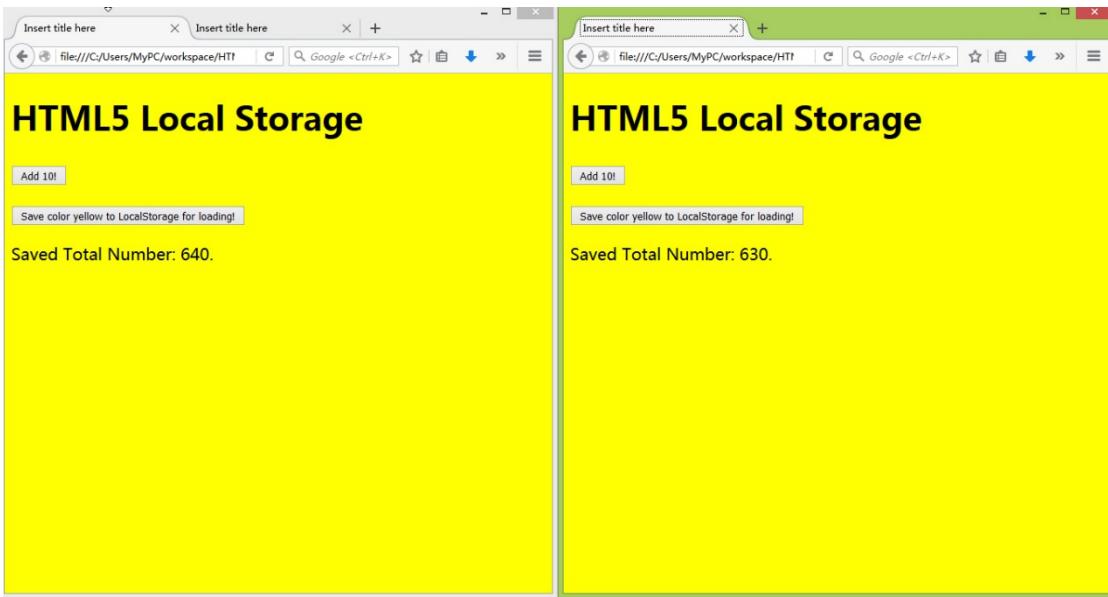
try
{
    localStorage.fontsize = 20;
    localStorage.bgcolor = '#ffff00';

}
catch (e)
{
    if (e == QUOTA_EXCEEDED_ERR) {
        alert('Quota exceeded!');
    }
}
else {
    document.getElementById("showNum").innerHTML = "does not support web
storage";
}
}

function changeColor()
{
    if (localStorage.length != 0) {
        document.body.style.backgroundColor = localStorage.bgcolor;
        document.body.style.fontSize = localStorage.fontsize + 'px';
        document.getElementById("showNum").innerHTML = "!=0" +
localStorage.bgcolor;
    } else {
        document.body.style.backgroundColor = '#000000';
        document.body.style.fontSize = '50px'
        document.getElementById("showNum").innerHTML = "0" +
localStorage.bgcolor;
    }
}

```

Output



Media (Video and Audio)

Introduction of topic

It is used to play video and audio on the website

Question

Download a sample video file and audio file and put them on the web page.

Code

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

<h1>HTML5 Media(Video and Audio)</h1>

<p>Audio: audio1.mp3 </p>
<audio controls>
  <source src="audio1.mp3" type="audio/mpeg">
</audio>
```

```

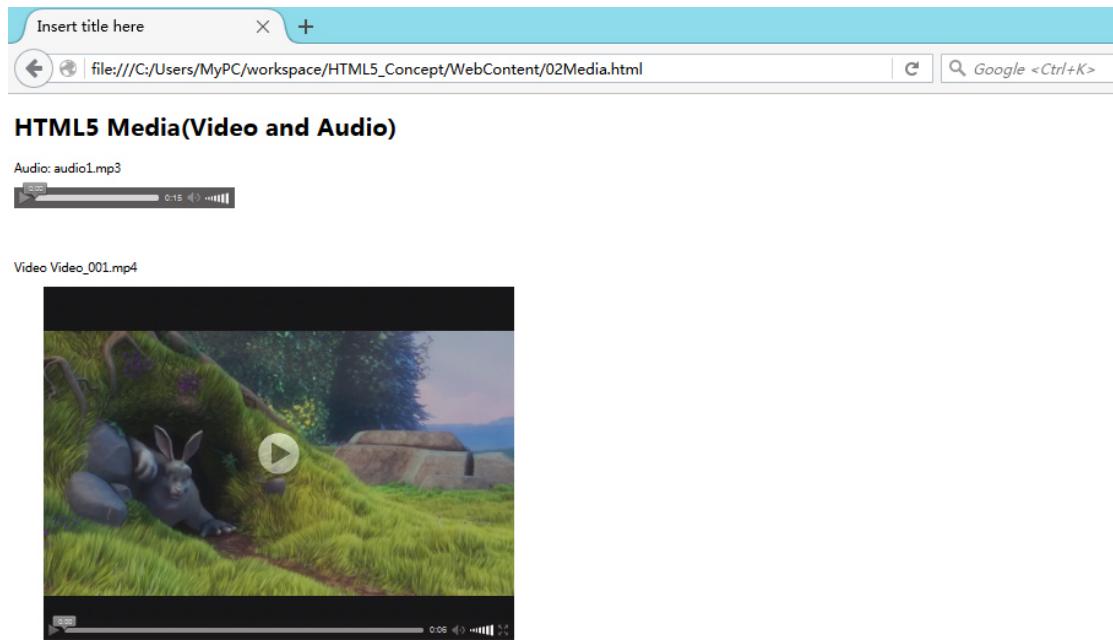
<br> <br><br>
<p>Video Video_001.mp4</p>
<video width="720" height="480" controls>
  <source src="Video_001.mp4" type="video/mp4">
</video>

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

</body>
</html>

```

Output (the video file is downloaded from a free website of internet)



Input Type (make use of different input property options in HTML5 like patterns, autofocus, required, email etc. Place types you want, mention the properties used in your Introduction to Topic section)

Introduction of topic

Describe the input types of the <input> element, and it has different input property options. In this part, the properties, disabled, readonly, autofocus, required , maxlength, pattern, autocomplete, email are used in the code to help to design a web page for input the information. For example, the university name is set to SJSU, as the students are all coming from the same university. As the output screenshot shows, if we did not input last name for the input type with attribute “required”, it shows red color on it. Similarly if we did not match the pattern, it also shows red color.

