**LAB 2.1**

*Write java program (Class) in which define some global of different data type (int, short, byte, char, boolean, etc.) and then print their default values.*

**Steps:**

* Create a java source file – VariablePractice.java ; Write public class VariablePractice
* Define variable global variables
* Try to use some reserve keywords as identifier (name of variables)
* Print the default value of these variables.

/\*\*

\* This class is use to various type of practice on

\* variables (Local, global) and constants; and on data types

\*/

**public** **class** VariablePractice1 {

/\*\*

\* please ignore static word here;

\* We will learn about static in next chapter

\* For now just ignore it and use it for this example

\*/

/\* define few global variable and not initializing it

\* of different data type (int, float, char, boolean, etc.)

\*

\* try to use any reserved keyword as identifier (name of variable)

\* and you will see, that compiler will not allow to do so

\*/

**static** **int** *x*;

**static** **boolean** *bool*;

**static** **char** *c*;

**static** **short** *s*;

**static** **byte** *b*;

**static** **double** *d*;

**static** **float** *f*;

**public** **static** **void** main(String[] args) {

/\* print the default value

\* i.e. value of uninitialized variable

\*/

System.*out*.println("default value of int: " + *x*);

System.*out*.println("default value of boolean: " + *bool*);

System.*out*.println("default value of char: " + *c*);

System.*out*.println("default value of short: " + *s*);

System.*out*.println("default value of byte: " + *b*);

System.*out*.println("default value of double: " + *d*);

System.*out*.println("default value of boolean: " + *f*);

}

}

**Learning’s:**

* How to define a global variable
* Use of data type in creating a variable
* Default value of global variables
* Reserve keywords cannot be use as identifier

**LAB 2.2**

*Modify LAB 2.1; now initialize the global variables and print them and then modify them in main method and again print them.*

*Now also create some constants; and try to modify them as well.*

/\*\*

\* This class is use to various type of practice on

\* variables (Local, global) and constants; and on data types

\*/

**public** **class** VariablePractice2 {

/\*\*

\* Please ignore static word here;

\* We will learn about static in next chapter

\* For now just ignore it and use it for this example

\*/

/\* define few global variable and initialize them

\* Also define some constants

\* (here for example I have defined an int type

\* variable and constants)

\*/

**static** **int** *x* = 12;

**static** **final** **int** *y* = 10;

**public** **static** **void** main(String[] args) {

// printing the value x

System.*out*.println("value of int x before modifying: " + *x*);

// modifying its value

*x* = 14;

// to check, if global variable can be modified locally

System.*out*.println("value of int x after modifying: " + *x*);

// try to modify constant, complier will not alloow

// u = 16;

}

}

**Learning’s:**

* How to define a constants
* Global variables can be modify in a method (but will be locally applicable) – will see it in future examples

**LAB 2.3**

*Write java program (Class) in which – define some variable of different type say int, shor, byte, long, etc. data type; now try to cast (assign) the int variable to a short variable, and vice versa; int value to long and vice versa.*

*Basically try various permutation and combination for up& down castings.*

/\*\*

\* This class is use to practice

\* various up and down casting

\*/

**public** **class** DataTypeCastingPractice {

/\*\* Main method

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

// create a int variable

**int** x = 1000;

// try to assign it to a short variable (without casting)

// short s = x; // compiler will not allow it

// now assign the same with casting

**short** s = (**short**)x;

System.*out*.println("int value is : " + x);

System.*out*.println("short value (cast from int) : " + s);

// now assign(cast) the same int value to a byte

**byte** b = (**byte**)x;

System.*out*.println("byte value (cast from int) : " + b);

/\*\* Learnings - down casting must be explicitly cast

\* value may lost in case of down casting

\*/

// now assign (cast) the int into a long variable

**long** l = x; // no explicit cast is required

/\*\* Learnings -

\* up casting is implicit

\*/

//**TODO** - do similar exercise for other data type

// (flaot, double, char, etc) as well

}

}

**Learning’s:**

* Down casting must be explicitly cast; but value may lost in case of down casting
* Up casting is implicit
* Casting leader – byte 🡪 short 🡪 char 🡪 int 🡪 long 🡪 float 🡪 double

**LAB 2.4**

*Write java program (Class) in which – define some numeric variables (int, short, etc.) perform some arithmetic, shift and unary operations. Check the precedence’s.*

**Steps:**

* Create a java source file – OperatorPractice.java ; Write public class OperatorPractice
* Define an int variables, practice some unary operation s
* Now, practice some arithmetic operation, with aim to check operator precedence
* Now, uses parenthesis in above arithmetic expression and check result
* Now, define another numeric variable and practice some shift operation…. And so on…

/\*\*

\* This class is use to practice various operation

\* like arithmetic, shift, unary, etc. in Java

\*/

**public** **class** OperatorPractice {

/\*\* Main Method (starting point of program)

