**LAB 3.1**

*Create two java classes, now in class1 (which will have main method) create object of the class2 (which will have a method called display () - which displays a message on console). Now call the display method of Class2.*

**Steps:**

* Create two Java classes Cls1 and Cls2
* Cls1 will have main method (public static void main(String args[]),
* In Cls2 write a method “public void display() “ which will display a message on console
* Create obj of Cls2 in main method of Cls1, and call the display method. Run it

**CLASS 2**

/\*\* This class have a method "display" which displays a message on console

\*/

**public** **class** Cls2 {

/\*\* ignore what is public, void here that we will learn

\* when we will see the method signature in details

\*/

**public** **void** display(){

System.*out*.println("Hi.... You are in display method of cls2");

}

}

**CLASS 1**

/\*\* This is main class, we will create object of cls2 and all its methods

\*/

**public** **class** Cls1 {

/\*\* Main method - stating point of program \*/

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

// instantiating object of Cls2

Cls2 cls2Obj = **new** Cls2();

// calling display method of Cls2

cls2Obj.display();

}

}

**OUTPUT**

Hi.... You are in display method of cls2

**Learning:**

* How to create (instantiate) an object
* Object have all the method, variables of the class
* How to call a method of a class using its object

**LAB 3.2**

*Now override the default constructor of class2 (cls2), display some message within constructor. This exercise is to understand default constructor.*

**MODIFY CLASS 2 (Cls2)**

Add a constructor

/\*\* Modifying the Default constructor \*/

**public** Cls2() {

System.*out*.println("Inside dafault constructor of Cls2");

}

**RUN CLASS 1 (Cls1)**

**OUTOUT**

Inside default constructor of Cls2

Hi.... You are in display method of cls2

**Learning:**

* Understanding of default constructor
* Constructor can be use to write code – if there is requirement of running a piece of code before instantiation of an object

**LAB 3.3**

*Now add a parameterized constructor of Cls2, and instantiate another object of Cls2 in cls1 using this.*

**IN CLASS 2 (Cls2)**

/\*\* One more constructor of cls2 \*/

**public** Cls2(String name){

System.*out*.println("Hi " + name

+ " you are Inside parametrized constructor of Cls2");

}

**IN CLASS 1 (Cls1) -** Instantiate another object using next constructor

// instantiating object of Cls2 using other constructor

Cls2 cls2Obj2 = **new** Cls2("MyName");

**OUTPUT**

Inside dafault constructor of Cls2

Hi MyName you are Inside parametrized constructor of Cls2

Hi.... You are in display mehtod of cls2

*Now remove the default constructor of cls2, and try to run the code.*

**Learning:**

* There can be ‘n’ numbers of constructor in a class
* If there are any constructor explicitly written in a class, there is nothing like default constructor

**LAB 3.4**

*Create two java classes -, first, Mainclass (which will have main method) and second, MyMath Class having methods like add, subtract, multiply, divide (each methods will have two parameters, that is will perform respective mathematical operation between two numbers). Now call the various methods of MyMath class from MainClass and pass the two numbers, receive results and print them.*

**Steps:**

* Create two Java classes MainClass and MyMath
* MainClass will have main method (public static void main(String args[]),
* In MyMath write all required methods with appropriate return type and paramteres
* Create obj of MyMath in main method of MainClass, and call various methods. Run it

/\*\*

\* This class performs various mathematical operations

\* like addition, substract, multiple, divide etc

\*/

**public** **class** MyMath {

/\*\*

\* **@param** firstNumber

\* **@param** secondNumber

\* **@return** double (addition of two numbers)

\*/

**public** **double** add(**double** firstNumber, **double** secondNumber){

**return** firstNumber + secondNumber;

}

**public** **double** substract(**double** firstNumber, **double** secondNumber){

**return** firstNumber - secondNumber;

}

**public** **double** multiple(**double** firstNumber, **double** secondNumber){

**return** firstNumber \* secondNumber;

}

**public** **double** divide(**double** firstNumber, **double** secondNumber){

**return** firstNumber / secondNumber;

}

**public** **double** modulus(**double** firstNumber, **double** secondNumber){

**return** firstNumber % secondNumber;

}

}

**public** **class** PerformingMathOperation {

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

MyMath myMath = **new** MyMath();

**double** firstNo = 23.5;

**double** secNo = 12;

**double** result = myMath.add(firstNo, secNo);

System.*out*.println("Sum of "

+ firstNo + " and "+ secNo + " : " + result);

}

}

**OUTPUT**

Sum of 23.5 and 12.0 : 35.5

Do similar operation for othe methods as well

**Learning:**

* How to pass argument while calling a method
* What is return type of a method; how to receive it while calling this method