Question

Make a web page to register your information on internet for class CMPE273

Code

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

<h1>HTML5 Input Types</h1>

<form>
University name:<br>
<input type="text" name="university" value="SJSU" disabled>
<br>Class name:<br>
<input type="text" name="classname" value="CMPE273" readonly>
<br>First name:<br>
<input type="text" name="fname" autofocus>
<br>Last name:<br>
```

```

<input type="text" name="Lname" required maxlength="20">
<br>Home Country<br>
<input type="text" name="countrycode" pattern="[A-Za-z]{3}" title="3
letter">
<br>Email<br>
<input type="email" name="email" autocomplete="off">
</form>

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

</body>
</html>

```

Output



HTML5 Input Types

University name:

Class name:

First name:

Last name:

Home Country

Email

Geolocation

Introduction of topic

The HTML Geolocation API is used to locate a user's position.

Question

Design a web page to find where I am and show the map

Code

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

<h1>HTML5 Geolocation</h1>

<p id="showStatus"></p>
<div id="mapDiv"></div>
<button onclick="getLoc()">Get Location: Find where I am</button>

<script>
var a = document.getElementById("showStatus");

function getLoc() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(getPos, showError);
    } else {
        a.innerHTML = "Geolocation is not supported.";
    }
}

function getPos(position) {
    a.innerHTML = "Latitude: " + position.coords.latitude + " Longitude: "
+ position.coords.longitude;
    var pos_xy = position.coords.latitude + "," +
position.coords.longitude;
    var img_url =
"http://maps.googleapis.com/maps/api/staticmap?center="+pos_xy+"&zoom=14&size=400x300&sensor=false";
    document.getElementById("mapDiv").innerHTML = "<img
src='"+img_url+"'">";
}

function showError(error) {
    switch(error.code) {
```

```

        case error.UNKNOWN_ERROR:
            a.innerHTML = "Unknown error occurred."
            break;
        case error.TIMEOUT:
            a.innerHTML = "Get user location timed out."
            break;
        case error.POSITION_UNAVAILABLE:
            a.innerHTML = "Location information is unavailable."
            break;
        case error.PERMISSION_DENIED:
            a.innerHTML = "Geolocation request denied by user."
            break;
    }
}
</script>

</body>
</html>

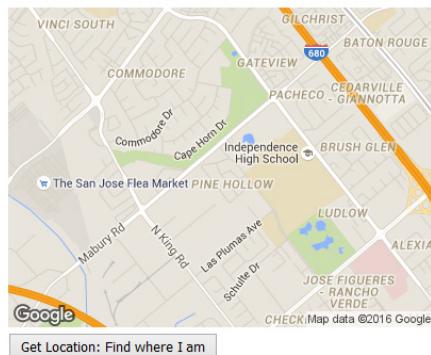
```

Output



HTML5 Geolocation

Latitude: 37.369560 Longitude: -121.8630065



JAVA

Queues

Introduction of topic

It is a kind of data structure. In Java, it is an interface, which is a subtype of the Collection interface.

Question

Create and process a sample queue containing string of "CMPE class 273"

Code

```
import java.util.*;

public class QueueInOut {

    public static void main(String[] args) {
        //TODO Auto-generated method stub
        Queue<String> myQueue = new LinkedList<String>();

        //int time = Integer.parseInt(args[0]);
        //Queue <Integer> queue = new LinkedList<Integer>();
        for (int i = 1; i <= 3; i++)
        {
            myQueue.add("Team member " + i );
            System.out.println(myQueue.toArray()[myQueue.size()-1] +
                " of CMPE273 added. " + "Number of elements: " +
                myQueue.size());
            System.out.println(Arrays.toString(myQueue.toArray()));
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

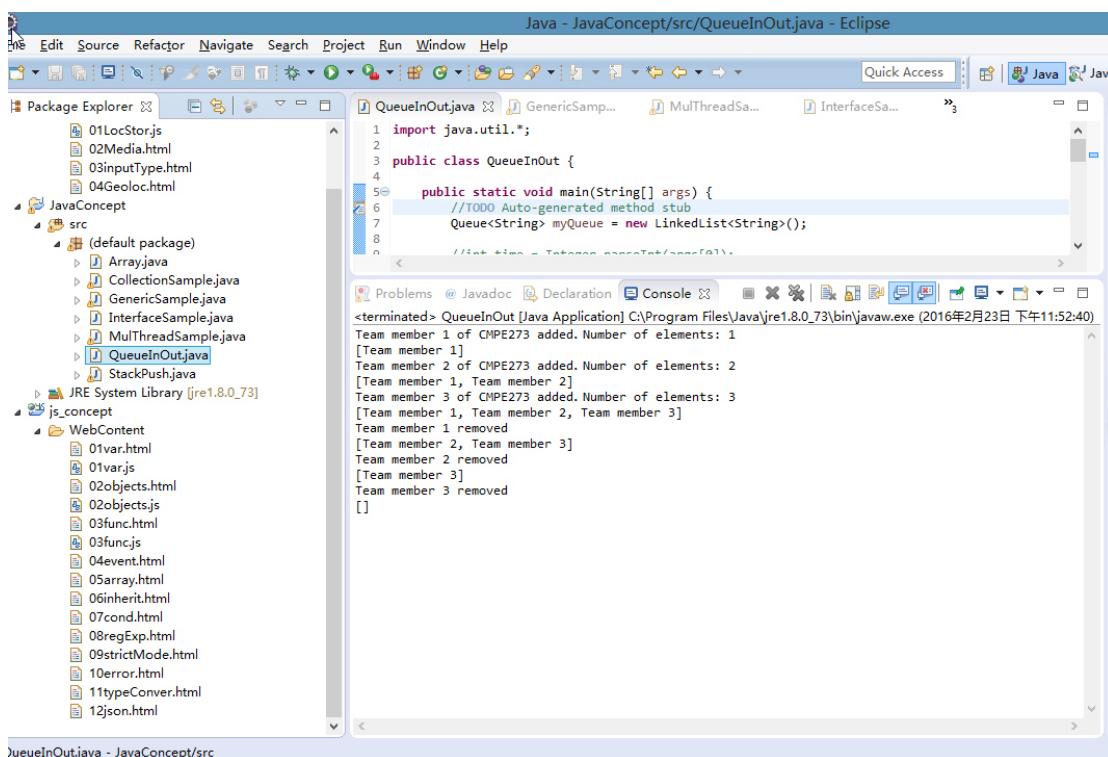
```

        while (!myQueue.isEmpty()) {
            System.out.println(myQueue.remove()+" removed");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println(Arrays.toString(myQueue.toArray()));
        }
    }

}

```

Output



Test using Junit Framework

```

import static org.junit.Assert.assertEquals;
import org.junit.Test;
public class QueueInOutTest {
    @Test
    public void evaluatesExpression() {
        QueueInOut queueInOut = new QueueInOut();
        assertEquals("[This is, Class]", queueInOut.outScreen ());
    }
}

```

```

    }

}

public class QueueInOut {
    public static String outScreen() {
        //TODO Auto-generated method stub
        Queue<String> myQueue = new LinkedList<String>();