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

// PART – A // Define a int variable x with initial value 10

**int** x = 10;

// practice unary operations x++ and ++x

System.*out*.println("x++ unary opr : " + (x++));

System.*out*.println("current value of x :" + x);

System.*out*.println("++x unary opr : " + (++x));

// PART - B // write an arithmetic expression, check result

**double** result1 = 10 + 8 - 6 \* 5 % 4 / 3;

System.*out*.println("result of 10 + 8 - 6 \* 5 % 4 / 3 :" + result1);

// PART - C

// use some parenthesis on the arithmetic expression and check result

**double** result2 = 10 + (8 - 6) \* 5 % 4 / 3;

System.*out*.println("resulr of 10 + (8 - 6) \* 5 % 4 / 3 :" + result2);

/\*\* Learning - precedence of operators \*/

// PART - D // define a numeric variable and practice shift operation

**int** y = 15;

y = y>>2;

System.*out*.println("value of y (15) after y>>2 : " + y);

// PART - E, F, G..... // Do some more similar practice

}

}

**Learning’s:**

* Various operations and operator precedence’s

**LAB 2.5**

*Write java program (Class) which take command line arguments and perform some mathematical operation on them.*

**Steps:**

* Create a java public class MathsOnUserArguments
* Pass 2 arguments (numbers) while running (use use as 🡪 run configuration of STS)
* Receive it as String, parse as number and perform arithmetic operations

/\*\*

\* In this class, we will accept

\* command line arguments and perform

\* mathematical operation on them

\*/

**public** **class** MathsOnUserArguments {

/\*\* Main methods

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

/\*\* Please ignore - what is String, args[, Integer and parseInt

\* just use it for now,

\* we will learn about String in Java.LANG session,

\* Integer and parseInt in WrapperClass session

\* and args[] in session on ARRAY

\*

\* For now just understand that,

\* String args[] is use to take command line arguments,

\* and parseInt to convert it into number

\*/

String firstArg = args[0];

String seconfArg = args[1];

**int** firstNo = Integer.*parseInt*(firstArg);

**int** secondNo = Integer.*parseInt*(seconfArg);

// sum of the two numbers

**int** result = firstNo + secondNo;

System.*out*.println("Sum of "+firstNo+" and "+secondNo+" is: "+result);

}

}

**Learning’s:**

* How to use numbers (argument) form command line (run time argument)

**LAB 2.6**

*Write java program (Class) which take 3 command line arguments (numbers) and perform some mathematical operation on last two numbers, first number will be use to tell what operation user want to perform (1 for (+), 2 for (-), 3 for (\*), 4 for (/) and 5 for (%))*

**Steps:**

* Create a java public class MathsOnUserArguments1
* Pass 3 arguments (numbers) while running (use use as 🡪 run configuration of STS)
* First argument must be any number from 1 to 5 as 1 for (+), 2 for (-), 3 for (\*), 4 for (/), 5 for (%)
* Perform operation on last two numbers and decide the operation on first argument using if-else

/\*\*

\* This class is use to take three argument from user

\* first as operation type and other two as number

\*

\* Aim of this assignment is to learn how to use

\* if-else-if block

\*/

**public** **class** MathsOnUserArguments1 {

**public** **static** **void** main(String[] args) {

/\*\* Please ignore - what is String, args[, Integer and parseInt

\* just use it for now,

\* we will learn about String in Java.LANG session,

\* Integer and parseInt in WrapperClass session

\* and args[] in session on ARRAY

\*

\* For now just understand that,

\* String args[] is use to take command line arguments,

\* and parseInt to convert it into number

\*/

// Receive three arguments; considering first as oprtation choice

String oprArg = args[0];

String firstArg = args[1];

String seconfArg = args[2];

// parsing them

**int** oprType = Integer.*parseInt*(oprArg);

**int** firstNo = Integer.*parseInt*(firstArg);

**int** secondNo = Integer.*parseInt*(seconfArg);

// create two local variable

**int** result = 0;

**boolean** isWrongChoice = **false**;

**if** (oprType == 1){

result = firstNo + secondNo;

}**else** **if** (oprType == 2){

result = firstNo - secondNo;

}**else** **if** (oprType == 3){

result = firstNo \* secondNo;

}**else** **if** (oprType == 4){

result = firstNo / secondNo;

}**else** **if** (oprType == 5){

result = firstNo % secondNo;

}**else** {

// in case first argument is beyond 1-5

isWrongChoice = **true**;

}

**if** (!isWrongChoice){

System.*out*.println("Result is : " + result);

} **else** {

System.*out*.println("Invalid choice!!! please enter 1-5 as choice.");

}

}

}

**Learning’s:**

* How to use if-else-if block

**LAB 2.7**

*Rewrite LAB 2.6; this time in place of if-else-if use switch block.*

Now, again use LAB 2.7, this time just remove break statement from switch block.

