

**动态Java计算器课程**

**课 程 名 称：Dynamic java**

**课程设计名称：Dynamic Java Calculator course**

**系部名称：International Education Center**

**学生姓名：Arafat Hoshen**

**班 级：class 3**

**学 号： 2…**

**成 绩：**

### **Index**

# [**Introduction to Dynamic Programming: Java……..**](#_Toc51520051) **1**

### **OutLook of My Dynamic calculator……………………..2**

### **source code of Dynamic calculator……………………..3**

**servlet File……………………………1**

**jsp File…………………………………1**

**java File………………………………1**

**Acknowledge …………………………………………………….4**

# **1.Introduction to Dynamic Programming: Java**

## **What is Dynamic Programming?**

Dynamic Programming is a programming technique used to solve recursive problems more efficiently. Specifically, it adds time efficiency, and it does so by taking advantage of data structures to store reusable solutions to intermediate steps, thus saving redundant computations. It’s a way of solving problems with recursive relationships by solving smaller problems and building up to the solution to the original problem.

Let’s take a look at a simple algorithm that can get computationally complex very quickly, and then let’s use dynamic programming to increase its efficiency.

## Fibonacci

The Fibonacci series is a classic mathematical series in which each number is equal to the sum of the two numbers before it, always starting with 0 and 1:

0, 1, 1, 2, 3, 5, 8, 13, 21, etc.

The 0th Fibonacci number is always 0 and first Fibonacci number is always 1. So the second Fibonacci number is 0 + 1 = 1, third Fibonacci number is 1 + 1 = 2, and so on. You could calculate the nth number iteratively this way, but you could also calculate it recursively:

fib(n)

if n is 0 or 1

return n

else

return fib(n - 1) + fib(n - 2)

This technique breaks up calculating the nth number into many smaller problems, calculating each step as the sum of calculating the previous two numbers.

Although this technique will certainly work to find the correct number, as n grows, the number of recursive calls grows very quickly. Let’s visualize all the function calls if we were to calculate the fourth Fibonacci number:

fib(4) -> fib(3) + fib(2)

fib(3) -> fib(2) + fib(1)

fib(2) -> fib(1) + fib(0)

fib(2) -> fib(1) + fib(0)

As you can seefib(2) is called twice, fib(1) is called three times. If n were larger than 4, you’d see these numbers of calls get high very quickly. For instance, to calculate the 10th number, we’d make 34 calls to fib(2) and 177 total function calls! Why do we need to call the same function multiple times with the same input?

We don’t! We can use a dynamic programming technique called memoization to cut down greatly on the number of function calls necessary to calculate the correct number.

## Memoization

Memoization is a specialized form of caching used to store the result of a function call. The next time that function is called, if the result of that function call is already stored somewhere, we’ll retrieve that instead of running the function itself again. Memoization can result in much faster overall execution times (although it can increase memory requirements as function results are stored in memory).

Memoization is a great technique to use alongside recursion. The memo can even be saved between function calls if it’s being used for common calculations in a program.

Memoizing Fibonacci:In the previous example, many function calls to fib() were redundant. Let’s memoize it in order to speed up execution.

To begin, we’ll use a Java HashMap to store the memoized values. Each key will represent n (starting from 0), and the corresponding value will be the result of that Fibonacci number. Then, whenever we need to calculate a number, if it’s already been calculated, we can retrieve the value from the map in O(1) timeIn pseudocode, our approach to memoization will look something like this:

Create a memo map

fibMemo(n, map)

if n is 0 or 1

return n

if n key exists in map

return map.get(n)