        //int time = Integer.parseInt(args[0]);
        //Queue <Integer> queue = new LinkedList<Integer>();
        for (int i = 1; i <= 3; i++)
        {
            myQueue.add("Team member " + i );
            System.out.println(myQueue.toArray()[myQueue.size()-1] +
                " of CMPE273 added. " + "Number of elements: " +
                myQueue.size());
            System.out.println(Arrays.toString(myQueue.toArray()));
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        while (!myQueue.isEmpty()) {
            System.out.println(myQueue.remove()+" removed");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println(Arrays.toString(myQueue.toArray()));
        }
    }
    return Arrays.toString(myQueue.toArray());
}
}

```

Stacks

Introduction of topic

A Stack is a data structure where you add elements to the "top" of the

stack, and also remove elements from the top again. In Java, it is a class found as `java.util.Stack` class.

Question

Create and process a sample stack containing string of "CMPE class 273"

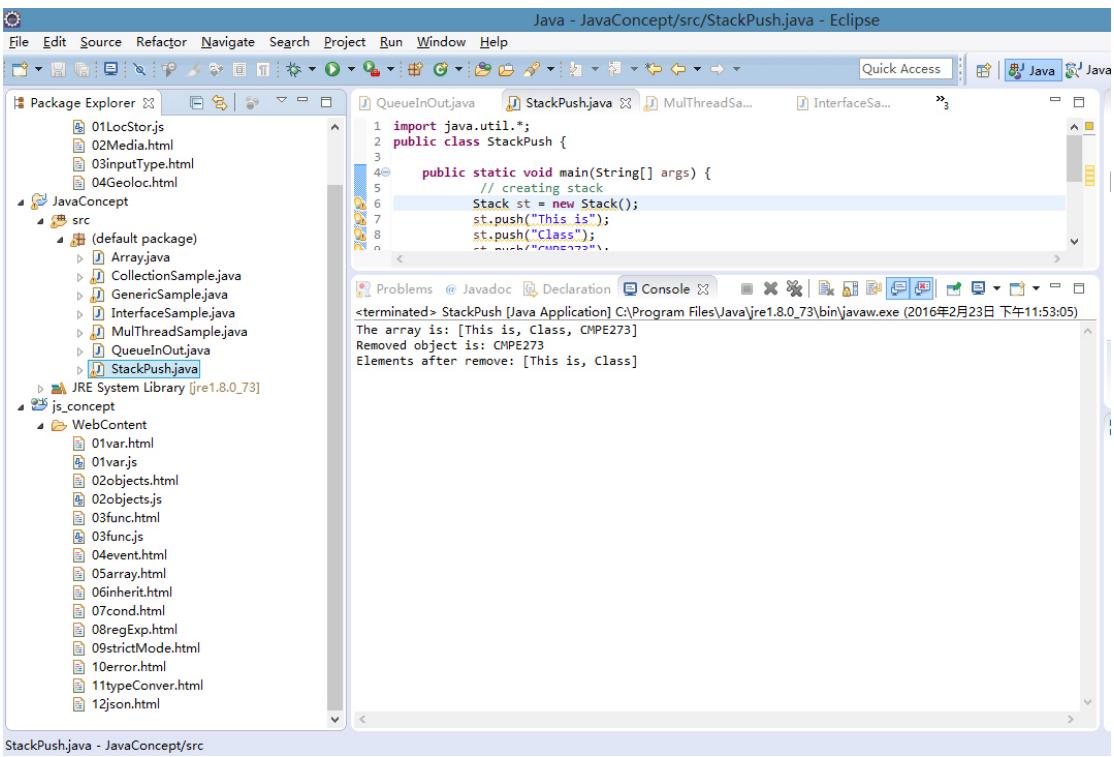
Code

```
import java.util.*;
public class StackPush {

    public static void main(String[] args) {
        // creating stack
        Stack st = new Stack();
        st.push("This is");
        st.push("Class");
        st.push("CMPE273");

        System.out.println("The array is: "+st);
        // removing one object
        System.out.println("Removed object is: "+st.pop());
        System.out.println("Elements after remove: "+st);
    }
}
```

Output



Arrays

Introduction of topic

It is a kind of data structure, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data.

Question

Create, initialize and process a sample arrays containing some digit numbers.

Code

```
public class Array {

    public static void main(String[] args) {
        //TODO Auto-generated method stub
        double[] List1 = {3.2, 1.8, 6.4, 10.9};
        for (int i = 0; i < List1.length; i++) {
            System.out.println(List1[i] + " ");
        }
    }
}
```

```

    }
    double total = 0;
    for (int i = 0; i < List1.length; i++) {
        total += List1[i];
    }
    System.out.println("Total is " + total);
}

}

```

Output

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Java - JavaConcept/src/Array.java - Eclipse
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure with packages like 01LocStor.js, 02Media.html, 03inputType.html, 04Geoloc.html, JavaConcept, and src containing files like CollectionSample.java, GenericSample.java, InterfaceSample.java, MultiThreadSample.java, QueueInOut.java, StackPush.java, and Array.java.
- Central Editor:** Displays the Java code for `Array.java`. The code initializes a double array `List1` with values {3.2, 1.8, 6.4, 10.9}, iterates through it to calculate the total, and prints the result.
- Console Tab:** Shows the output of the program execution.
- Console Output:**

```

3.2
1.8
6.4
10.9
Total is 22.3

```

Interfaces

Introduction of topic

An interface is a reference type in Java, and it is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

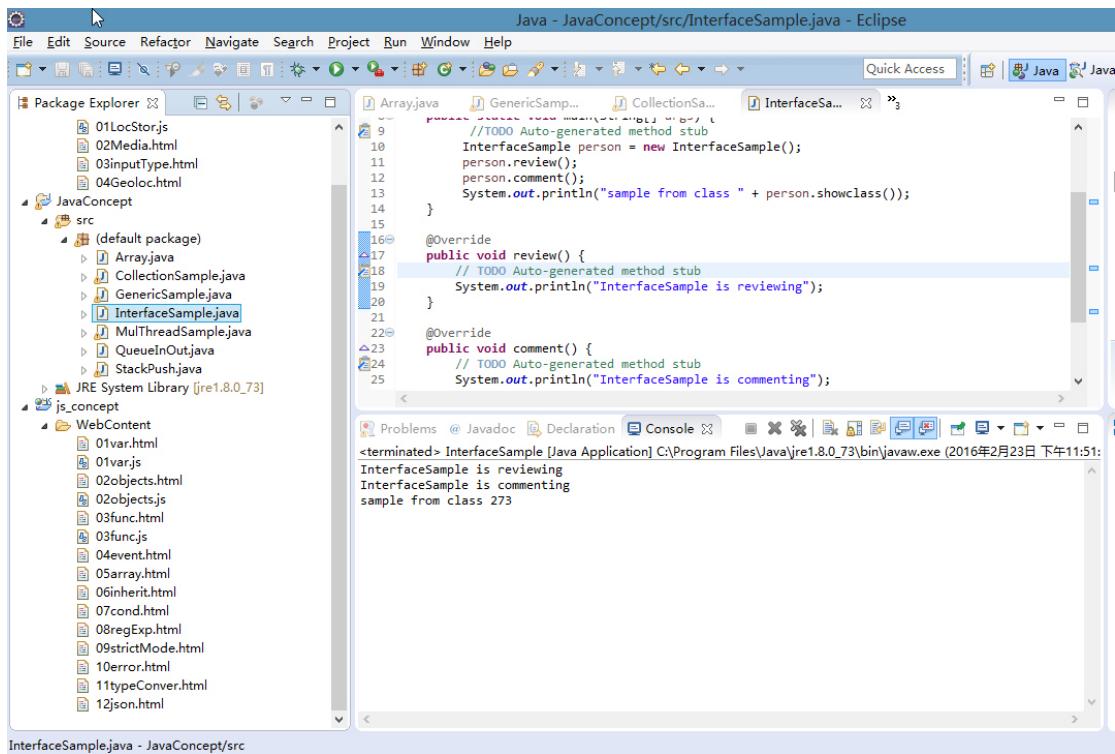
Question

Give a sample class which implements defined interface reviewer.

Code

