**DSA-Lab-Manual (Solved)**

(All task & Assignments)

**Lab: 01**

**Lab Tasks 1:**

**Develop Java Class IU\_Mark\_Sheet with following Constructor and Methods.**

**Constructor:**

* **IU\_Mark\_Sheet(String Student, String Registration Number);**

**Methods:**

* **Void Subject\_Name(String Subject[]);**
* **Void Subject\_Max\_Mark(double MaxMark[]);**
* **Void Subject\_Scored\_Mark(double ScoredMark[]);**
* **Double StudentGPA();**

**Code:**

**main.java:**

package Task;

import java.util.Scanner;

public class main {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter Student Name: ");

String Name = scan.nextLine();

System.out.println("Enter Student Registration Number : ");

String RegNo = scan.nextLine();

System.out.println("Enter Subject Name : ");

String SubName = scan.nextLine();

System.out.println("Enter Max Marks : ");

double MaxMarks = scan.nextDouble();

System.out.println("Enter Obtained Marks : ");

double ObtMarks = scan.nextDouble();

IU\_Mark\_Sheet MS = new IU\_Mark\_Sheet(Name, RegNo);

MS.Subject\_Name(SubName);

MS.Subject\_Max\_Mark(MaxMarks);

MS.Subject\_Scored\_Mark(ObtMarks);

MS.StudentGPA();

MS.IU\_Marks\_Sheet();

}

}

**IU\_Mark\_Sheet.java:**

package Task;

public class IU\_Mark\_Sheet {

String Student;

String RegistrationNumber;

String Subject;

double MaxMark;

double ScoredMark;

double gpa;

IU\_Mark\_Sheet(String Student, String RegistrationNumber) {

this.Student= Student;

this.RegistrationNumber = RegistrationNumber;

}

void Subject\_Name(String Subject) {

this.Subject=Subject;

}

void Subject\_Max\_Mark(double MaxMark) {

this.MaxMark=MaxMark;

}

void Subject\_Scored\_Mark(double ScoredMark) {

this.ScoredMark=ScoredMark;

}

double StudentGPA() {

double percentage;

percentage = (ScoredMark/MaxMark)\*100;

if(percentage>=88)

gpa = 4.0;

else if(percentage<=87 && percentage>=81)

gpa = 3.5;

else if(percentage<=80 && percentage>=74)

gpa = 3.0;

else if(percentage<=73 && percentage>=67)

gpa = 2.5;

else if(percentage<=66 && percentage>=60)

gpa = 2.0;

else

gpa = 0;

return gpa;

}

void IU\_Marks\_Sheet() {

System.out.println("\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n");

System.out.println("\t\t IQRA UNIVERSITY MARK SHEET\n\t\t ");

System.out.println("\nNAME : " + Student + "\nRegistration Number : " + RegistrationNumber);

System.out.println("\n\n\nSUBJECT\t\tMAX MARKS \tOBTAINED MARKS \tGPA\n");

System.out.println(Subject + "\t\t" + MaxMark + "\t\t" + ScoredMark + "\t\t" + gpa);

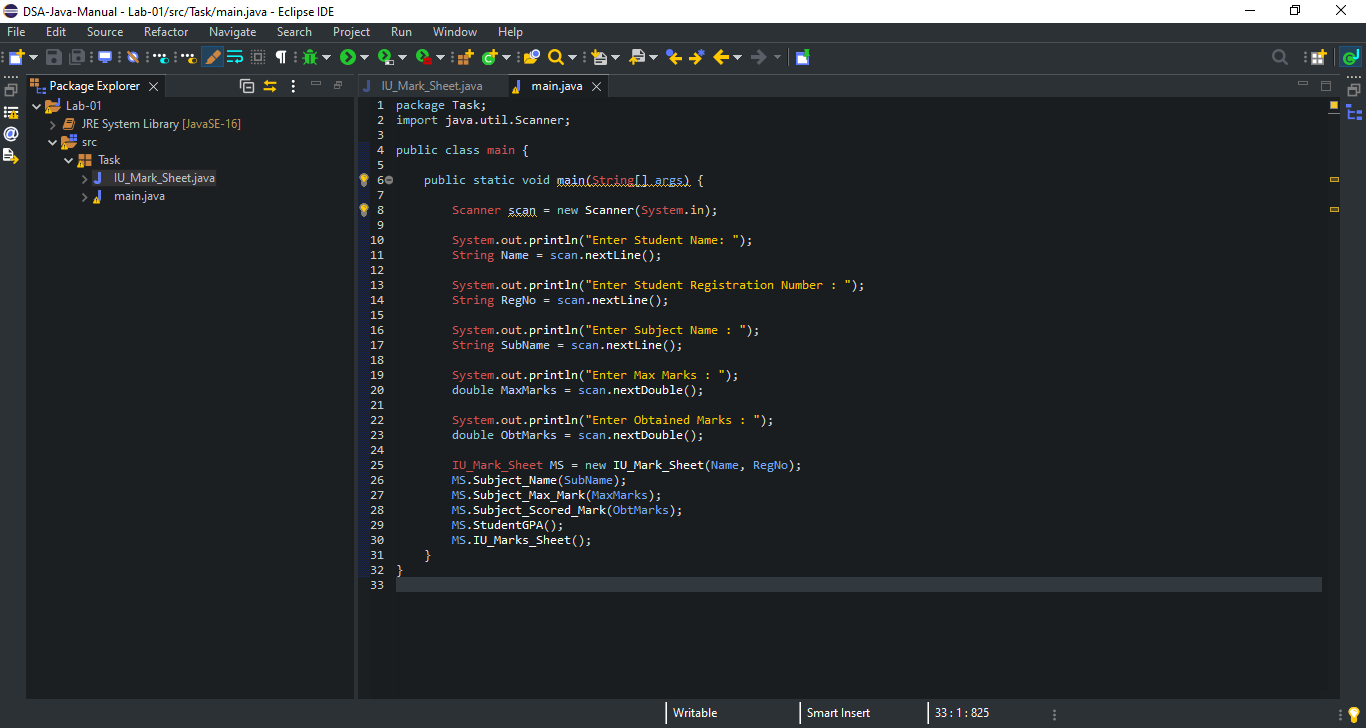
System.out.println("\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n");

