# Simple for loop

Let us revise about for loops which you have learnt before in C105 - Introduction to Programming and C208 - Object Oriented Programming modules

for (initialization; condition; update) {

statements

}

1. Write down the output of the code given below. Write down the difference between the two for loops

|  |
| --- |
| int j;  for ( j = 0; j < 5; j++ )  {  System.out.print( j + " " );  }  System.out.println( ); |

**Code Segment A**

|  |
| --- |
| int j;  for ( j = 5; j > 0; j-- )  {  System.out.print( j + " " );  }  System.out.println( ); |

**Code Segment B**

1. Write down the output of the code segment A.

|  |
| --- |
|  |

1. Write down the output of the code segment B.

|  |
| --- |
|  |

1. Write down the difference between code segment A and code segment B

|  |
| --- |
|  |

# Nested for loop

A nested for loop is a for loop within a for loop. Refer to the following link for more information.

<http://mathbits.com/MathBits/Java/Looping/NestedFor.htm>

Study the code below which prints a series of *two* digit numbers. For each number printed, the *first* and *second* digits are related to the i and j variables respectively.

|  |
| --- |
| **public** **class** TwoDimensionalForLoop {  **public** **static** **void** main(String args[]) {  **for** (**int** i = 0; i < 3; i++) {  **for** (**int** j = 2; j < 4; j++) {  System.*out*.print(" i = " + i + " , " + " j = " + j);  }  System.*out*.println();  }  }  } |

1. Write the output of this code.

|  |
| --- |
|  |

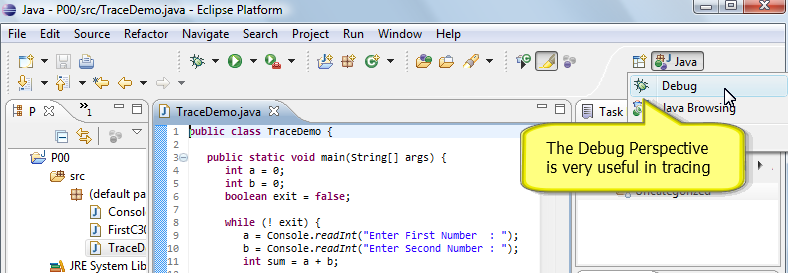
1. Study the code given below which finds the max and min element in the 2D array. Fill in the blanks in the code.

|  |  |  |
| --- | --- | --- |
| import java.io.\* ;  class ArrayMaxMin  {  public static void main ( String[] args )  {  int[][] data = { {3, 2, 5, 7, 1 },  {1, 4, 4, 8, 13 },  {9, 1, 0, 2, 8},  {0, 2, 6, 3, -1} };    // declare and initialize the max and the min   |  | | --- | | //write your code here |     for ( int row=0; row < data.length; row++)  {  for ( int col=0; col < data[row].length; col++)  {   |  | | --- | | //write your code here |   }  }    // write out the results  System.out.println( "max = " + max + "; min = " + min );  }  } |

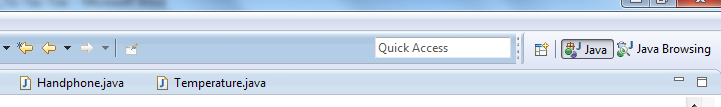
# Tracing and Debugging using Eclipse

Let us revise about tracing which you have learnt before in C105 - Introduction to Programming and C208 - Object Oriented Programming modules. Visit the following link and learn how tracing can be done in Eclipse.

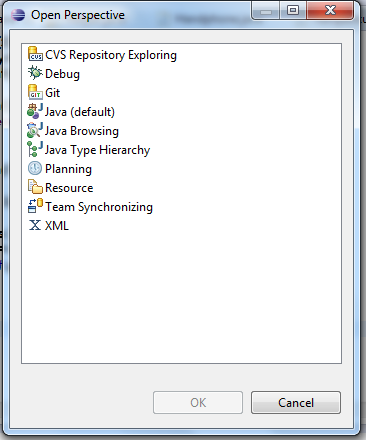
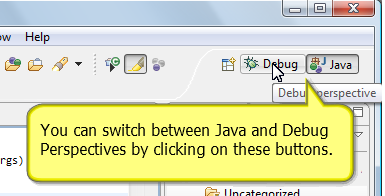
<http://homes.soi.rp.edu.sg/frankie_cha/EclipseC306Tracing/DebuggingInEclipse.htm>



Copy the code into a new Eclipse Project. Trace the code and compare the result with what you have written down earlier.