**LAB 3.5**

*In Lab 3.4; receive the two numbers for user as runtime argument, now use these numbers.*

**Steps:**

* Just change the 2 line from Lab 3.4 of mainClass (instread of directly assigning any value to 2 double varibles, take the values from command line arguments and parse them

**In LAB 3.4**

**double** firstNo = 23.5;

**double** secNo = 12;

CHANGE IT IN LAB 3.5 AS

**double** firstNo = Double.*parseDouble*(args[0]);

**double** secNo = Double.*parseDouble*(args[1]);

Now while running, pass the two numbers as commond line argument

**LAB 3.6**

*Further improvement (variance in Lab 3.5) receive the two numbers and type of operation from user as runtime argument (command line) and use either if-else-if or switch block to call appropriate method of MyMath class.*

**Steps:**

* There will be no change in MyMath class
* In Mainclass, now the choice of operation will be proivded by user as common line argument
* Lets consider that user is passing first argument as operation choice (1 for add, 2 for substraction, 3 for multiply, 4 for divide and 5 for modulus)

**public** **class** PerformingMathOperation {

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

MyMath myMath = **new** MyMath();

/\* First command line argument - to select type of operations

\* 1 - add, 2 - substract, 3 - multiply, 4 - divide, 5 - modules

\*/

**int** oprType = Integer.*parseInt*(args[0]);

// other two arguments are the numbers,

// on which operation will be performed

**double** firstNo = Double.*parseDouble*(args[1]);

**double** secNo = Double.*parseDouble*(args[2]);

**double** result = 0.0;

**boolean** isValidOpration = **true**;

String operation = "";

**switch** (oprType){

**case** 1:

result = myMath.add(firstNo, secNo);

operation = "Addition ";

**break**;

**case** 2:

result = myMath.substract(firstNo, secNo);

operation = "Substration ";

**break**;

**case** 3:

result = myMath.multiple(firstNo, secNo);

operation = "Multiplication ";

**break**;

**case** 4:

result = myMath.divide(firstNo, secNo);

operation = "Division ";

**break**;

**case** 5:

result = myMath.add(firstNo, secNo);

**break**;

**default**:

operation = "Invalid Operation";

isValidOpration = **false**;

**break**;

}

**if** (isValidOpration){

System.*out*.println(operation + "of "

+ firstNo + " and "+ secNo + " : " + result);

} **else** {

System.*out*.println(operation);

}

}

}

**LAB 3.7**

*A class will have a global variable (say int x); now create the two objects of this class in main class (having main method), now change the variable value in both the object (value should be diff, say object1.x = 10 and object2.x = 12), print this value.*

*Now make the global variable static and run the same program. Analyze the different in result.*

**Steps:**

* Create a class (say Practice) which have two global varibale (x and y); one non-static and other static .
* In main method of some other class; create 2 objects of Practice class,
* Set different value for non static variable(x) in both the objetcs
* Now set different value for static varible (y) in both the objects
* Check the value of x & y (non static and static variable) in respectice objects

**public** **class** StaticVariablePractice {

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

// craete 2 objects of practice class

Practice obj1 = **new** Practice();

Practice obj2 = **new** Practice();

// set different value of x (non static variable) for obj1 & obj2

obj1.x = 12;

obj2.x = 14;

// set different value of y (static variable) for obj1 & obj2

obj1.*y* = 23;

obj2.*y* = 28;

System.*out*.println("Value of X & Y in obj1 is : "

+ obj1.x + " and " + obj1.*y*);

System.*out*.println("Value of X & Y in obj2 is : "

+ obj2.x + " and " + obj2.*y*);

}

}

**class** Practice{

**int** x;

**static** **int** *y*;

}

**OUTPUT**

Value of X & Y in obj1 is : 12 and 28

Value of X & Y in obj2 is : 14 and 28

**Learning:**

* Understand difference between static and non-static variables
* How static variable belong to class, so any change in it will not limited to a particulat object but all the objects of that class

**LAB 3.8**

*This is extension of Lab 3.7; now write two methods in class, one static and one non static, now create object of this class in main method, call this methods; call the method without using object instance (directly from name of class).*

**Steps:**

* Write A class having two methods; one static and other non-static
* Call the static method from some other class, wthout using object of practice

**public** **class** StaticMethodPractice {

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

// instantiate objects of practice class

Practice2 obj1 = **new** Practice2();

//calling non-static method of Practice class

obj1.display();

//calling static method of Practice class, using object

obj1.*displayStatic*();

//calling non-static method of Practice class, without object

Practice2.*displayStatic*();

}

}

**class** Practice2{

**static** **void** displayStatic(){

System.*out*.println("Inside a static mathod of practice class");

}

**void** display(){

System.*out*.println("Inside a non - static mathod of practice class");

}

}

**Learning:**

* Understanding that static method (members) of a class can be accessed without object instance of that class.

