اور جو اس کے روبرو ایماندار ہو کر آئے گا اور عمل بھی نیک کئے ہوں گے تو ایسے لوگوں کے لئے اونچے اونچے درجے ہیں ۔ طہٰ آیت ۷۵

Insanity: doing the same thing over and over again and expecting different results. Albert Einstein

Lecture 08

Further Decision Making If & Nested If Statements

In this lesson we will learn how to do decision making using if statements and later we will do programming practice. On first page there is review of previous lecture.

Review of Previous Lecture

Selection is one of basic elements of programming. Selection allows programmers to write statements that can be automatically selected or rejected during the execution of program. Selection allows a provision to handle different inputs and their related actions. Like if salary is below limit tax rate is different and otherwise tax rate is different. Similarly if education is masters or above, select candidate, else if education is bachelor and experience is more than 2 years select candidate otherwise reject candidate.

Normally If statement is used for selection in programming. There are three flavors of if statement:

1. Single if statement – to reject or select a statement
2. if else statement – to select one of the two statements
3. if-else chain – to select one or none out of multiple statements

See table for examples:

`days=30;`
`if (m==1 || m==3 || ...)`
`   days = 31;`
`if (m==2)`
`   days = 28;`
`wDays = days - holidays;`
	`if (amount>=balance)`
	`   ? ("Short of Bal");`
	`else{`
	`   newB = balance-amount;`
	`   balance = newB;`
	`   ? ("New Balance: ");`
	`   ? (balance);`
	`}`
		`if (s<=100000)`
		`   tax = 0;`
		`else if (s<=200000)`
		`   tax = s*0.1;`
		`else if (s<=300000)`
		`   tax = s*0.15;`
		`else if (s<=400000)`
		`   tax = s*0.2;`
		`else`
		`   tax = s*0.25;`

wDays for working day days are initialized by 30 but for January, March,…, December days are 31 and for February 28	? use for System.out.print	s for salary, tax calculated according to salary slab

Comparison

Students should understand that comparing primitive variables **[**int, double, char**]** is different from comparing other objects like String. To compare primitive variables relational operators **[**==, < , >, <=, >=, !=**]** are used, whereas, to compare objects following two methods used:

- equals	s1.equals(s2)	return true/ false
- compareTo	s1.compareTo(s2)	return 0, +ive, -ive values showing equal, greater, less

if (s1.compareTo(s2)==0)) `? "Equal"`

if (s1.compareTo(s2)<=0)) `? "S1 is smaller"`

Next Lecture

Now we proceed ahead there are some common errors using if statement:

- Writing multiple statements without creating block
- Missing parenthesis writing condition of if statement
- Thinking that if statement will do something more than just executing statements written. This is very dangerous to think that computer is intelligent and should do something according to your expectation. Computer performs only what you write. Therefore, in any case only statements written will be executed and statements written after if block will also executed unless ***return*** statement is given inside block

Comparing Decimals

Comparing decimal numbers is very critical for example consider following line:

if (10/9.0==1.1)   result in false because 10/9.0 is 1.11111…, to handle if you write

if (10/9.0>=1.1)   result in true, but it is true for any value less than equal 1.1

In such cases solution is to check range by adding and subtracting some value to one side like:
if (1.1+0.02>=10/9.0 && 1.1-0.02<=10/9.0) results true, and is true for values within range 1.08 to 1.12 and you may confine it further by making factor value smaller like if we add and subtract 0.015 valid range will confine 1.085 to 1.115.

Nested If

Sometimes we need decision making in levels for example tax slabs for salaried person may be different from person doing business. Therefore at first level you have to take decision what is the category of person for tax purpose and next to choose appropriate tax slab. See pseudo code:

```
if category is salaried
    if income <= 200000
        tax = income * 0.10
    else if income <= 300000
        tax = income * 0.20
    else
        tax = income * 0.25
else if category is business
    if income <= 250000
        tax = income * 0.15
    else if income <= 350000
        tax = income * 0.25
    else
        tax = income * 0.30
```

We may handle nested if with composite if conditions but once again it is difficult to write code and also it is expensive because in worst case there will be double comparisons. See example:

```
if category is salaried and income <= 200000
    tax = income * 0.10
else if category is salaried and income <= 300000
    tax = income * 0.20
else if category is salaried
    tax = income * 0.25
else if category is business and income <= 250000
    tax = income * 0.15
...
```

In above code almost every if followed by composite conditions where one condition is repeating, therefore, at the time of execution it may passes same condition is examined multiple times; whereas in previous code no condition is repeating.

Syntax

In case of **nested if** block is mandatory, otherwise there may be logical or syntax error occur. Like:

```
if (cond…){
        nested if block…
}
else if (cond…){
        nested if block…
}
```

See above example in java coding:

```
if (category.equals("salaried"){
        if (income <= 200000)
                tax = income * 0.10;
        else if (income <= 300000)
                tax = income * 0.20;
        else
                tax = income * 0.25;
}
else if (category.equals("business"){
        if (income <= 250000)
                tax = income * 0.15;
        else if (income <= 350000)
                tax = income * 0.25;
        else
                tax = income * 0.30;
}
```

see another example to write a, b, c 3 integers in order using nested if:

```
if (a<b){
     if (b<c)
          System.out.println(a+","+b+","+c);
     else if (a<c)
          System.out.println(a+","+c+","+b);
     else
          System.out.println(c+","+a+","+b);
     }
else{
     if (a<c)
          System.out.println(b+","+a+","+c);
     else if (b<c)
          System.out.println(b+","+c+","+a);
     else
          System.out.println(c+","+b+","+a);
}
```

Not only code is simpler but a lot more efficient.

Consider another example [Reference Big JAVA Exercise P5.9.] A year with 366 days is called a leap year. A year is a leap year if it is divisible by 4 (for example, 1980). However, since the introduction of the Gregorian calendar on October 15, 1582, a year is not a leap year if it is divisible by 100 (for example, 1900); however, it is a leap year if it is divisible by 400 (for example, 2000). Write a program that asks the user to enter year. Find and show year is leap or not?

```
//Year input here
if (year%4==0){//to check year is divisible by 4 or not
      if (year%400==0) //to check year is divisible by 400 or not
            System.out.println("Leap year");
      else if (year%100==0) //to check year is divisible by 100 or not
            System.out.println("Not a leap year");
      else
            System.out.println("Leap year");
      }
else
      System.out.println("Not a leap year");
```

Check code working fine and explore other ways to handle same problem. There are 3 to 4 different obvious ways to write this code. Exploring them and implementing correctly will improve your understanding.

Solve Homework 4 for further practice…coming soon