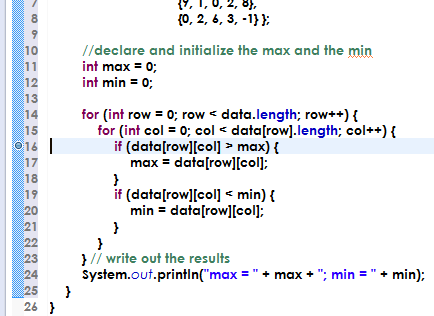
**Alternatively click on open** perspective

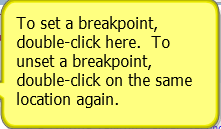
 

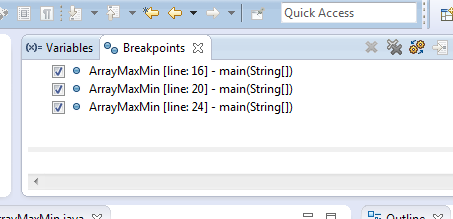
**Select debug perspective**

1. Try to debug the code in the class ArrayMaxMin,
   1. Write a print statement to print the row and col values inside the inner for loop
   2. Place a break statement at this line number 16, 20 and 24.
   3. Did you get the same outcome, in the console screen? Write in your observation of the value of row and col after each step in and step out.

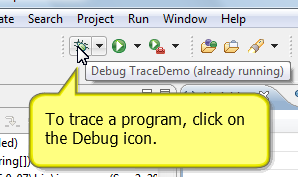
|  |
| --- |
|  |

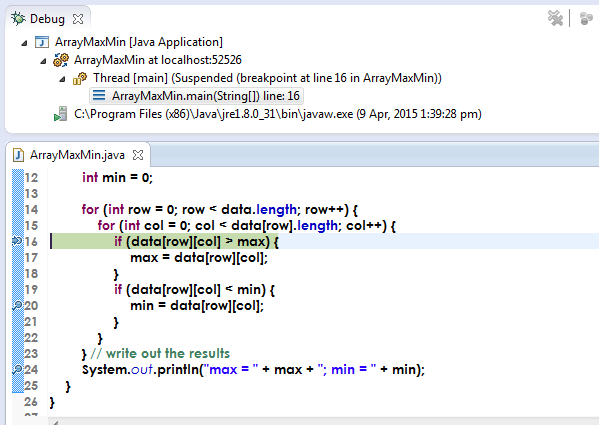
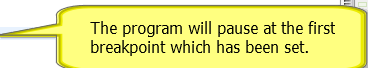


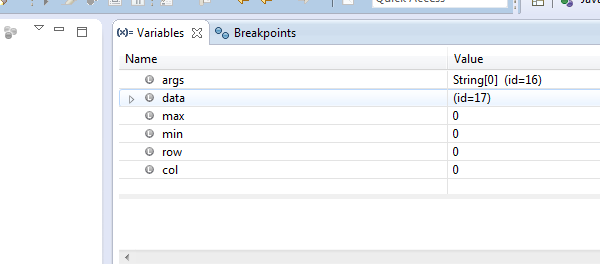
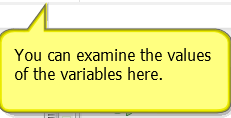




**Break point at line 16, 20 and 24 is set**

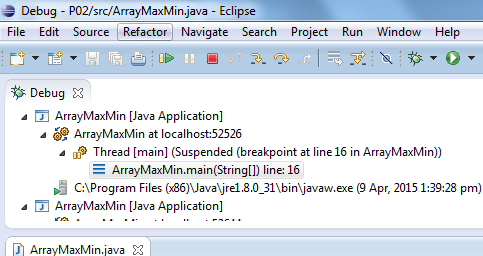
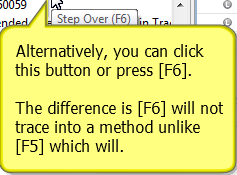


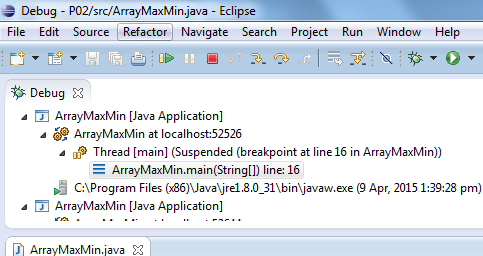
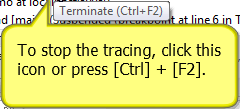


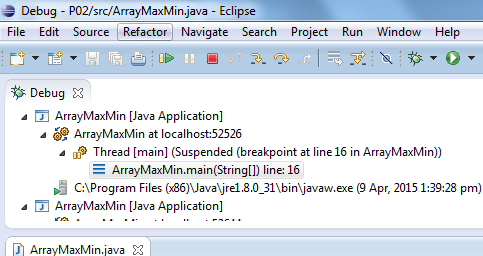
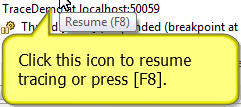