```
interface Reviewer {  
    public void review();  
    public void comment();  
}  
  
public class InterfaceSample implements Reviewer {  
  
    public static void main(String[] args) {  
        //TODO Auto-generated method stub  
        InterfaceSample person = new InterfaceSample();  
        person.review();  
        person.comment();  
        System.out.println("sample from class " + person.showclass());  
    }  
  
    @Override  
    public void review() {  
        // TODO Auto-generated method stub  
        System.out.println("InterfaceSample is reviewing");  
    }  
  
    @Override  
    public void comment() {  
        // TODO Auto-generated method stub  
        System.out.println("InterfaceSample is commenting");  
    }  
  
    public int showclass(){  
        return 273;  
    }  
}
```

Output



Collections

Introduction of topic

A collection, sometimes called a container, is simply an object that groups multiple elements into a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data. Typically, they represent data items that form a natural group, such as a poker hand, a mail folder, or a telephone directory.

Question

Give various class examples to show implementations of a collection interface method, for example, add().

Code

```
import java.util.*;

public class CollectionSample {
```

```

public static void main(String[] args) {
    //TODO Auto-generated method stub

    //LinkedList
    List l1 = new LinkedList();
    l1.add("class");
    l1.add("273");
    System.out.println(" LinkedList Elements: ");
    System.out.print(l1);

    //HashMap
    Map m1 = new HashMap();
    m1.put("university", "SJSU");
    m1.put("class", "273");
    System.out.println();
    System.out.println(" Map Elements: ");
    System.out.print(m1);

}

}

```

Output

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Java - JavaConcept/src/CollectionSample.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure with packages like JavaConcept and js_concept, and files like QueueInOut.java, StackPush.java, Array.java, CollectionSample.java, GenericSample.java, InterfaceSample.java, MultThreadSample.java, QueueInOut.java, and StackPush.java.
- Code Editor:** Displays the CollectionSample.java code. The line `System.out.print(l1);` is highlighted in blue.
- Console:** Shows the terminal output of the application's execution.
- Output:**

```

LinkedList Elements:
[class, 273]
Map Elements:
{university=SJSU, class=273}

```

Generics

Introduction of topic

Java Generic methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods or, with a single class declaration, a set of related types, respectively. Using Java Generic concept, we might write a generic method for sorting an array of objects.

Question

Give a Java Generics method print which can handle different input types.

Code

```
import java.util.*;  
  
public class GenericSample {  
  
    public static < E > void printArray( E[] inputArray ) {  
        // Display array elements  
        for ( E element : inputArray ) {  
            System.out.printf( "%s ", element );  
        }  
        System.out.println();  
    }  
  
    public static void main(String[] args) {  
        // Create arrays of Integer, Double and Character  
        Integer[] intArray = { 5, 3, 8, 1, 9 };  
        Double[] doubleArray = { 1.1, 4.2, 3.1, 8.4 };  
        Character[] charArray = { 'C', 'M', 'P', 'E', '2', '7',  
        '3' };  
  
        System.out.println( "\nArray characterArray contains:" );  
        printArray( charArray ); // pass a Character array  
        System.out.println( "Array integerArray contains:" );  
        printArray( intArray ); // pass an Integer array  
        System.out.println( "Array doubleArray contains:" );
```

```

        printArray( doubleArray ); // pass a Double array
    }
}

```

Output

The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The left sidebar has a Package Explorer tab showing a project structure with JavaConcept, src, and js.concept folders containing various files like 01LocStor.js, 02Media.html, etc. The main editor window displays the GenericSample.java code. The code defines a static void main method that creates three arrays: characterArray (Character[]), integerArray (Integer[]), and doubleArray (Double[]). It then prints each array's elements. The bottom right corner shows the Console view with the output of the program:

```

Array characterArray contains:
C M P E 2 7 3
Array integerArray contains:
5 3 8 1 9
Array doubleArray contains:
1.1 4.2 3.1 8.4

```

Multithreading

Introduction of topic

A multi threaded program contains two or more small programs that can run concurrently and each one can handle different task at the same time making optimal use of the available resources specially multiple CPUs.

Question

When we play some games in the phone or pad,we need to use both hands to control handle at the same time. So design a thread representing task one to keep pressing the button on the left and

another thread representing pressing the button on the right. Create the two running threads, and show the running time since start. After 5 seconds stop task one, and after 10 seconds stop task two, to finish the concurrent running situation.

Code

```
import java.util.*;  
  
//create a thread: new thread? -> thread combined with interface  
"Runnable", so create class  
//output thread status: where?? -> three methods, public create; public  
void run(); public void start;  
//start a thread: thread.run()? -> Thread(Runnable this, String name);  
Thread.start(); Thread.sleep(1000);  
class TaskConcur implements Runnable {  
  
    private Thread T;  
    private String TaskName;  
    private int ProcDur;  
  
    //create  
    public TaskConcur(String strIn, int numDur) {  
        TaskName = strIn;  
        ProcDur = numDur;  
    }  
    public void run() {  
        //business logic  
        for(int i=ProcDur; i>=0; i--) {  
            try {  
                System.out.println(TaskName + "ing " + (ProcDur-i) + "  
seconds.");  
                Thread.sleep(1000);  
            } catch(InterruptedException e) {  
                System.out.println(TaskName + " interrupted");  
            }  
        }  
        System.out.println(TaskName + " exit.");  
    }  
    public void start() {  
        T = new Thread(this, TaskName);  
    }  
}
```

```

        T.start();
    }

}

public class MulThreadSample {
    public static void main(String[] args) {
        TaskConcur taskOne = new TaskConcur("Left Hand Press", 5);
        taskOne.start();
        TaskConcur taskTwo = new TaskConcur("Right Hand Press", 10);
        taskTwo.start();
    }
}

```

Output