}

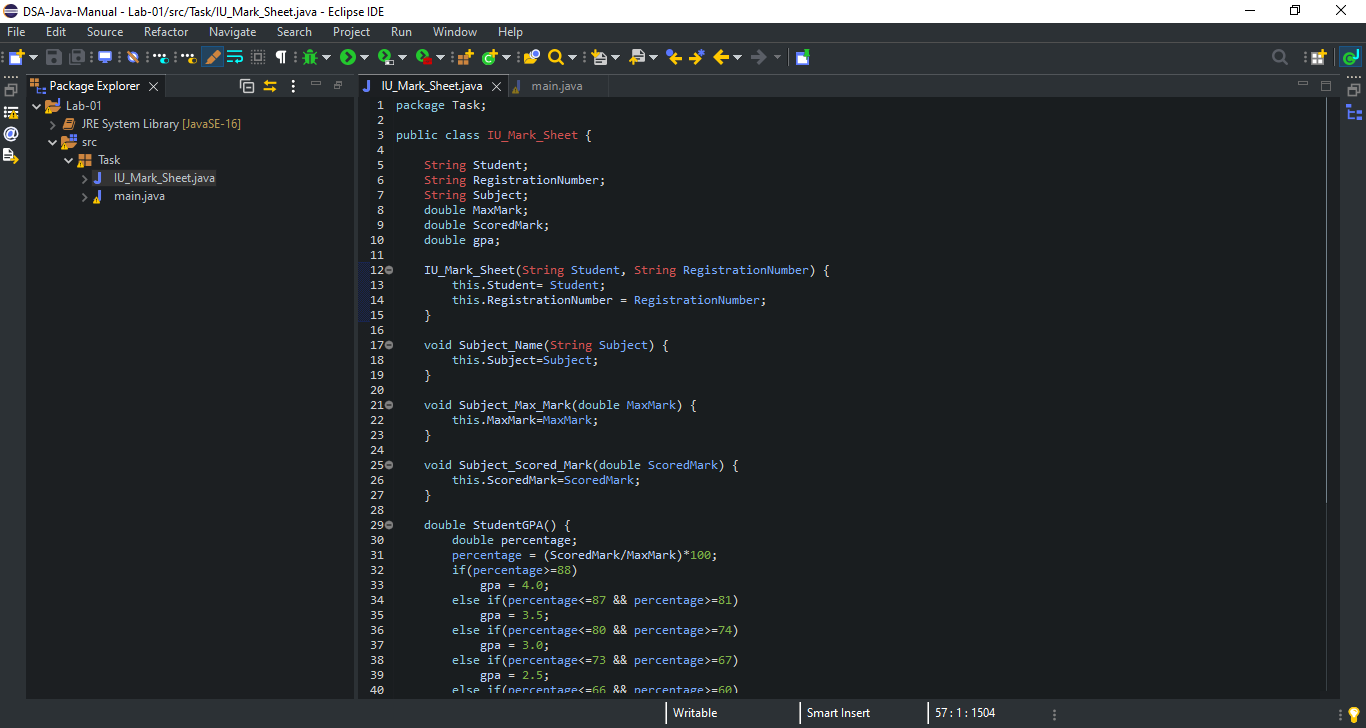
}

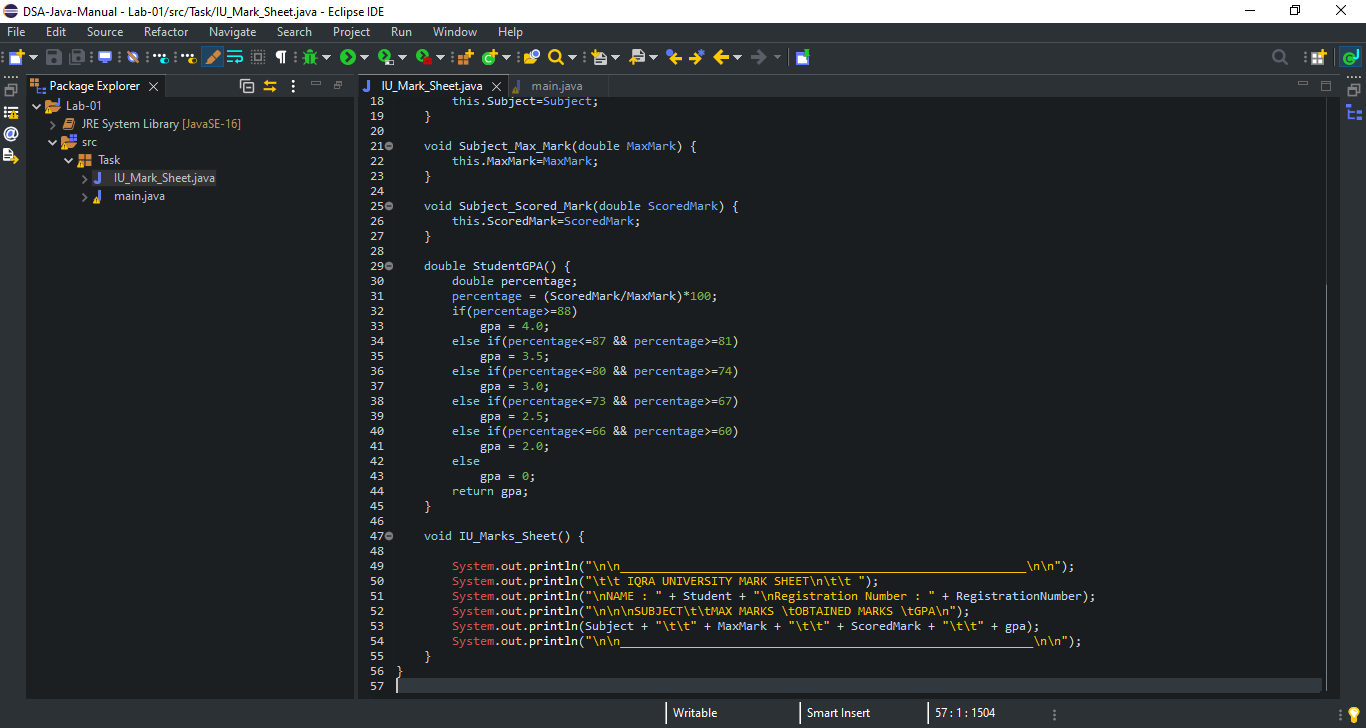
**Code Screen Shots:**

**main.java:**

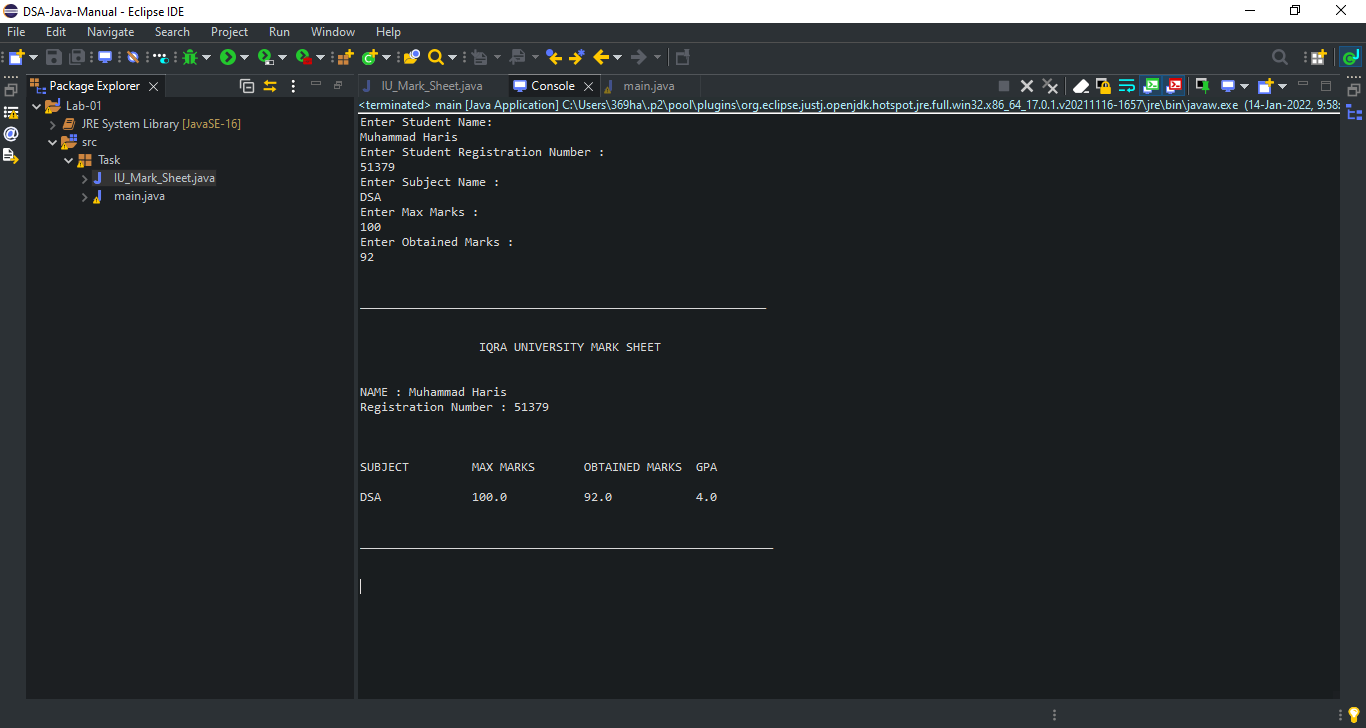


**IU\_Mark\_Sheet.java:**





**Code Output Screen Shot:**



**Assignment:**

**Program Statement:**

**You are required to model a vehicle parking lot system. The parking lot has a facility to park cars and scooters. The parking lot contains four parking lanes-two for cars and two for scooters.**

**Each lane can hold ten vehicles. There is an operator with a console at the East end of the parking lot. The system should be able to handle following scenarios.**

**Arrival of a vehicle:**

**1. Type of vehicle (car or scooter) and Registration No. of vehicle should be entered**

**2. Program should display suitable parking slot**

**3. Vehicles arrive at East end, leave from West end**

**Departure of Car:**

**1. If not western most, all cars should be moved to the west**

**2. When the car is driven out, rest of the cars should be moved to the left**

**3. Vehicle data should be updated**

**Departure of Scooter:**

**1. Scooters should be able to drive out at random**

**2. Vehicle data should be updated**

**Note that when desired the operator must be able to obtain information like number of vehicles, number of scooters or number of cars currently parked in the parking lot. Also, the system should be able to display all the parking lots (currently occupied) if desired.**

**Code:**

**CarParkManagement.java:**

package Assignment;

public class CarParkManagement {

public static void main(String[] args) {

Car BMW = new Car("Z4", "red",4, "Auto",false);

Bike Superfly = new Bike("XTR", "Black",2,true );

Car Hyundai = new Car("i30","Light Blue",4,"Auto",true);

CarPark myCarPark = new CarPark();

//Parking Vehicles

myCarPark.addVehicles(BMW);

myCarPark.addVehicles(Hyundai);

myCarPark.addVehicles(Superfly);

// Comment some of the above to see if it works

System.out.println("No. of Parked vehicles: " + myCarPark.numOfOccupiedSpots());

System.out.println("No. of Available Spots: "+ myCarPark.numOfAvailableSpots());

// Printing details

System.out.println("====================\n");

System.out.println("List of All Parked Vehicles is as folows: ");

myCarPark.printParkedVehicleDetails();

}

}

**Vechile.java:**

package Assignment;

public class Vehicle {

private String model;

private String color;

private int numOfWheels;

private String transmission;

// constructor

public Vehicle(String model, String color, int numOfWheels, String transmission){

this.model = model;

this.color = color;

this.numOfWheels = numOfWheels;

this.transmission = transmission;