# 

**To trace through the program line by line you can click this button or press F5**







# Printing Patterns

Here are some exercises on nested for loops

Key in the following codes in *Eclipse* and run. You should be able see Pattern **A** printed.

|  |
| --- |
| public class PatternFor {  public static void main(String args[]) {  for (int row = 0; row < 5; row++) {  for (int col = 0; col < 5; col++) {  System.out.print("% ");  }  System.out.println();  }  } }  Number of rows = 5  Number of columns = 5 |

### Pattern A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| % | % | % | % | % |
| % | % | % | % | % |
| % | % | % | % | % |
| % | % | % | % | % |
| % | % | % | % | % |

### Analyse Pattern A

Number of columns = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Col 0** | **Col 1** | **Col 2** | **Col 3** | **Col 4** |
| **Row0** | % | % | % | % | % |
| **Row1** | % | % | % | % | % |
| **Row2** | % | % | % | % | % |
| **Row3** | % | % | % | % | % |
| **Row4**  Number of rows = 5 | % | % | % | % | % |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. Replace % with \*. Modify the above code to display the pattern to display a grid with value \*



1. Modify the above codes to produce the rest of the patterns below.

### Pattern B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| % | % | % | % | % |
| % |  |  |  | % |
| % |  |  |  | % |
| % |  |  |  | % |
| % | % | % | % | % |

### Analyse Pattern B

**Col 0 Col 4**

**Row 0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| % | % | % | % | % |
| % |  |  |  | % |
| % |  |  |  | % |
| %  **Row 4** |  |  |  | % |
| % | % | % | % | % |
|  |  |  |  |  |

Have a closer look at pattern B and you find that only Row 0, Col 0 , Row 4 and Row 5 are printed, rest of them. **Hint –** Inside the for loop, before printing place an if condition :

[ if (row == 0 || row == 4 || col == 0 || col == 4) ]

### Pattern C

### Hint – The maximum number of column items for each row decrements by 1, ( so the max value in the condition for column must be decremented for each row iteration )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| % | % | % | % | % |
| % | % | % | % |  |
| % | % | % |  |  |
| % | % |  |  |  |
| % |  |  |  |  |

### 

### Pattern D

## Hint – The maximum number of column items for each row increments by 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| % |  |  |  |  |
| % | % |  |  |  |
| % | % | % |  |  |
| % | % | % | % |  |
| % | % | % | % | % |

## Challenging Patterns

### Pattern E

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| % |  |  |  |  |
|  | % |  |  |  |
|  |  | % |  |  |
|  |  |  | % |  |
|  |  |  |  | % |

### Pattern F

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| % |  | % |  | % |
|  | % |  | % |  |
| % |  | % |  | % |
|  | % |  | % |  |
| % |  | % |  | % |

## 

## Introducing 2D Arrays

The following code declares and initializes a **one** dimensional array of nine int.

int[] list = new int[9];  
list[0] = 2;  
list[1] = 6;  
list[2] = 8;  
list[3] = 9;  
list[4] = 5;  
list[5] = 7;  
list[6] = 2;  
list[7] = 9;  
list[8] = 3;

Based on the above codes, what is the value of the array element when the index in the array matches in the index in the following table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Values |  |  |  |  |  |  |  |  |  |

The following code declares and initializes a **two** dimensional array of three rows and three columns. This makes a total of nine int.

// rows first, columns second  
int[][] table = new int[3][3];  
table[0][0] = 2;  
table[0][1] = 6;  
table[0][2] = 8;  
table[1][0] = 9;  
table[1][1] = 5;  
table[1][2] = 7;  
table[2][0] = 2;  
table[2][1] = 9;  
table[2][2] = 3;

1. Fill in the values of the following table based on the above code definition.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column 0 | Column 1 | Column 2 |
| Row 0 |  |  |  |
| Row 1 |  |  |  |
| Row 2 |  |  |  |

1. Write down the similarities and differences between 1D and 2D arrays when executing the following operations in the following table.

|  |  |  |
| --- | --- | --- |
| Operations | 1D Array | 2D Array |
| Creating |  |  |
| Getting value |  |  |
| Setting value |  |  |

Assuming the variable table has been re-assigned with the following values:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column 0 | Column 1 | Column 2 |
| Row 0 | 0 | 6 | 4 |
| Row 1 | 7 | 5 | 8 |
| Row 2 | 1 | 9 | 3 |

1. What is the output of the following code?

|  |
| --- |
| if (table[1][2] == 9) {  System.out.println("Apple"); } else {  System.out.println("Pineapple"); } |
|  |

1. What is the output of the following code?

|  |
| --- |
| if (table[0][1] == 6) {  System.out.println("Banana"); } else {  System.out.println("Cherry"); } |
|  |

# Introducing Enums

It is a data type (similar to class) with fixed possible values (Java 1.5 and above). It is used in place of **CONSTANTS** int and String for readability and preventing errors.