```

Java - JavaConcept/src/MulThreadSample.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Java
Package Explorer
GenericSamp... MulThreadSa... CollectionSa... InterfaceSa...
01LocStor.js
02Media.html
03inputType.html
04Geoloc.html
JavaConcept
src
( default package )
Array.java
CollectionSample.java
GenericSample.java
InterfaceSample.java
MulThreadSample.java
QueueInOut.java
StackPush.java
JRE System Library [jre1.8.0_73]
js_concept
WebContent
01var.html
01var.js
02objects.html
02objects.js
03func.html
03func.js
04event.html
05array.html
06inherit.html
07cond.html
08regExp.html
09strictMode.html
10error.html
11typeConver.html
12json.html

```

```

1 import java.util.*;
2
3 //create a thread: new thread? -> thread combined with interface "Runnable", so create class
4 //output thread status: where?? -> three methods, public create; public void run(); public void start();
5 //start a thread: thread.run() -> Thread(Runnable this, String name); Thread.start();
6 class TaskConcur implements Runnable {
7
8     private Thread T;
9
10    public void start() {
11        T.start();
12    }
13
14    public void run() {
15        if (name.equals("Left Hand Press")) {
16            for (int i = 0; i < 5; i++) {
17                System.out.println("Left Hand Pressing " + i + " seconds.");
18            }
19        } else if (name.equals("Right Hand Press")) {
20            for (int i = 0; i < 10; i++) {
21                System.out.println("Right Hand Pressing " + i + " seconds.");
22            }
23        }
24    }
25
26    public String getName() {
27        return name;
28    }
29
30    public void setName(String name) {
31        this.name = name;
32    }
33
34    public void setRunTime(int runTime) {
35        this.runTime = runTime;
36    }
37
38    public int getRunTime() {
39        return runTime;
40    }
41
42    public void setPriority(int priority) {
43        this.priority = priority;
44    }
45
46    public int getPriority() {
47        return priority;
48    }
49
50    public void setThreadName(String name) {
51        this.name = name;
52    }
53
54    public String getThreadName() {
55        return name;
56    }
57
58    public void setThreadTime(int time) {
59        this.time = time;
60    }
61
62    public int getThreadTime() {
63        return time;
64    }
65
66    public void setThreadPriority(int priority) {
67        this.priority = priority;
68    }
69
70    public int getThreadPriority() {
71        return priority;
72    }
73
74    public void setThreadRunTime(int runTime) {
75        this.runTime = runTime;
76    }
77
78    public int getThreadRunTime() {
79        return runTime;
80    }
81
82    public void setThreadName(String name) {
83        this.name = name;
84    }
85
86    public String getThreadName() {
87        return name;
88    }
89
90    public void setThreadTime(int time) {
91        this.time = time;
92    }
93
94    public int getThreadTime() {
95        return time;
96    }
97
98    public void setThreadPriority(int priority) {
99        this.priority = priority;
100    }
101
102    public int getThreadPriority() {
103        return priority;
104    }
105
106    public void setThreadRunTime(int runTime) {
107        this.runTime = runTime;
108    }
109
110    public int getThreadRunTime() {
111        return runTime;
112    }
113
114    public void setThreadName(String name) {
115        this.name = name;
116    }
117
118    public String getThreadName() {
119        return name;
120    }
121
122    public void setThreadTime(int time) {
123        this.time = time;
124    }
125
126    public int getThreadTime() {
127        return time;
128    }
129
130    public void setThreadPriority(int priority) {
131        this.priority = priority;
132    }
133
134    public int getThreadPriority() {
135        return priority;
136    }
137
138    public void setThreadRunTime(int runTime) {
139        this.runTime = runTime;
140    }
141
142    public int getThreadRunTime() {
143        return runTime;
144    }
145
146    public void setThreadName(String name) {
147        this.name = name;
148    }
149
150    public String getThreadName() {
151        return name;
152    }
153
154    public void setThreadTime(int time) {
155        this.time = time;
156    }
157
158    public int getThreadTime() {
159        return time;
160    }
161
162    public void setThreadPriority(int priority) {
163        this.priority = priority;
164    }
165
166    public int getThreadPriority() {
167        return priority;
168    }
169
170    public void setThreadRunTime(int runTime) {
171        this.runTime = runTime;
172    }
173
174    public int getThreadRunTime() {
175        return runTime;
176    }
177
178    public void setThreadName(String name) {
179        this.name = name;
180    }
181
182    public String getThreadName() {
183        return name;
184    }
185
186    public void setThreadTime(int time) {
187        this.time = time;
188    }
189
190    public int getThreadTime() {
191        return time;
192    }
193
194    public void setThreadPriority(int priority) {
195        this.priority = priority;
196    }
197
198    public int getThreadPriority() {
199        return priority;
200    }
201
202    public void setThreadRunTime(int runTime) {
203        this.runTime = runTime;
204    }
205
206    public int getThreadRunTime() {
207        return runTime;
208    }
209
210    public void setThreadName(String name) {
211        this.name = name;
212    }
213
214    public String getThreadName() {
215        return name;
216    }
217
218    public void setThreadTime(int time) {
219        this.time = time;
220    }
221
222    public int getThreadTime() {
223        return time;
224    }
225
226    public void setThreadPriority(int priority) {
227        this.priority = priority;
228    }
229
230    public int getThreadPriority() {
231        return priority;
232    }
233
234    public void setThreadRunTime(int runTime) {
235        this.runTime = runTime;
236    }
237
238    public int getThreadRunTime() {
239        return runTime;
240    }
241
242    public void setThreadName(String name) {
243        this.name = name;
244    }
245
246    public String getThreadName() {
247        return name;
248    }
249
250    public void setThreadTime(int time) {
251        this.time = time;
252    }
253
254    public int getThreadTime() {
255        return time;
256    }
257
258    public void setThreadPriority(int priority) {
259        this.priority = priority;
260    }
261
262    public int getThreadPriority() {
263        return priority;
264    }
265
266    public void setThreadRunTime(int runTime) {
267        this.runTime = runTime;
268    }
269
270    public int getThreadRunTime() {
271        return runTime;
272    }
273
274    public void setThreadName(String name) {
275        this.name = name;
276    }
277
278    public String getThreadName() {
279        return name;
280    }
281
282    public void setThreadTime(int time) {
283        this.time = time;
284    }
285
286    public int getThreadTime() {
287        return time;
288    }
289
290    public void setThreadPriority(int priority) {
291        this.priority = priority;
292    }
293
294    public int getThreadPriority() {
295        return priority;
296    }
297
298    public void setThreadRunTime(int runTime) {
299        this.runTime = runTime;
300    }
301
302    public int getThreadRunTime() {
303        return runTime;
304    }
305
306    public void setThreadName(String name) {
307        this.name = name;
308    }
309
310    public String getThreadName() {
311        return name;
312    }
313
314    public void setThreadTime(int time) {