}

// default constructor

public Vehicle() {

// TODO Auto-generated constructor stub

}

// getters

public String getModel(){

return model;

}

public String getColor(){

return color;

}

public int NumOfWheels(){

return numOfWheels;

}

public String transmission(){

return transmission;

}

// setters

public void setNumOfWheels (int numOfWheels){

this.numOfWheels = numOfWheels;

}

public void setModel(String model){

this.model = model;

}

public void setColor(String color){

this.color = color;

}

public void setTransmission(String transmission){

this.transmission = transmission;

}

// Print Method

public void printDetails(){

System.out.println("The color of this vehicle is " + this.color);

System.out.println("The model of this vehicle is " + this.model);

System.out.println("The transmission for this vehicle is " + this.transmission);

System.out.println("The number of wheels in this vehicle is "+ this.numOfWheels);

}

}

**Bike.java:**

package Assignment;

public class Bike extends Vehicle {

private boolean suspension;

public Bike(){

super();

}

public Bike(String model, String color, int numOfWheels, boolean suspension){

super( model, color, numOfWheels, "na");

// na for transmission as Bike class does not need a suspension property

this.suspension = suspension;

}

// getter

public boolean getSuspension(){

return suspension;

}

// setter

public void setSuspension(){

this.suspension = suspension;

}

// method

public void printBikeDetails(){

super.printDetails();

System.out.println("Does this bike have suspension ?? " + this.suspension);

}

}

**Car.java:**

package Assignment;

public class Car extends Vehicle{

private Boolean roof;

// Constructors

public Car(){

super();

}

public Car(String model, String color, int numOfWheels, String transmission, Boolean roof){

super(model, color, numOfWheels, transmission);

this.roof = roof;

}

// getter

public boolean getRoof(){

return roof;

}

// setter

public void setRoof(boolean roof){

this.roof=roof;

}

// Methods

public void printdetailscar(){

super.printDetails(); // Calling superclass method

System.out.println(" does this car has a roof: " + this.roof);

}

}

**CarPark.java:**

package Assignment;

import java.util.ArrayList;

public class CarPark {

private static final int maxCapacity = 3;

private ArrayList <Vehicle>carParkArray;

// Every member of carParkArray will be of type Vehicle

// constructor

public CarPark(){

this.carParkArray = new ArrayList<Vehicle>(maxCapacity);

//This ensures car PArk has three vehicles max

}

// Method to find maximum capacity of the car park

public static int getMaxCapacity(){

return maxCapacity;

}

// Method to find available spots in the carpark

public int numOfAvailableSpots(){

return this.maxCapacity - this.carParkArray.size();

}

// Method to find number of occupied spaces in the carpark

public int numOfOccupiedSpots(){

return this.carParkArray.size();

}

//Method to add vehicles to the carpark

public void addVehicles(Vehicle vehicle){

if(this.numOfAvailableSpots() > 0){

this.carParkArray.add(vehicle);

vehicle.printDetails();

System.out.println("++++++++++++ \n");

} else {

System.out.println("Car Park is full");

}

}

//Method to print parked vehicle details

public void printParkedVehicleDetails(){

// For every vehicle in array carParkArray, do blah blah

for(Vehicle vehicle: this.carParkArray){

vehicle.printDetails();

System.out.println("++++++++++++++++ \n");