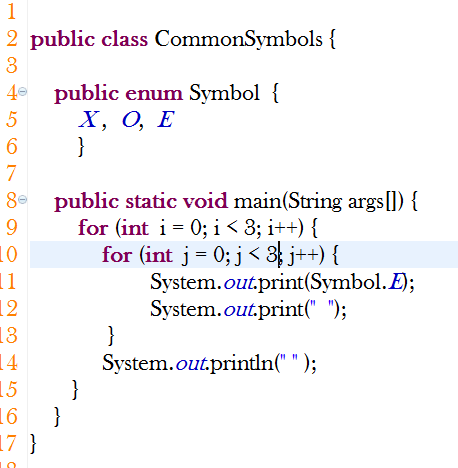
Any location in the Tic Tac Toe grid has *three* possibilities. It is either *unoccupied* or it is occupied by a *cross* or *zero*.

<http://download.oracle.com/javase/tutorial/java/javaOO/enum.html>

1. Create an enum to represent a location in the grid as shown below.

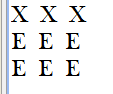
## Thinking in small steps

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | E | E | E |
| 1 | E | E | E |
| 2 | E | E | E |
|  |  |  |  |



1. Create a 2D array of enum to represent the grid with the symbol X
2. Create a 2D array of enum to represent the grid with the symbol O

1. Create a 2D array of enum to represent the symbols E for Empty. Symbol X printed only at row 0 as shown in the pattern below.



# Introducing Algorithms

1. Use an algorithm to check the row number that contains the symbol X
2. List all the cells that you have to check for a *HORIZONTAL* to win in tic-tac-toe:

The first row is done for you already.

* Row 0, Column 0 and Row 0, Column 1 and Row 0, Column 2
* \_\_\_\_\_\_\_
* \_\_\_\_\_\_\_

1. Can you see the pattern to combine the above into a for loop?

|  |
| --- |
|  |

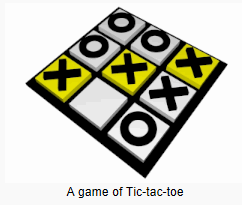
1. List all the cells that you have to check for a *VERTICAL* to win in tic-tac-toe: Write a for loop to check.

|  |
| --- |
|  |

1. List all the cells that you have to check for a *DIAGONAL* to win in tic-tac-toe: Write a for loop to check

# Analysing the Problem

In this section we will analyse the solution to the problem statement. We will first identify the Inputs to the program, process the inputs and then provide the result.

Analyse the Input Data

* There are two players in this game 🡪 **X, O**
* The grid is a 3×3 grid 🡪 board

1. Build the class and Identify the members in the class

**Main class (GameUI class)**

**Game class**

currentPlayer 🡪 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Board class**

tBoard 🡪 \_ 2D arrray\_\_\_\_

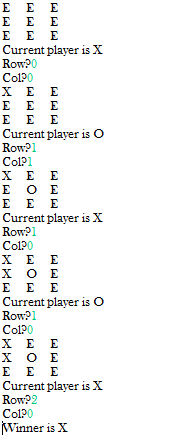
Process the Data

Let’s write a pseudo code for this program and then address one by one.

* Display the 3×3 grid filled with the value “ E “
* Allow player 1 to use the symbol “X” and player 2 to use the symbol “O”.
* Provide the option for the first player to choose the row and column of the space that he intend to mark. After the players chooses, mark the cell chosen with the symbol “X”
* Provide the option for the second player to choose the row and column of the space that he intends to mark. After the players chooses, mark the cell chosen with the symbol “O”
* The players must take turns to choose the option and continue to play the game
* The program must be able to **identify** if any of the horizontal, vertical or diagonal rows are marked completely.
* Once the program determines the row filled, the program must be able to guess the winner of the game based on the symbol forming the completed row.
* The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game.

Display results

The player who wins must be displayed at the end of the game



## Advanced Challenges

* The program must be able to give another choice to the player if

the player chooses the cell which is not empty or already chosen.

* The program must be able to choose the size of the board.

*End of Worksheet*