**LAB 3.9**

*Write a java class having main method; now define some non static and some static global variable and try to access them in main method.*

**Steps:**

* Write a java class
* Define some static and non static variable
* Write a static method and a non static method
* Try to call access non static varible from static method (without using object of the class)
* Try to access static varible from

/\*\*

\* This class is use to practice that

\* a static method cab only access static variable

\* while non static method cab access both

\* static and non static variable of class

\*/

**public** **class** StaticVariableAccessPractice {

**static** **int** *iStatic* = 10;

**int** iNonStatic = 15;

**public** **static** **void** staticFirstMehtod(){

// accessing static variable from static method

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

// accessing non-static variable from static method

System.*out*.println(iNonStatic); // COMPILATION ERROR

}

**public** **void** nonStaticSecondMehtod(){

// accessing static variable from non-static method

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

// accessing non-static variable from non-static method

System.*out*.println(iNonStatic);

}

}

**Learning:**

* Static method cannot access non-static members.

**LAB 3.10**

*At present all methods of MyMath class (Add, Subtract, Multiply, Divide) have two parameters (i.e. for two numbers only). Now the business requirement is to have all these method for 3 parameters as well. Overload all methods for 3 parameters. (Do not modify old methods, add new methods with same name and return type with different number of parameters)*

**Steps:**

* Overload all methods of MyMath Class

/\*\*

\* This class performs various mathematical operations

\* like addition, substract, multiple, divide etc

\* Now the class is overloaded methods

\* as per different numbers of arguments

\*/

**public** **class** MyMath {

/\*\*

\* **@param** firstNumber

\* **@param** secondNumber

\* **@return** double (addition of two numbers)

\*/

**public** **double** add(**double** firstNumber, **double** secondNumber){

**return** firstNumber + secondNumber;

}

/\*\*

\* Overloaded add method

\* **@param** firstNumber

\* **@param** secondNumber

\* **@param** thirdNumber

\* **@return** double - sum of three numbers

\*/

**public** **double** add(**double** firstNumber, **double** secondNumber, **double** thirdNumber){

**return** firstNumber + secondNumber + thirdNumber;

}

}

Above code is displaying overlaod add method. Here we have two add method, but number of parameter in 2 and 3 respecticely in two methos.

Do the similar for other methods as well

*Additionally, check if overloading is applicable*

* *In case of same number of parameters but for different data type.*
* *In case of different return type but same parameters*

**Learning:**

* What is method overloading, how and why

**LAB 3.11**

*Write a java class - MySubMath which extends the MyMath class; Create an instance (object) of MySubMath class in MainClass (class having main method), and check if all methods of MyMath is available with the instance of MySubMethod.*

**Steps:**

* Create MySubMath Class by extending MyMath class
* Do not write any method in MySubMath class
* Now instantiate an object of MySubMath in some other class and add two numbers

/\*\*

\* MySubMath is sub class of MyMath class

\*/

**public** **class** MySubMath **extends** MyMath {

}

**public** **class** MainClassForMySubMath {

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

MySubMath subMath = **new** MySubMath();

// though MySubMath do not have its own method,

// it has inherited all method of MyMath Class

subMath.add(12, 13);

}

}

**Learning:**

* How to create a sub class
* A subclass by default inherits all members of super class.

**LAB 3.12**

*Override all/some methods of MyMath in MySubMath (like add 5 in the result and then return it, this is just to distinguish the method operations in Super and Sub (overridden) class. Create instance of both the classes in MainClass (in main method) and call the same methods of both the class with same argument values. (Example – call add method from both MyMath and overridden add method from MySubMath class). Check the result.*

**Steps:**

* Override Add method of MyMath class in MySubMath class
* Modify the method by addiing 5; this is just to ensure that output of add methods of MyMath and MySubMath classo should be different)
* Create object of MyMath and MySubMath class in some other class
* Call the add method from both the objetcs and pass same number

/\*\* MySubMath is sub class of MyMath class \*/

**public** **class** MySubMath **extends** MyMath {

@Override

**public** **double** add(**double** firstNumber, **double** secondNumber) {

**return** firstNumber + secondNumber + 5;

}

}

/\*\* This class is use to call/perform various operation of MySubMath Class \*/

**public** **class** MainClassForMySubMath {

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

MyMath math = **new** MyMath();

**double** mathResult = math.add(12, 13);

MySubMath subMath = **new** MySubMath();

**double** subMathResult = subMath.add(12, 13);

System.*out*.println("Add of 12 & 13 in MyMath class " + mathResult);

System.*out*.println("Add of 12 & 13 in MySubMath class " + subMathResult);