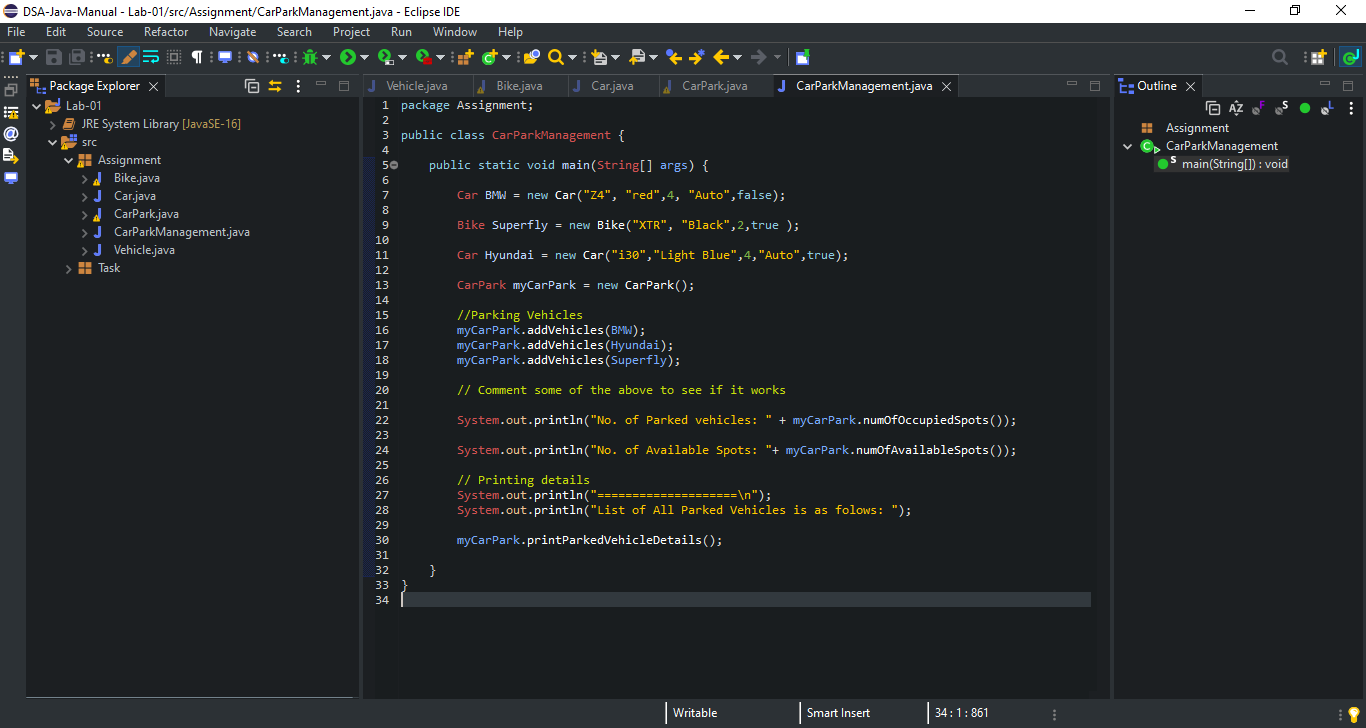
}

}

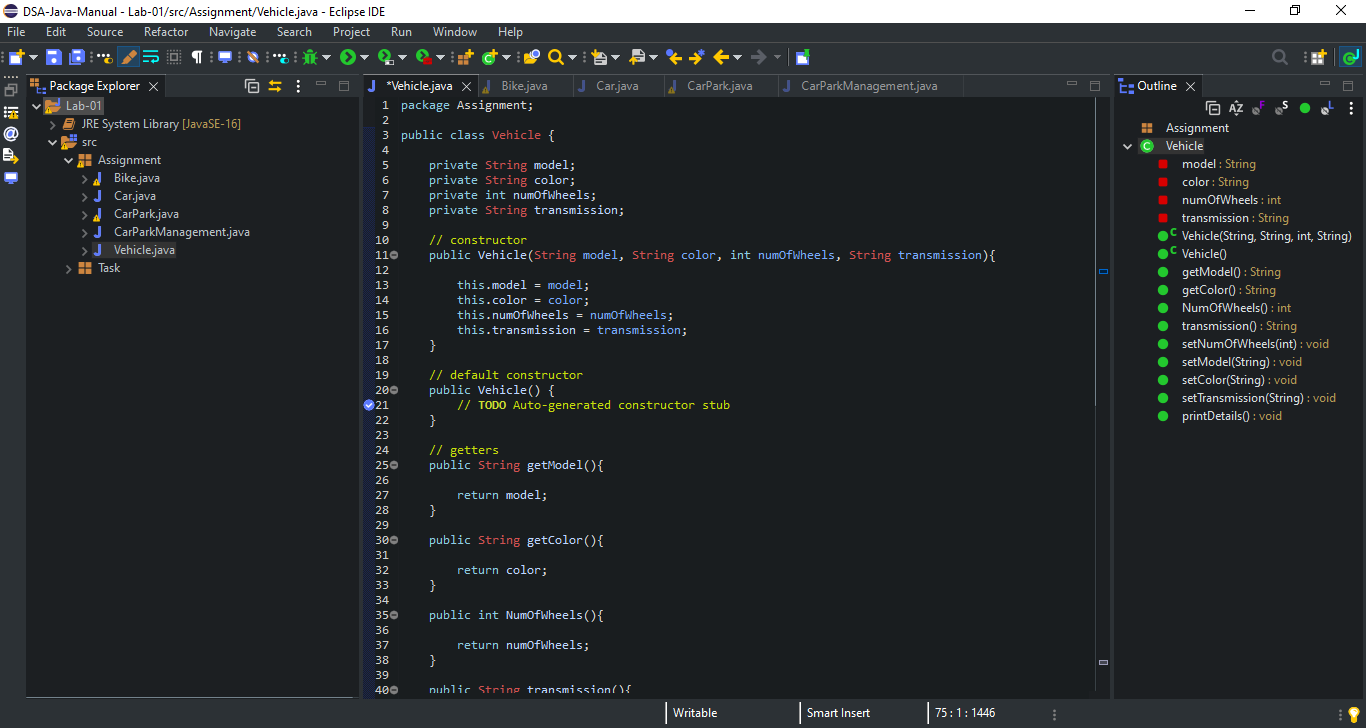
}

**Code Screen Shots:**

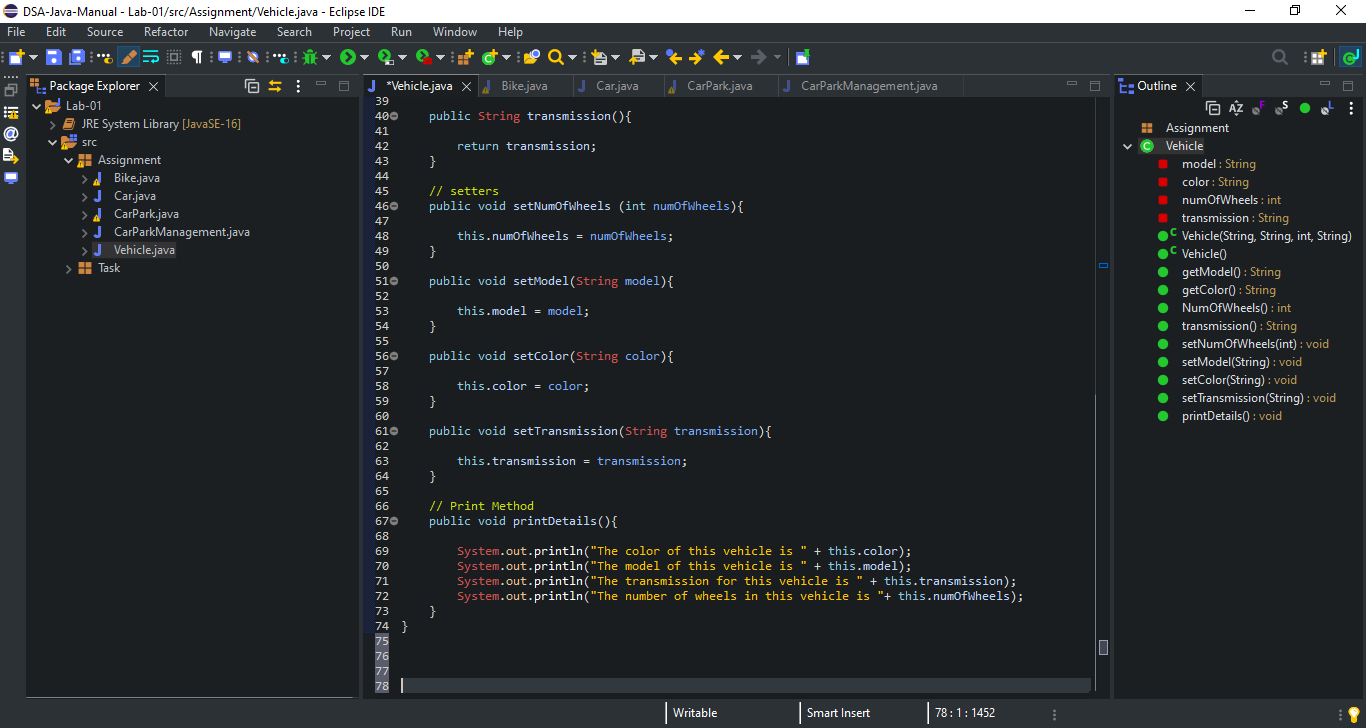
**CarParkManagement.java:**



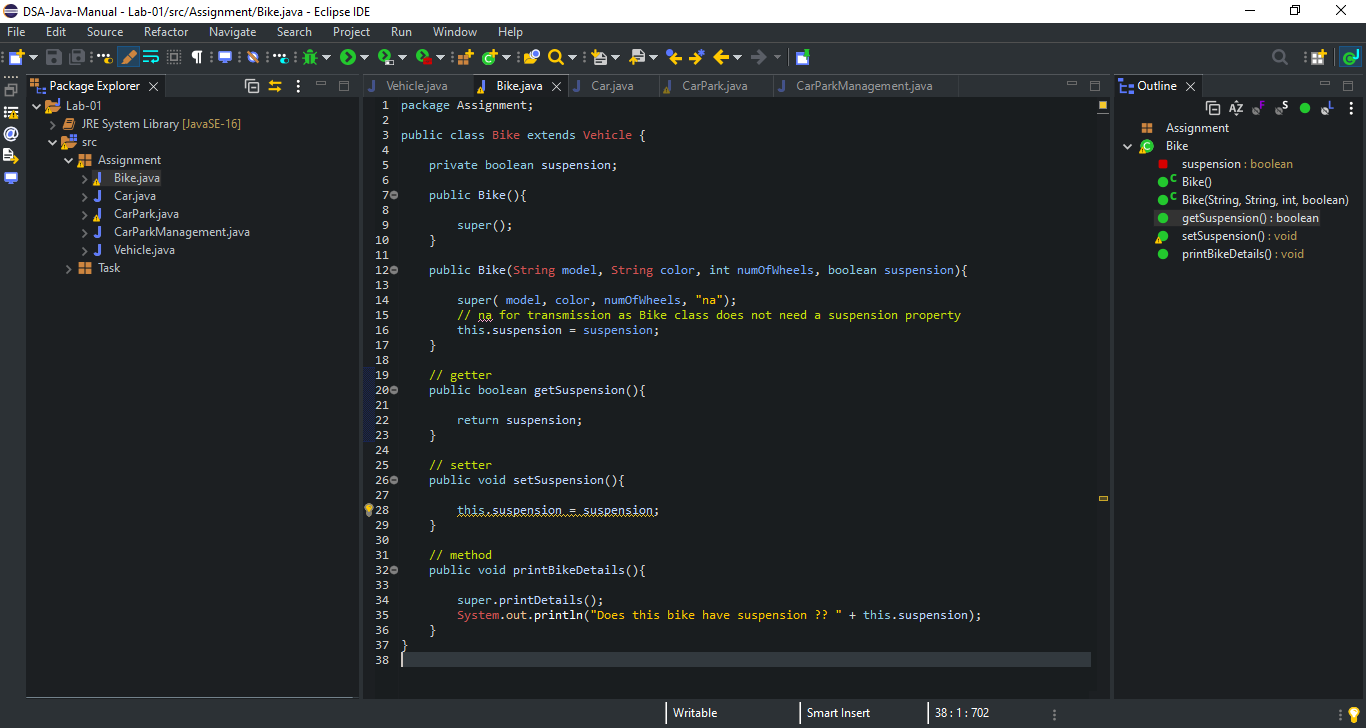
**Vechile.java:**



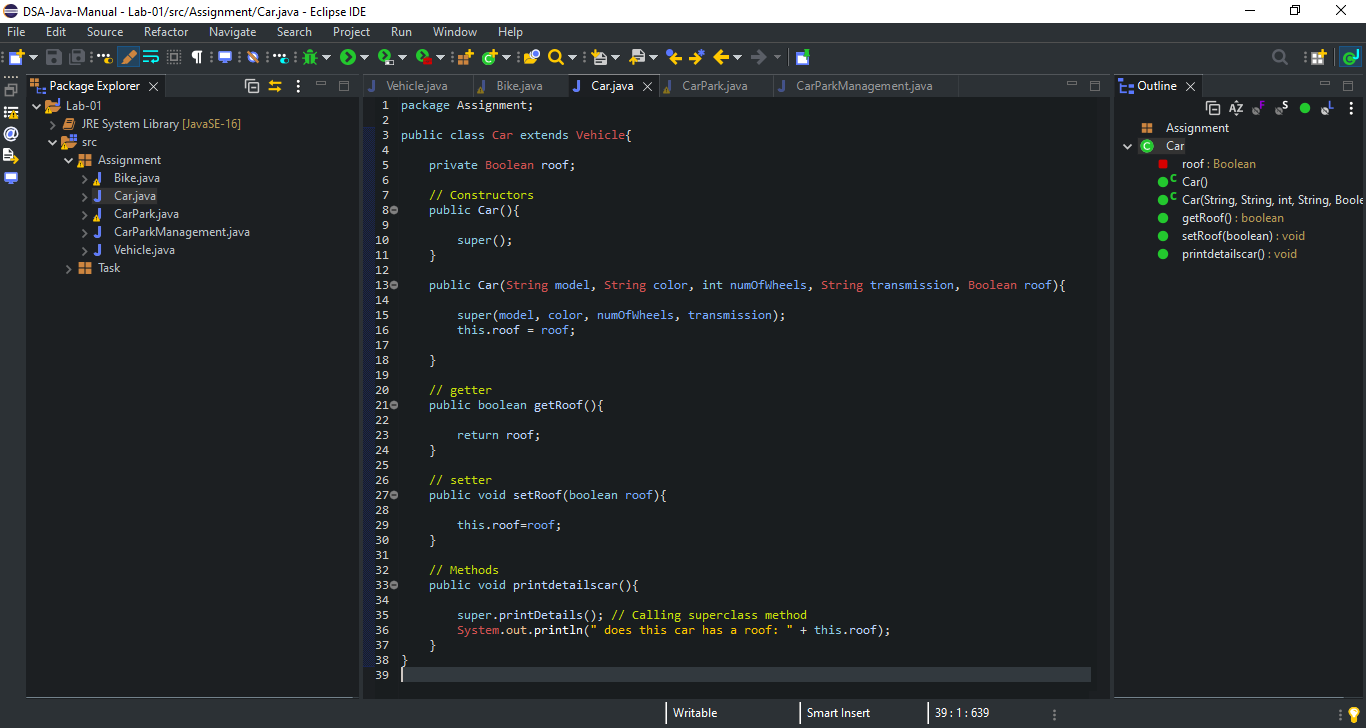
**Continue…**



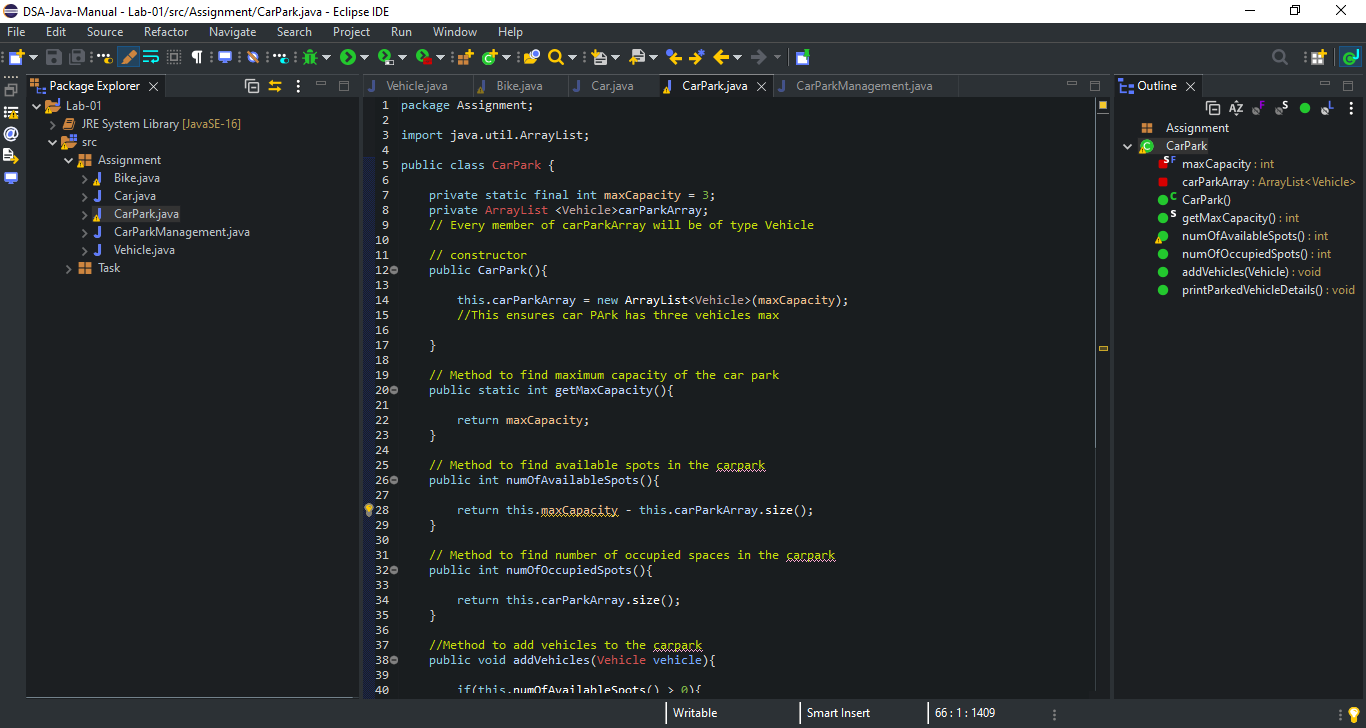
**Bike.java:**

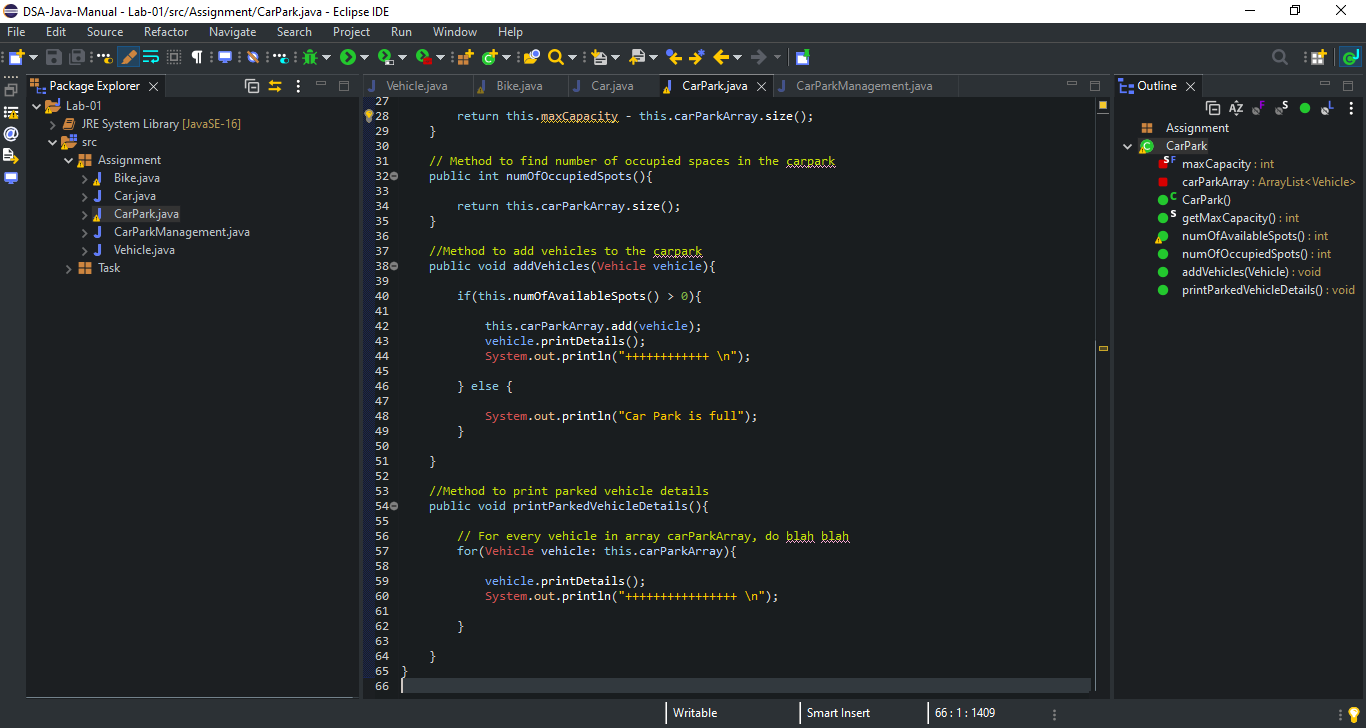


**Car.java:**

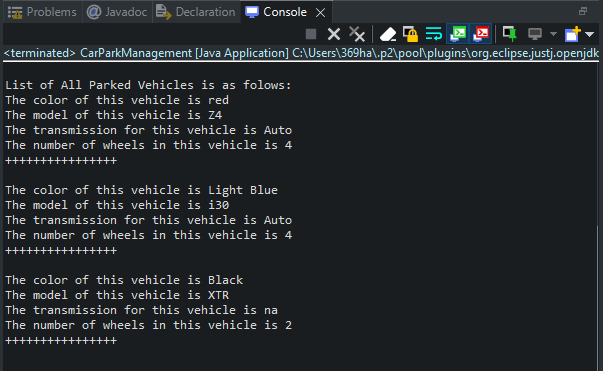


**CarPark.java:**





**Code Output Screen Shot:**



**Lab: 02**

**Lab Tasks 1:**

* **Develop Method in java which can find the smallest and the largest value from integer Array.**
  + **int[] TwoExtreamValue(int A[]).**
* **Develop Method in java which can sort integer array using bubble sort.** 
  + **Int[] SortArray(int A[]).**
* **Vector Multiplication Using two integer Arrays**
  + **int VectorMultiplication(int A[],int B[]).**