**Learning’s:**

* How to use switch block
* Important of break statement
* Valid argument (in terms of data type) for switch condition

**LAB 2.8**

*Write table of 3 with the help of FOR loop (display every number in a new line)*

/\*\*

\* This class is use to practice FOR loop

\*/

**public** **class** LoopPractice1 {

/\*\* Main Method (starting point)

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

**int** x = 3;

System.*out*.println("Table of : " + x);

**for**(**int** i = 1; i < 11; i++){

System.*out*.println(x + " \* " + i + " : " + x \* i);

}

}

}

You can write the same program (LAB 2.8) for table of any number provided by user as run time argument.

**Learning’s:**

* How to use for block

**LAB 2.9**

*Write table of 3 with the help of for loop; but every third no. (Which is 9, 18, 27 in table of 3), should not be visible.*

/\*\*

\* This class is use to practice FOR loop

\*/

**public** **class** LoopPractice2 {

/\*\* Main Method (starting point)

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

**int** x = 3;

System.*out*.println("Table of : " + x);

**for**(**int** i = 1; i < 11; i++){

**if** (i % 3 != 0)

System.*out*.println(x + " \* " + i + " : " + x \* i);

}

}

}

**Learning’s:**

* Logical use of if block (this is just to give a feel of benefit of flow control mechanism)

**LAB 2.10**

*Write table of 3 with the help of for loop (display every number in a new line); on every third no. (which is 9, 18, 27 in table of 3),this no should not be visible rather there will be table of 10, 20, 30 (display in single line) respectively*

/\*\*

\* This class is use to practice FOR loop

\*/

**public** **class** LoopPractice3 {

/\*\* Main Method (starting point)

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

// since we have to print table of 3

**int** x = 3; // x can be change with run time argument

System.*out*.println("Table of : " + x);

// this variable will be use to print table of 10, 20, 30 etc.

**int** multipleOfThreecount = 0;

**for**(**int** i = 1; i < 11; i++){

**if** (i % 3 != 0){

System.*out*.println(x + " \* " + i + " : " + x \* i);

} **else** {

// first time number will be 10, next time 20 and so on

multipleOfThreecount++;

**int** number = multipleOfThreecount \* 10;

**for** (**int** j = 1; j < 11; j ++){

System.*out*.print(j \* number + ", ");

}

System.*out*.println(" "); // just to give a line break

}

}

}

}

**Learning’s:**

* Use of nested loop

**LAB 2.11**

Rewrite 2.8, 2.9 and 2.10 with help of while and do-while loop

**LAB 2.12**

*Write java program (Class), in which, we will create an array initialize, populate and use it*

/\*\*

\* This class is use to practice Array

\*/

**public** **class** ArrayPractice {

**public** **static** **void** main(String[] args) {

// creates an int array of size 10

**int** [] intArray = **new** **int**[10];

// populate it with value of table of 2

**for** (**int** i=1; i<11; i++){

intArray[i-1] = i \* 2;

}

// print intArray // you will see it will print an object ref.

System.*out*.println(intArray);

//to print value of intArray

**int** j = 0;

**for** (;;){

j++;

**if** (j == 10)

**break**;

System.*out*.println(intArray[j]);

}

}

}

**LAB 2.13**

*Now modify the LAB 2.12, try to enter more than 10 elements in intArray; or try to run the loop which prints value of array for more the 10 times.*

*Just change line - if (j ==* ***10****) to if (j ==* ***11****)*

*OR change line -* **for** (**int** i=1; **i<11**; i++) to **for** (**int** i=1; i<12; i++)

*Also try to enter any non int value in intArray.*

*You will fine array out of bound exception – this is to understand the limitation of array as*

* *Size of array is fixed*
* *Only homogenous element can be stored in an array*

**Learning’s:**

* Understand how to create, initialize, populate, use an array and how to retrieve value of an array.
* Limitations of array

**LAB 2.14**

*Write java program (Class), in which, create a two dimensional array, initialize it, and populate with value of table of 2*

/\*\*

\* This class is use to practice Array

\*/

**public** **class** ArrayMultiDimensionalPractice {

**public** **static** **void** main(String[] args) {

// creates an int array of size 10

**int** [][] intArray = **new** **int**[5][2];

// populate it with value of table of 2

**int** count = 0;

**for** (**int** i=0; i<5; i++){

**for** (**int** j = 0; j<2; j++){

count++;

intArray[i][j] = count \* 2;

}

}

**for** (**int** i=0; i<5; i++){

**for** (**int** j = 0; j<2; j++){

System.*out*.print(intArray[i][j] + " ");

}

System.*out*.println(" ");

}

}

}

**OUTPUT**

2 4

6 8

10 12

14 16

18 20

**Learning’s:**

* Understand how to create, initialize, populate, and use a multidimensional array