315        this.time = time;
316    }
317
318    public int getThreadTime() {
319        return time;
320    }
321
322    public void setThreadPriority(int priority) {
323        this.priority = priority;
324    }
325
326    public int getThreadPriority() {
327        return priority;
328    }
329
330    public void setThreadRunTime(int runTime) {
331        this.runTime = runTime;
332    }
333
334    public int getThreadRunTime() {
335        return runTime;
336    }
337
338    public void setThreadName(String name) {
339        this.name = name;
340    }
341
342    public String getThreadName() {
343        return name;
344    }
345
346    public void setThreadTime(int time) {
347        this.time = time;
348    }
349
350    public int getThreadTime() {
351        return time;
352    }
353
354    public void setThreadPriority(int priority) {
355        this.priority = priority;
356    }
357
358    public int getThreadPriority() {
359        return priority;
360    }
361
362    public void setThreadRunTime(int runTime) {
363        this.runTime = runTime;
364    }
365
366    public int getThreadRunTime() {
367        return runTime;
368    }
369
370    public void setThreadName(String name) {
371        this.name = name;
372    }
373
374    public String getThreadName() {
375        return name;
376    }
377
378    public void setThreadTime(int time) {
379        this.time = time;
380    }
381
382    public int getThreadTime() {
383        return time;
384    }
385
386    public void setThreadPriority(int priority) {
387        this.priority = priority;
388    }
389
390    public int getThreadPriority() {
391        return priority;
392    }
393
394    public void setThreadRunTime(int runTime) {
395        this.runTime = runTime;
396    }
397
398    public int getThreadRunTime() {
399        return runTime;
400    }
401
402    public void setThreadName(String name) {
403        this.name = name;
404    }
405
406    public String getThreadName() {
407        return name;
408    }
409
410    public void setThreadTime(int time) {
411        this.time = time;
412    }
413
414    public int getThreadTime() {
415        return time;
416    }
417
418    public void setThreadPriority(int priority) {
419        this.priority = priority;
420    }
421
422    public int getThreadPriority() {
423        return priority;
424    }
425
426    public void setThreadRunTime(int runTime) {
427        this.runTime = runTime;
428    }
429
430    public int getThreadRunTime() {
431        return runTime;
432    }
433
434    public void setThreadName(String name) {
435        this.name = name;
436    }
437
438    public String getThreadName() {
439        return name;
440    }
441
442    public void setThreadTime(int time) {
443        this.time = time;
444    }
445
446    public int getThreadTime() {
447        return time;
448    }
449
450    public void setThreadPriority(int priority) {
451        this.priority = priority;
452    }
453
454    public int getThreadPriority() {
455        return priority;
456    }
457
458    public void setThreadRunTime(int runTime) {
459        this.runTime = runTime;
460    }
461
462    public int getThreadRunTime() {
463        return runTime;
464    }
465
466    public void setThreadName(String name) {
467        this.name = name;
468    }
469
470    public String getThreadName() {
471        return name;
472    }
473
474    public void setThreadTime(int time) {
475        this.time = time;
476    }
477
478    public int getThreadTime() {
479        return time;
480    }
481
482    public void setThreadPriority(int priority) {
483        this.priority = priority;
484    }
485
486    public int getThreadPriority() {
487        return priority;
488    }
489
490    public void setThreadRunTime(int runTime) {
491        this.runTime = runTime;
492    }
493
494    public int getThreadRunTime() {
495        return runTime;
496    }
497
498    public void setThreadName(String name) {
499        this.name = name;
500    }
501
502    public String getThreadName() {
503        return name;
504    }
505
506    public void setThreadTime(int time) {
507        this.time = time;
508    }
509
510    public int getThreadTime() {
511        return time;
512    }
513
514    public void setThreadPriority(int priority) {
515        this.priority = priority;
516    }
517
518    public int getThreadPriority() {
519        return priority;
520    }
521
522    public void setThreadRunTime(int runTime) {
523        this.runTime = runTime;
524    }
525
526    public int getThreadRunTime() {
527        return runTime;
528    }
529
530    public void setThreadName(String name) {
531        this.name = name;
532    }
533
534    public String getThreadName() {
535        return name;
536    }
537
538    public void setThreadTime(int time) {
539        this.time = time;
540    }
541
542    public int getThreadTime() {
543        return time;
544    }
545
546    public void setThreadPriority(int priority) {
547        this.priority = priority;
548    }
549
550    public int getThreadPriority() {
551        return priority;
552    }
553
554    public void setThreadRunTime(int runTime) {
555        this.runTime = runTime;
556    }
557
558    public int getThreadRunTime() {
559        return runTime;
560    }
561
562    public void setThreadName(String name) {
563        this.name = name;
564    }
565
566    public String getThreadName() {
567        return name;
568    }
569
570    public void setThreadTime(int time) {
571        this.time = time;
572    }
573
574    public int getThreadTime() {
575        return time;
576    }
577
578    public void setThreadPriority(int priority) {
579        this.priority = priority;
580    }
581
582    public int getThreadPriority() {
583        return priority;
584    }
585
586    public void setThreadRunTime(int runTime) {
587        this.runTime = runTime;
588    }
589
590    public int getThreadRunTime() {
591        return runTime;
592    }
593
594    public void setThreadName(String name) {
595        this.name = name;
596    }
597
598    public String getThreadName() {
599        return name;
600    }
601
602    public void setThreadTime(int time) {
603        this.time = time;
604    }
605
606    public int getThreadTime() {
607        return time;
608    }
609
610    public void setThreadPriority(int priority) {
611        this.priority = priority;
612    }
613
614    public int getThreadPriority() {
615        return priority;
616    }
617
618    public void setThreadRunTime(int runTime) {
619        this.runTime = runTime;
620    }
621
622    public int getThreadRunTime() {
623        return runTime;
624    }
625
626    public void setThreadName(String name) {
627        this.name = name;
628    }
629
630    public String getThreadName() {
631        return name;
632    }
633
634    public void setThreadTime(int time) {
635        this.time = time;
636    }
637
638    public int getThreadTime() {
639        return time;
640    }
641
642    public void setThreadPriority(int priority) {
643        this.priority = priority;
644    }
645
646    public int getThreadPriority() {
647        return priority;
648    }
649
650    public void setThreadRunTime(int runTime) {
651        this.runTime = runTime;
652    }
653
654    public int getThreadRunTime() {
655        return runTime;
656    }
657
658    public void setThreadName(String name) {
659        this.name = name;
660    }
661
662    public String getThreadName() {
663        return name;
664    }
665
666    public void setThreadTime(int time) {
667        this.time = time;
668    }
669
670    public int getThreadTime() {