**Code:**

**valueFinder.java:**

package Task\_01;

public class multiplyVector {

public static int VectorMultiplication(int A[], int B[]) {

int size = A.length;

int mul = 0;

for (int i = 0; i < size; i++) {

mul = mul + A[i] \* B[i];

}

return mul;

}

}

**sortArray.java:**

package Task\_01;

public class sortArray {

public static int[] SortArray(int a[]) {

int size = a.length;

for(int i=0;i<size;i++) {

for (int j=0;j<size;j++) {

if(a[i]<a[j]) {

int temp = a[i];

a[i]=a[j];

a[j] = temp;

}

}

}

return a;

}

}

**multiplyVector.java:**

package Task\_01;

public class multiplyVector {

public static int VectorMultiplication(int A[], int B[]) {

int size = A.length;

int mul = 0;

for (int i = 0; i < size; i++) {

mul = mul + A[i] \* B[i];

}

return mul;

}

}

**main.java:**

package Task\_01;

import java.util.Scanner;

public class main {

public static void main(String[] args) {

valueFinder vf = new valueFinder();

sortArray sa = new sortArray();

multiplyVector mv = new multiplyVector();

Scanner scan = new Scanner(System.in);

int[] array = new int[3];

int max = 3;

System.out.println("1.Value Finder (smallest/largest)\n2.Sort Array \n3.Multiply Two Vectors\nSelect your option (1, 2, 3) : ");

double selectedOption = scan.nextDouble();

if (selectedOption == 1) {

System.out.println("Enter integers numbers : ");

for(int i = 0; i <= max; i++) {

array[i] = scan.nextInt();

vf.TwoExtreamValue(array);

}

}

else if (selectedOption == 2) {

System.out.println("Enter integers numbers Array : ");

for(int i = 0; i <= max; i++) {

array[i] = scan.nextInt();

sa.SortArray(array);

}

}

else {

System.out.println("Enter integers numbers (with 'space') : ");

for(int i = 0; i <= max; i++) {

array[i] = scan.nextInt();

mv.VectorMultiplication(null, null);