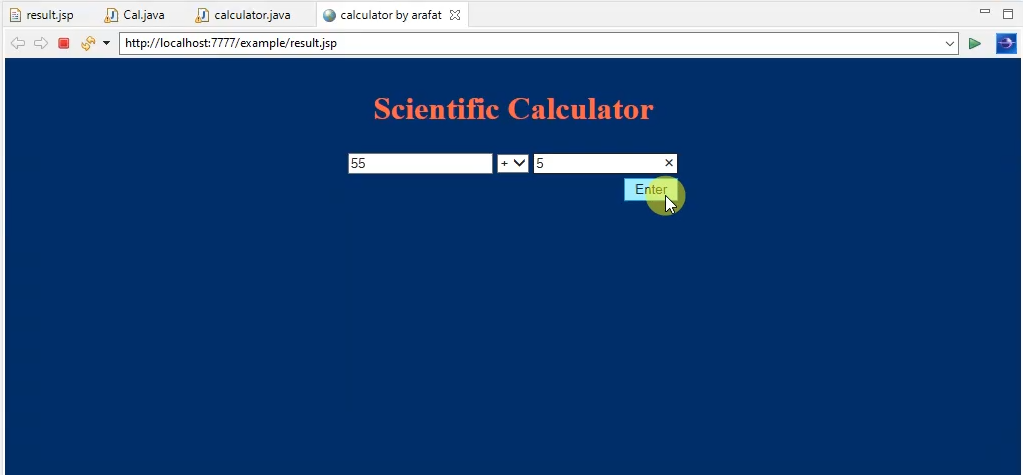
else

calculate current fibonacci number through a recursive call

store value in map

return value

### **3.OutLook of My Dynamic calculator**



**1.Servlet File**

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.RequestDispatcher;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** example.calculator;

/\*\*

\* Servlet implementation class calculator

\*/

@WebServlet("/Cal4")

**public** **class** Cal **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** Cal() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\* **@see** HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

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

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* **@see** HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

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

//doGet(request, response);

PrintWriter out = response.getWriter();

String firstNum = request.getParameter("firstNum");

String secondNum = request.getParameter("secondNum");

String operator = request.getParameter("operator");

**double** result = 0;

calculator cal = **new** calculator();

RequestDispatcher rd = request.getRequestDispatcher("result.jsp");

//out.println(operator ! = "-1")

**if**(firstNum !=**null** && secondNum !=**null** && operator != **null** && !"-1".contentEquals(operator)) {

result = cal.calculator(firstNum, operator, secondNum);

//out.println(firstNum + operator + secondNum + "=" + result);

request.setAttribute("result", result);

}

rd.forward(request, response);

}

}

**2.jsp File**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>calculator by arafat</title>

</head>

<body>

<body style="background-color:*rgb(4, 47, 102)*;">

<form action=*"Cal4"* method=*"post"*>

<table align=*"center"* border=*"0"*>

<tr>

<th><h1 style="color:*Tomato*;"> Scientific Calculator </h1></th>

</tr>

<tr>

<td>

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

<select name=*"operator"*>

<option value=*"+"*>+</option>

<option value"-">-</option>

<option value=*"\*"*>\*</option>

<option value=*"/"*>/</option>

</select>

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

</td>

</tr>

<tr>

<td align=*"right"*> <input type=*"submit"* value=*"Enter"*></td>

</tr>

<tr>

<td><%

**if**(request.getAttribute("result") != **null**){

out.println(request.getAttribute("result"));

}

%></td>

</tr>

</table>

</form>

</body>

</body>

</html>

**3.java File**

**package** example;

**public** **class** calculator {

**public** **double** calculator(String firstNum, String operator, String secondNum) {

**double** result = 0;

**if**("+".contentEquals(operator)) {

result = Double.*valueOf*(firstNum) + Double.*valueOf*(secondNum);

}**else** **if**("-".equals(operator)) {

result = Double.*valueOf*(firstNum) - Double.*valueOf*(secondNum);

}**else** **if**("\*".equals(operator)) {

result = Double.*valueOf*(firstNum) \* Double.*valueOf*(secondNum);

}**else** **if**("/".equals(operator)) {

result = Double.*valueOf*(firstNum) / Double.*valueOf*(secondNum);

}

**return** result;

}

}

### **4.Acknowledge**

I would like to express my special thanks of gratitude to my teacher jim laoshi as well as our professor Jim.who gave me the golden opportunity to do this wonderful project on the topic making calculator by using Dynamic java.which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them.

Secondly i would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

\*\*\* laoshi my request you play my game on your computer \*\*\*