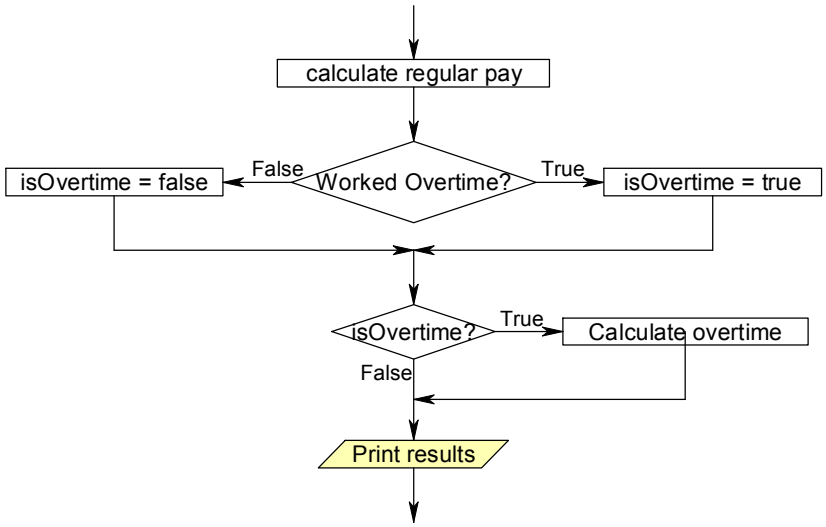
Virtual Lecture Notes (Part 2)

A classic assignment in an introductory programming course is to determine an employee's salary based on the hourly pay rate and the number of hours worked. If the employee works more that 40 hours, time-and-a-half is paid for the overtime hours.
The Salary_v1 class is a program designed to calculate the regular or overtime salary of an employee.

- Carefully study the source code for the Salary_v1 class you downloaded previously.
- Run the program and observe its performance and output.

This overall structure of the Salary_v1 class should seem very familiar to you after studying the GPA_v1 class. For example, take a closer look at the section of the flowchart shown below.



If you compare this to the flowchart for Salary_v1, you will notice some decision points where the flow of control branches based on evaluation of a boolean condition. Compare the flowchart with the corresponding code segment below.

```
...
<21>      double totalSalary = totalHours * payRate;
<22>
<23>      boolean isOvertime = totalHours > 40;
<24>
<25>      if(isOvertime)
<26>      totalSalary += (totalHours-40)*payRate/2;
...
```

Line <21> calculates the default value for **`totalSalary`**.
Line <23> declares **`isOvertime`** to be a **`boolean`** primitive data type and assigns **`true`** or **`false`** to the variable based on whether the total hours worked is greater than 40.
Line <25> evaluates the value of **`isOvertime`** (i.e. true or false).
Line <26> calculates the additional overtime pay and adds it to the **`totalSalary`** variable.

Continue to analyze the program line-by-line and make sure you understand the syntax and purpose of each statement.

If you have ever been employed and eligible for overtime, the calculation on Line <26> may seem a little unorthodox. Can you think of another way to calculate the salary of an employee?

You are quickly moving on to more advanced programming topics, many of which will depend on the use of boolean variables and conditional statements. As your experience grows from writing more programs and analyzing more code, begin to notice the design patterns that re-appear over and over again.