671        return time;
672    }
673
674    public void setThreadPriority(int priority) {
675        this.priority = priority;
676    }
677
678    public int getThreadPriority() {
679        return priority;
680    }
681
682    public void setThreadRunTime(int runTime) {
683        this.runTime = runTime;
684    }
685
686    public int getThreadRunTime() {
687        return runTime;
688    }
689
690    public void setThreadName(String name) {
691        this.name = name;
692    }
693
694    public String getThreadName() {
695        return name;
696    }
697
698    public void setThreadTime(int time) {
699        this.time = time;
700    }
701
702    public int getThreadTime() {
703        return time;
704    }
705
706    public void setThreadPriority(int priority) {
707        this.priority = priority;
708    }
709
710    public int getThreadPriority() {
711        return priority;
712    }
713
714    public void setThreadRunTime(int runTime) {
715        this.runTime = runTime;
716    }
717
718    public int getThreadRunTime() {
719        return runTime;
720    }
721
722    public void setThreadName(String name) {
723        this.name = name;
724    }
725
726    public String getThreadName() {
727        return name;
728    }
729
730    public void setThreadTime(int time) {
731        this.time = time;
732    }
733
734    public int getThreadTime() {
735        return time;
736    }
737
738    public void setThreadPriority(int priority) {
739        this.priority = priority;
740    }
741
742    public int getThreadPriority() {
743        return priority;
744    }
745
746    public void setThreadRunTime(int runTime) {
747        this.runTime = runTime;
748    }
749
750    public int getThreadRunTime() {
751        return runTime;
752    }
753
754    public void setThreadName(String name) {
755        this.name = name;
756    }
757
758    public String getThreadName() {
759        return name;
760    }
761
762    public void setThreadTime(int time) {
763        this.time = time;
764    }
765
766    public int getThreadTime() {
767        return time;
768    }
769
770    public void setThreadPriority(int priority) {
771        this.priority = priority;
772    }
773
774    public int getThreadPriority() {
775        return priority;
776    }
777
778    public void setThreadRunTime(int runTime) {
779        this.runTime = runTime;
780    }
781
782    public int getThreadRunTime() {
783        return runTime;
784    }
785
786    public void setThreadName(String name) {
787        this.name = name;
788    }
789
790    public String getThreadName() {
791        return name;
792    }
793
794    public void setThreadTime(int time) {
795        this.time = time;
796    }
797
798    public int getThreadTime() {
799        return time;
800    }
801
802    public void setThreadPriority(int priority) {
803        this.priority = priority;
804    }
805
806    public int getThreadPriority() {
807        return priority;
808    }
809
810    public void setThreadRunTime(int runTime) {
811        this.runTime = runTime;
812    }
813
814    public int getThreadRunTime() {
815        return runTime;
816    }
817
818    public void setThreadName(String name) {
819        this.name = name;
820    }
821
822    public String getThreadName() {
823        return name;
824    }
825
826    public void setThreadTime(int time) {
827        this.time = time;
828    }
829
830    public int getThreadTime() {
831        return time;
832    }
833
834    public void setThreadPriority(int priority) {
835        this.priority = priority;
836    }
837
838    public int getThreadPriority() {
839        return priority;
840    }
841
842    public void setThreadRunTime(int runTime) {
843        this.runTime = runTime;
844    }
845
846    public int getThreadRunTime() {
847        return runTime;
848    }
849
850    public void setThreadName(String name) {
851        this.name = name;
852    }
853
854    public String getThreadName() {
855        return name;
856    }
857
858    public void setThreadTime(int time) {
859        this.time = time;
860    }
861
862    public int getThreadTime() {
863        return time;
864    }
865
866    public void setThreadPriority(int priority) {
867        this.priority = priority;
868    }
869
870    public int getThreadPriority() {
871        return priority;
872    }
873
874    public void setThreadRunTime(int runTime) {
875        this.runTime = runTime;
876    }
877
878    public int getThreadRunTime() {
879        return runTime;
880    }
881
882    public void setThreadName(String name) {
883        this.name = name;
884    }
885
886    public String getThreadName() {
887        return name;
888    }
889
890    public void setThreadTime(int time) {
891        this.time = time;
892    }
893
894    public int getThreadTime() {
895        return time;
896    }
897
898    public void setThreadPriority(int priority) {
899        this.priority = priority;
900    }
901
902    public int getThreadPriority() {
903        return priority;
904    }
905
906    public void setThreadRunTime(int runTime) {
907        this.runTime = runTime;
908    }
909
910    public int getThreadRunTime() {
911        return runTime;
912    }
913
914    public void setThreadName(String name) {
915        this.name = name;
916    }
917
918    public String getThreadName() {
919        return name;
920    }
921
922    public void setThreadTime(int time) {
923        this.time = time;
924    }
925
926    public int getThreadTime() {
927        return time;
928    }
929
930    public void setThreadPriority(int priority) {
931        this.priority = priority;
932    }
933
934    public int getThreadPriority() {
935        return priority;
936    }
937
938    public void setThreadRunTime(int runTime) {
939        this.runTime = runTime;
940    }
941
942    public int getThreadRunTime() {
943        return runTime;
944    }
945
946    public void setThreadName(String name) {
947        this.name = name;
948    }
949
950    public String getThreadName() {
951        return name;
952    }
953
954    public void setThreadTime(int time) {
955        this.time = time;
956    }
957
958    public int getThreadTime() {
959        return time;
960    }
961
962    public void setThreadPriority(int priority) {
963        this.priority = priority;
964    }
965
966    public int getThreadPriority() {
967        return priority;
968    }
969
970    public void setThreadRunTime(int runTime) {
971        this.runTime = runTime;
972    }
973
974    public int getThreadRunTime() {
975        return runTime;
976    }
977
978    public void setThreadName(String name) {
979        this.name = name;
980    }
981
982    public String getThreadName() {
983        return name;
984    }
985
986    public void setThreadTime(int time) {
987        this.time = time;
988    }
989
990    public int getThreadTime() {
991        return time;
992    }
993
994    public void setThreadPriority(int priority) {
995        this.priority = priority;
996    }
997
998    public int getThreadPriority() {
999        return priority;
1000    }
1001
1002    public void setThreadRunTime(int runTime) {
1003        this.runTime = runTime;
1004    }
1005
1006    public int getThreadRunTime() {
1007        return runTime;
1008    }
1009
1010    public void setThreadName(String name) {
1011        this.name = name;
1012    }
1013
1014    public String getThreadName() {
1015        return name;
1016    }
1017
1018    public void setThreadTime(int time) {
1019        this.time = time;
1020    }
1021
1022    public int getThreadTime() {
1023        return time;
1024    }
1025
1026    public void setThreadPriority(int priority) {