}

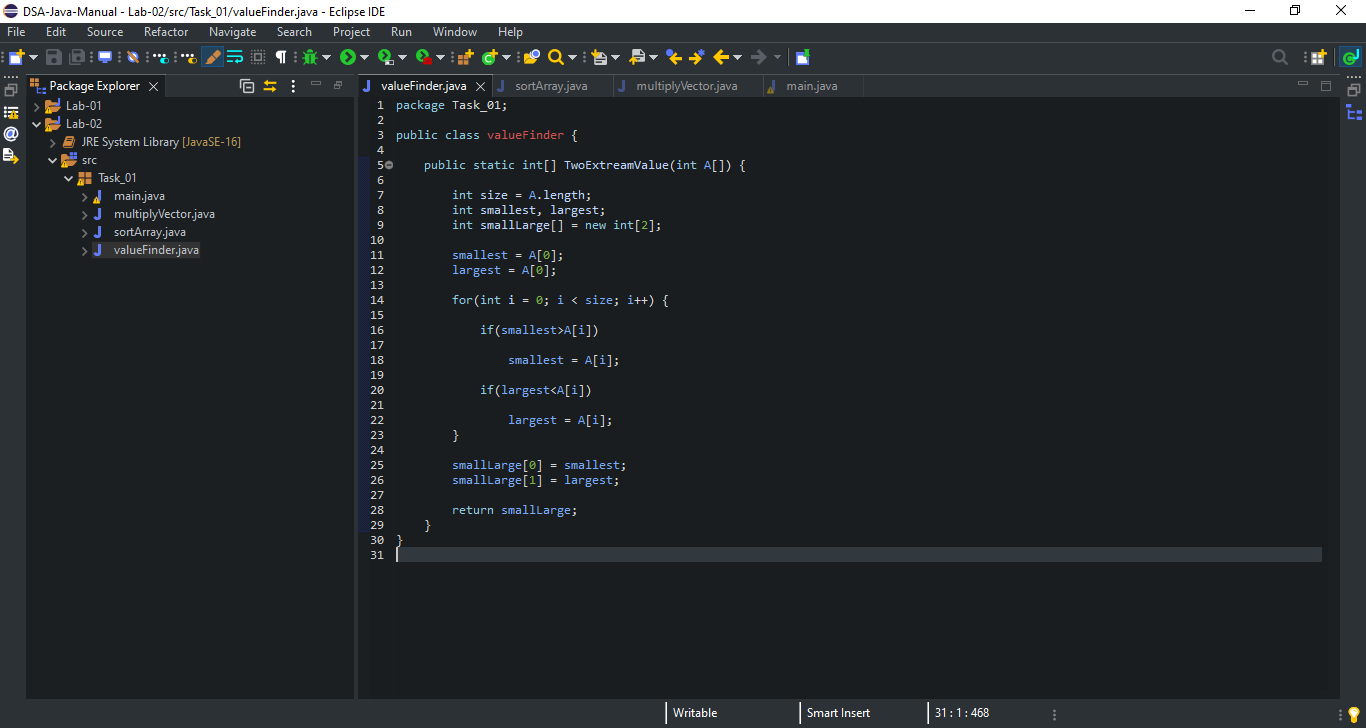
}

}

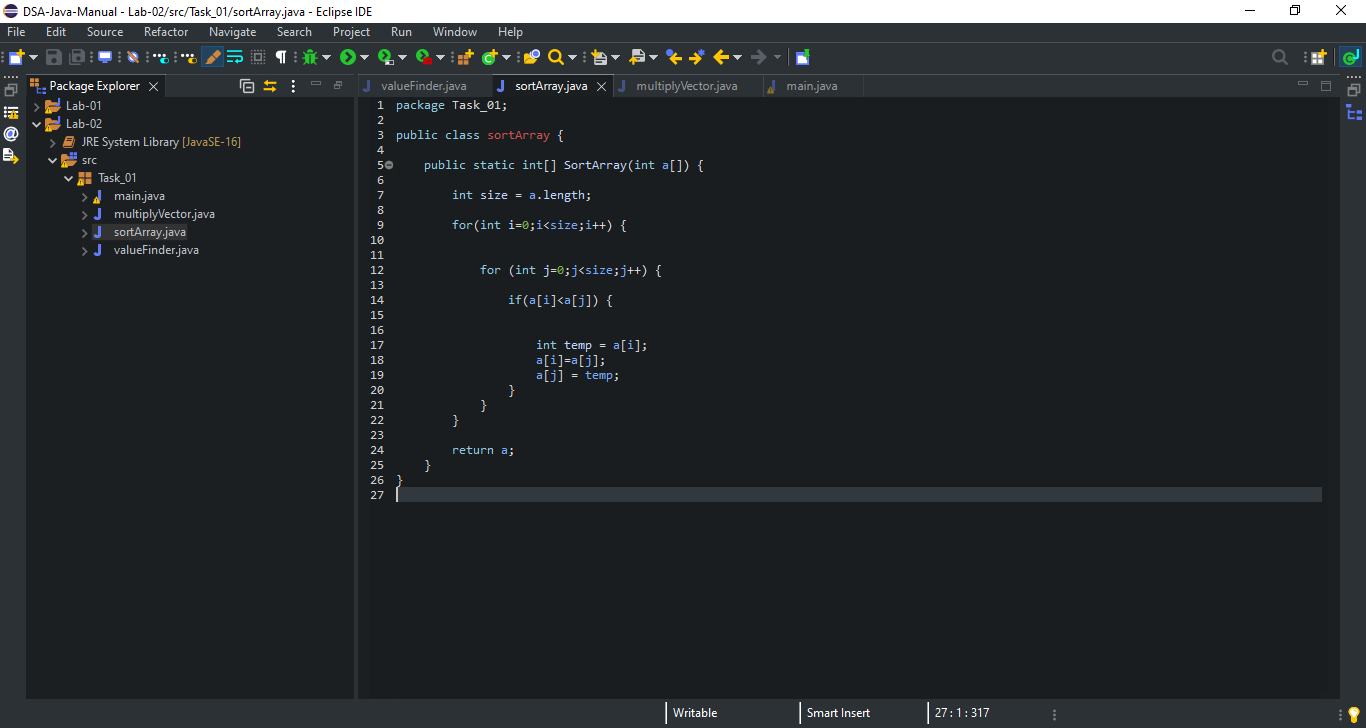
}

**Code Screen Shots:**

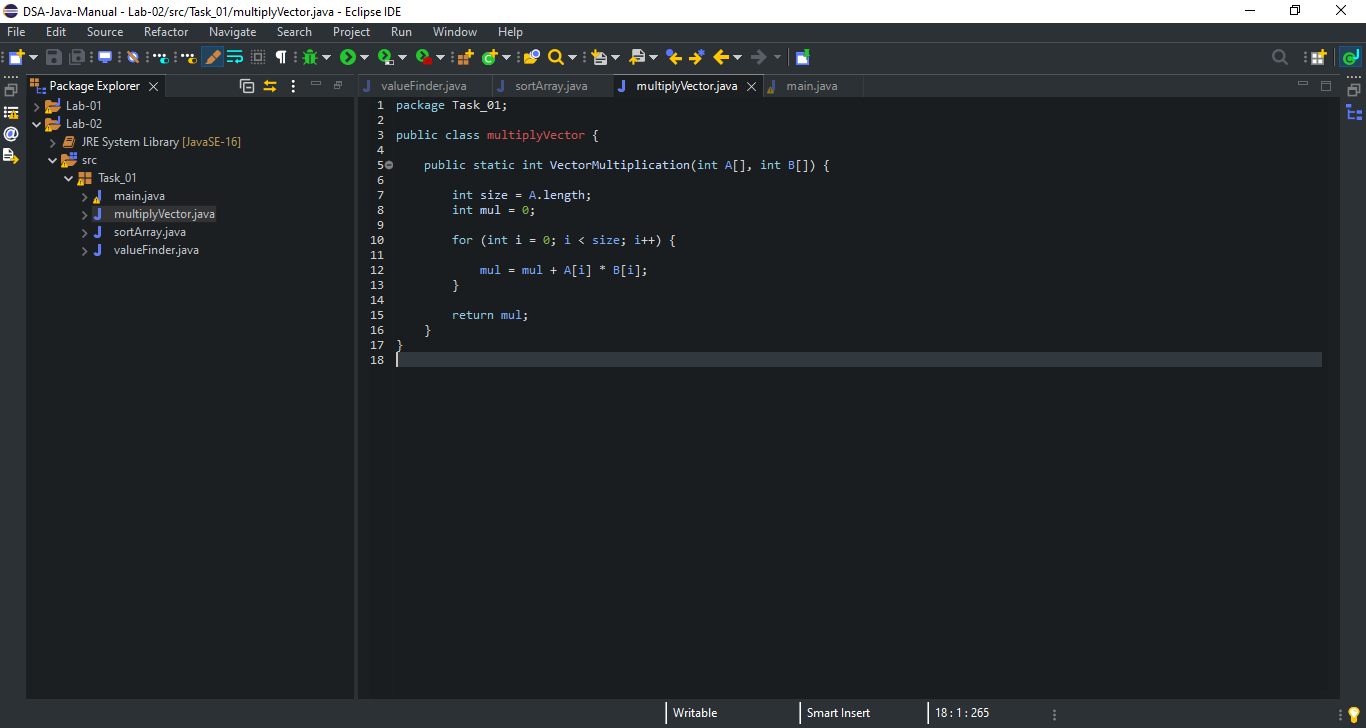
**valueFinder.java:**



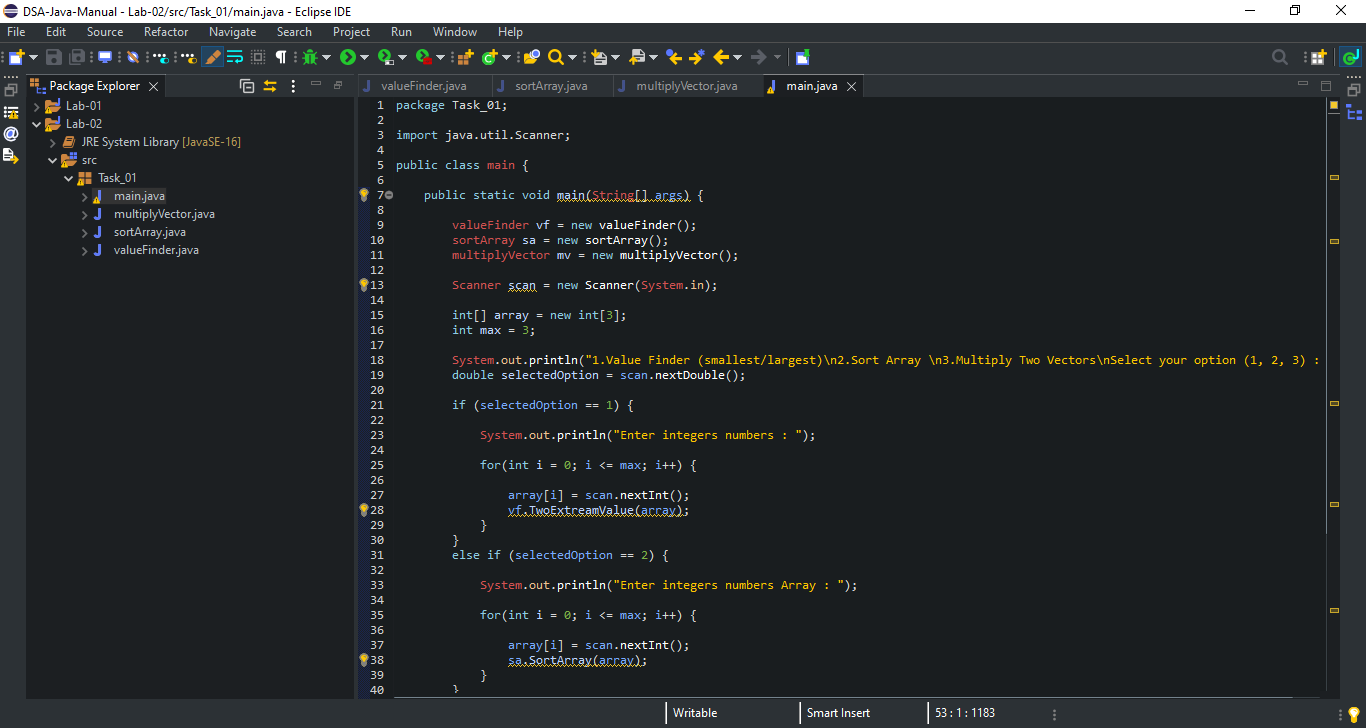
**sortArray.java:**

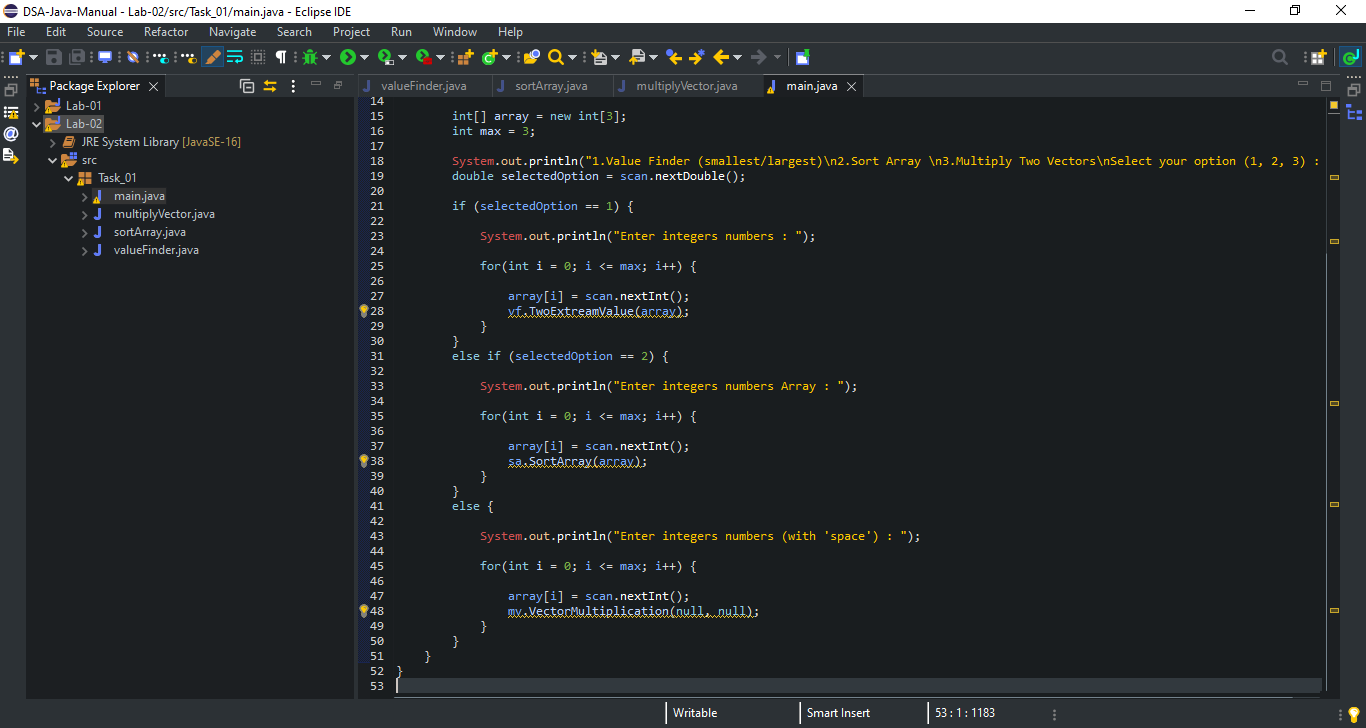


**multiplyVector.java:**



**main.java:**





**Lab Tasks 2:**

**Develop Generic Class which can set Data type of Single dimension array.**

**Code:**

**main.java:**

package Task\_02;

public class main {

public static void main(String[] args) {

final int length = 5;