}

}

**OUTPUT**

Add of 12 & 13 in MyMath class 25.0

Add of 12 & 13 in MySubMath class 30.0

**Learning:**

* Understanding of overriding (what, how and why)
* Overridden method is applicable for sub class only; for objetc of super class, method will not be mofidiy

**LAB 3.13**

*Write a final class; and try to extends this class*

**Steps:**

* Write a final class (by metionaling final modifer in class signature)

**final** **class** ExampleFinalClass{

**void** display(){

System.*out*.println("in display of practice3 class");

}

}

**class** ExtendingFinalClass **extends** ExampleFinalClass{ // COMPILATION ERROR

}

**Learning:**

* What is final class
* How to create a final class
* Final classes cannot be extends (thus modified or override)

**LAB 3.14**

*Write a normal public class, but write a final method inside I; Now try to override this final method*

**Steps:**

* Write a class (say ExampleFinalClass) having a final method
* Write another class (say ExtendingFinalClass) which extends first class (ExampleFinalClass)
* Try to override the final method

**class** ExampleFinalClass{

**final** **void** display(){

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

}

}

**class** ExtendingFinalClass **extends** ExampleFinalClass{

@Override

**void** display() { // COMPILATION ERROR

System.*out*.println("Overridden display method");

}

}

**Learning:**

* Final method of a class cannot be override

**LAB 3.15**

*Write a program; where class1 will call 2 methods of class2, first method will take primitive data as argument and second will take object of class one. Now change the primitive data value in method 1 for class 2 and check if this value is getting changed in class1; check the similar situation for method 2.*

*Aim of this program is to understand call by value and call by ref*

**Steps:**

* Write a class (say PracticeClass) having 2 methods, one take primitive data type and other object of another class as argument.
* Write mainClass, call the two methods
* Check if primitive data value and object’s variable value will modify or not

/\*\*

\* This class demonstrate the difference of

\* Call By Value and Call by Ref

\*/

**public** **class** CallByRefnValuePractice {

**int** x = 10;

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

CallByRefnValuePractice object = **new** CallByRefnValuePractice();

object.callByValueExample();

object.callByRefExample();

}

/\*\* This method calls a method of another class, which

\* multiply the passed(argument) value with 2

\* Here we are passing a primitive data value

\*/

**private** **void** callByValueExample(){

PracticeClass pracClass = **new** PracticeClass();

pracClass.multiplyBy2(x);

System.*out*.println(x);

}

/\*\* This method calls a method of another class, which

\* multiply the passed(argument) value with 2

\* Here we are passing Object of current class

\*/

**private** **void** callByRefExample(){

CallByRefnValuePractice obj = **new** CallByRefnValuePractice();

PracticeClass pracClass = **new** PracticeClass();

pracClass.multiplyBy2(obj);

System.*out*.println(obj.x);

}

}

**class** PracticeClass{

**public** **int** multiplyBy2(**int** a){

a = a \* 2;

**return** a \* 2;

}

**public** **int** multiplyBy2(CallByRefnValuePractice obj){

obj.x = obj.x \* 2;

**return** obj.x \* 2;

}

}

**OUTPUT**

10

20

**Learning:**

* Understand difference between call of ref and call by value

**LAB 3.16**

*Write a program; where class Outer will have a inner class; both outer and inner class will have a display ( ) method, which display a message.*

*Now create object of inner class and call its display method from some other class.*

**Steps:**

* Create a Outer class
* Write a display method, which display a message
* Create a Inner class inside outer class
* Write a display method inside inner class as well
* Cretae MainClass , which will have publis static void main method
* Now create objetc of Inner class and call its display method

/\*\* This class will have a Inner class \*/

**public** **class** Outer {

//Display method of Outer Class

**public** **void** display (){

System.*out*.println("inside display method of OUTER class");

}

**class** Inner{

//Display method of Inner Class

**void** display (){

System.*out*.println("inside display method of INNER class");

}

}

}

/\*\* This class is use to run the pragram as this class has p.s.v. main method

\*/

**public** **class** MainClass {

/\*\* Main method, starting point of program

\*/

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

// Instantiating object of inner class

Outer.Inner innerObj = **new** Outer().**new** Inner();

innerObj.display();

}

}

**OUTPUT**

inside display method of INNER class

**Learning:**

* How to write a inner class
* How to create instance of an inner class

**LAB 3.17**

*Modift Lab 3.15, add a static inner class, which will have a display () method as well. Now call this method from some other class.*

**Steps:**

* Modify the Outer class – change the class Inner to **static** class Inner (basically, only add static modifier to Inner class)
* Now instatinte the object of inner class and call its display method

Outer.Inner inner = **new** Outer.Inner();

inner.display();

**Learning:**

* Difference between static inner class (nested class) and inner class
* How to create instance of an static inner class or nested class