1027        this.priority = priority;
1028    }
1029
1030    public int getThreadPriority() {
1031        return priority;
1032    }
1033
1034    public void setThreadRunTime(int runTime) {
1035        this.runTime = runTime;
1036    }
1037
1038    public int getThreadRunTime() {
1039        return runTime;
1040    }
1041
1042    public void setThreadName(String name) {
1043        this.name = name;
1044    }
1045
1046    public String getThreadName() {
1047        return name;
1048    }
1049
1050    public void setThreadTime(int time) {
1051        this.time = time;
1052    }
1053
1054    public int getThreadTime() {
1055        return time;
1056    }
1057
1058    public void setThreadPriority(int priority) {
1059        this.priority = priority;
1060    }
1061
1062    public int getThreadPriority() {
1063        return priority;
1064    }
1065
1066    public void setThreadRunTime(int runTime) {
1067        this.runTime = runTime;
1068    }
1069
1070    public int getThreadRunTime() {
1071        return runTime;
1072    }
1073
1074    public void setThreadName(String name) {
1075        this.name = name;
1076    }
1077
1078    public String getThreadName() {
1079        return name;
1080    }
1081
1082    public void setThreadTime(int time) {
1083        this.time = time;
1084    }
1085
1086    public int getThreadTime() {
1087        return time;
1088    }
1089
1090    public void setThreadPriority(int priority) {
1091        this.priority = priority;
1092    }
1093
1094    public int getThreadPriority() {
1095        return priority;
1096    }
1097
1098    public void setThreadRunTime(int runTime) {
1099        this.runTime = runTime;
1100    }
1101
1102    public int getThreadRunTime() {
1103        return runTime;
1104    }
1105
1106    public void setThreadName(String name) {
1107        this.name = name;
1108    }
1109
1110    public String getThreadName() {
1111        return name;
1112    }
1113
1114    public void setThreadTime(int time) {
1115        this.time = time;
1116    }
1117
1118    public int getThreadTime() {
1119        return time;
1120    }
1121
1122    public void setThreadPriority(int priority) {
1123        this.priority = priority;
1124    }
1125
1126    public int getThreadPriority() {
1127        return priority;
1128    }
1129
1130    public void setThreadRunTime(int runTime) {
1131        this.runTime = runTime;
1132    }
1133
1134    public int getThreadRunTime() {
1135        return runTime;
1136    }
1137
1138    public void setThreadName(String name) {
1139        this.name = name;
1140    }
1141
1142    public String getThreadName() {
1143        return name;
1144    }
1145
1146    public void setThreadTime(int time) {
1147        this.time = time;
1148    }
1149
1150    public int getThreadTime() {
1151        return time;
1152    }
1153
1154    public void setThreadPriority(int priority) {
1155        this.priority = priority;
1156    }
1157
1158    public int getThreadPriority() {
1159        return priority;
1160    }
1161
1162    public void setThreadRunTime(int runTime) {
1163        this.runTime = runTime;
1164    }
1165
1166    public int getThreadRunTime() {
1167        return runTime;
1168    }
1169
1170    public void setThreadName(String name) {
1171        this.name = name;
1172    }
1173
1174    public String getThreadName() {
1175        return name;
1176    }
1177
1178    public void setThreadTime(int time) {
1179        this.time = time;
1180    }
1181
1182    public int getThreadTime() {
1183        return time;
1184    }
1185
1186    public void setThreadPriority(int priority) {
1187        this.priority = priority;
1188    }
1189
1190    public int getThreadPriority() {
1191        return priority;
1192    }
1193
1194    public void setThreadRunTime(int runTime) {
1195        this.runTime = runTime;
1196    }
1197
1198    public int getThreadRunTime() {
1199        return runTime;
1200    }
1201
1202    public void setThreadName(String name) {
1203        this.name = name;
1204    }
1205
1206    public String getThreadName() {
1207        return name;
1208    }
1209
1210    public void setThreadTime(int time) {
1211        this.time = time;
1212    }
1213
1214    public int getThreadTime() {
1215        return time;
1216    }
1217
1218    public void setThreadPriority(int priority) {
1219        this.priority = priority;
1220    }
1221
1222    public int getThreadPriority() {
1223        return priority;
1224    }
1225
1226    public void setThreadRunTime(int runTime) {
1227        this.runTime = runTime;
1228    }
1229
1230    public int getThreadRunTime() {
1231        return runTime;
1232    }
1233
1234    public void setThreadName(String name) {
1235        this.name = name;
1236    }
1237
1238    public String getThreadName() {
1239        return name;
1240    }
1241
1242    public void setThreadTime(int time) {
1243        this.time = time;
1244    }
1245
1246    public int getThreadTime() {
1247        return time;
1248    }
1249
1250    public void setThreadPriority(int priority) {
1251        this.priority = priority;
1252    }
1253
1254    public int getThreadPriority() {
1255        return priority;
1256    }
1257
1258    public void setThreadRunTime(int runTime) {
1259        this.runTime = runTime;
1260    }
1261
1262    public int getThreadRunTime() {
1263        return runTime;
1264    }
1265
1266    public void setThreadName(String name) {
1267        this.name = name;
1268    }
1269
1270    public String getThreadName() {
1271        return name;
1272    }
1273
1274    public void setThreadTime(int time) {
1275        this.time = time;
1276    }
1277
1278    public int getThreadTime() {
1279        return time;
1280    }
1281
1282    public void setThreadPriority(int priority) {
1283        this.priority = priority;
1284    }
1285
1286    public int getThreadPriority() {
1287        return priority;
1288    }
1289
1290    public void setThreadRunTime(int runTime) {
1291        this.runTime = runTime;
1292    }
1293
1294    public int getThreadRunTime() {
1295        return runTime;
1296    }
1297
1298    public void setThreadName(String name) {
1299        this.name = name;
1300    }
1301
1302    public String getThreadName() {
1303        return name;
1304    }
1305
1306    public void setThreadTime(int time) {
1307        this.time = time;
1308    }
1309
1310    public int getThreadTime() {
1311        return time;
1312    }
1313
1314    public void setThreadPriority(int priority) {
1315        this.priority = priority;
1316    }
1317
1318    public int getThreadPriority() {
1319        return priority;
1320    }
1321
1322    public void setThreadRunTime(int runTime) {
1323        this.runTime = runTime;
1324    }
1325
1326    public int getThreadRunTime() {
1327        return runTime;
1328    }
1329
1330    public void setThreadName(String name) {
1331        this.name = name;
1332    }
1333
1334    public String getThreadName() {
1335        return name;
1336    }
1337
1338    public void setThreadTime(int time) {
1339        this.time = time;
1340    }
1341
1342    public int getThreadTime() {
1343        return time;
1344    }
1345
1346    public void setThreadPriority(int priority) {
1347        this.priority = priority;
1348    }
1349
1350    public int getThreadPriority() {
1351        return priority;
1352    }
1353
1354    public void setThreadRunTime(int runTime) {
1355        this.runTime = runTime;
1356    }
1357
1358    public int getThreadRunTime() {
1359        return runTime;
1360    }
1361
1362    public void setThreadName(String name) {
1363        this.name = name;
1364    }
1365
1366    public
```