Array<Integer>int\_Array = new Array(length);

System.out.print("\nGeneric Array <Integer>:" + " ");

for (int i = 0; i < length; i++)

int\_Array.set(i, i \* 2);

System.out.println(int\_Array);

Array<String>str\_Array = new Array(length);

System.out.print("\nGeneric Array <String>:" + " ");

for (int i = 0; i < length; i++)

str\_Array.set(i, String.valueOf((char)(i + 97)));

System.out.println(str\_Array);

}

}

**array.java:**

package Task\_02;

import java.util.Arrays;

class Array<E> {

private final Object[] obj\_array;

public final int length;

public Array(int length) {

obj\_array = new Object [length];

this.length = length;

}

E get(int i) {

@SuppressWarnings("unchecked")

final E e = (E)obj\_array[i];

return e;

}

void set(int i, E e) {

obj\_array[i] = e;

}

@Override

public String toString() {

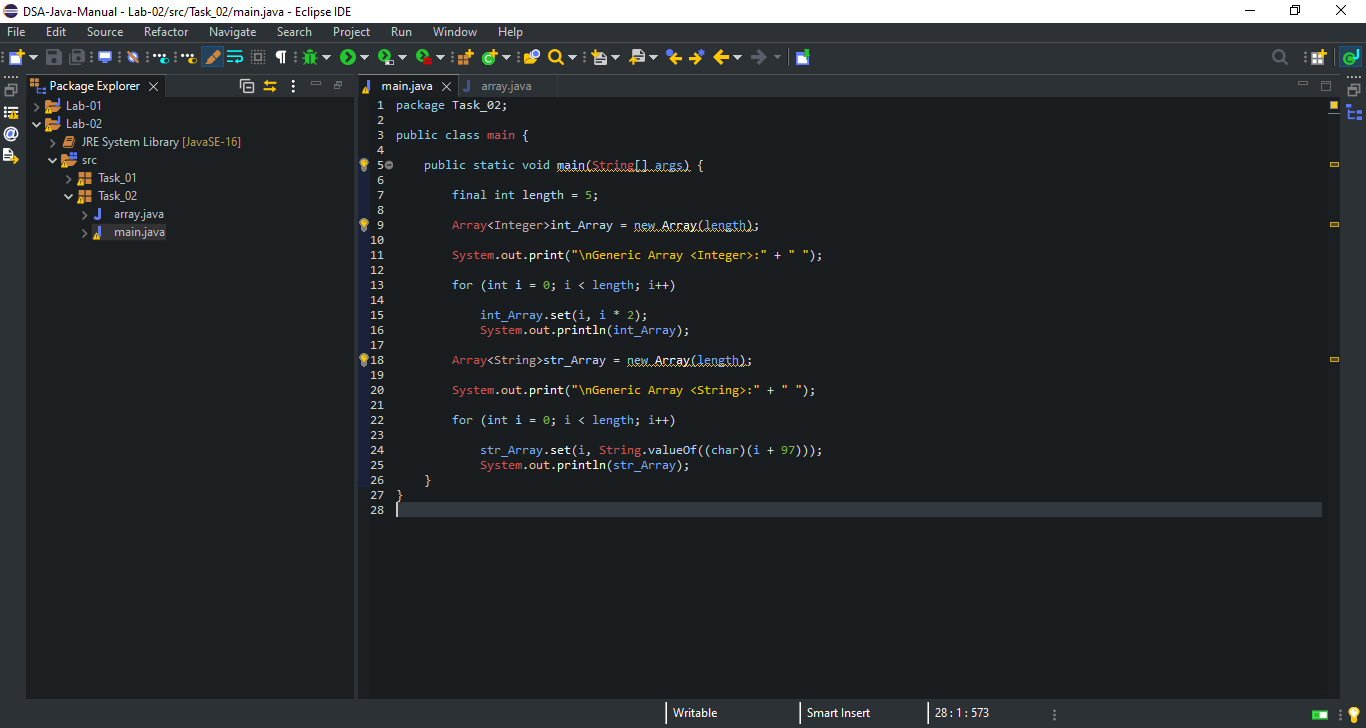
return Arrays.toString(obj\_array);

}

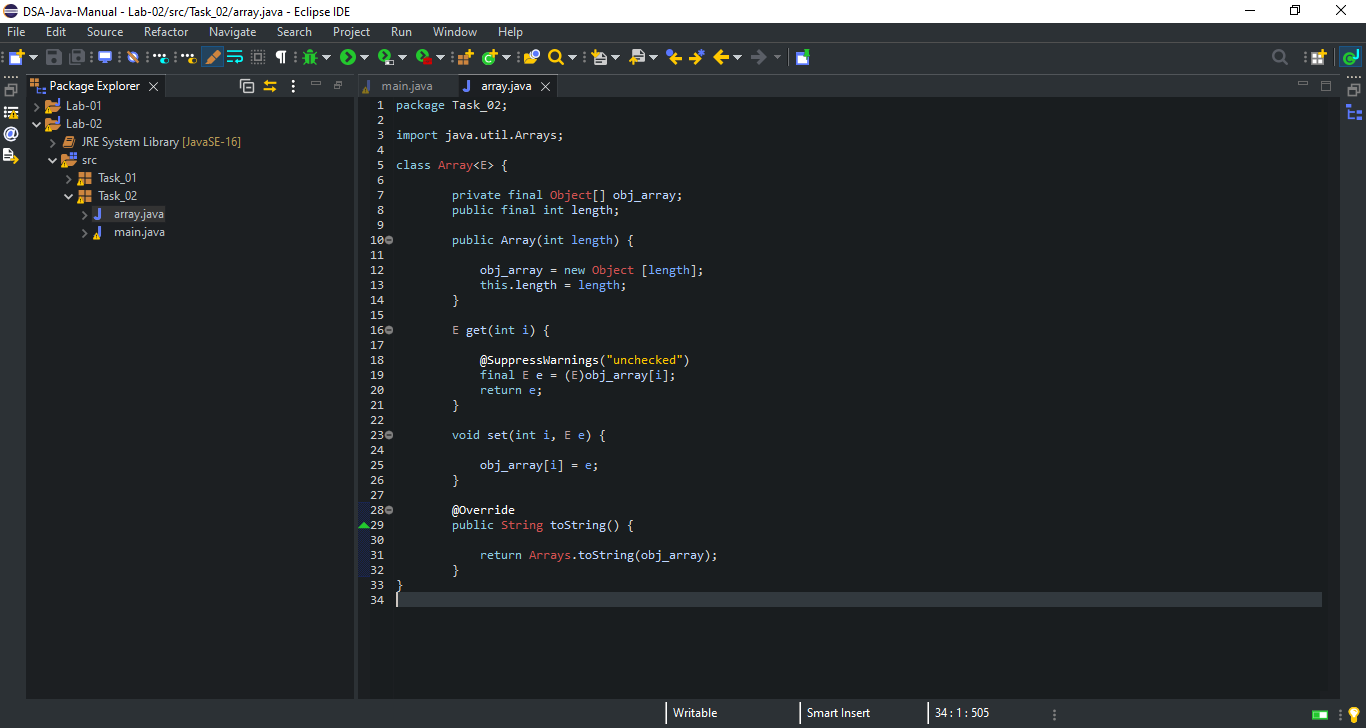
}

**Code Screen Shot:**

**main.java:**



**array.java:**



**Code Output Screen Shot:**



**Lab: 03**

**Task 1:**

**Develop your own ArrayList in Java having following methods:**

* **add ()**
* **remove()**
* **get()**
* **contain()**
* **is\_empty()**
* **IndexOff()**
* **ToArray()**
* **TrimToSize()**

**Code:**

**CustomArrayList.java:**

package Task\_01;

public class CustomArrayList<T>

{

private T[] array;

private int size = 0;

public CustomArrayList()

{

array = (T[]) new Object[2];

}

public void add(T value)

{

array[size] = value;

size++;

if(size==array.length)

{

T[] newArray = (T[]) new Object[array.length\*2];

for(int i=0;i<array.length;i++)

newArray[i] = array[i];

array = newArray;

}

}

public T remove(T value)

{

for(int i=0;i<size-1;i++)

{

if(value==array[i])

array[i] = array[i+1];

}

size--;

return value;

}

public int getSize()

{

return size;

}

public boolean contain(T value)

{

for(int i=0;i<size;i++)

{

if(value==array[i])

return true;

}

return false;

}

public int indexOff(T value)

{

for(int i=0;i<size;i++)

{

if(value==array[i])

{

return i;

}

}

return -1;

}

public boolean isEmpty()

{

if(size==0)

return true;

else

return false;

}

public T display(int index)

{

return array[index];

}

}

**main.java:**

package Task\_01;

public class main {

public static void main(String[] args)

{

CustomArrayList<Integer> ref = new CustomArrayList<Integer>();

CustomArrayList<String> ref1 = new CustomArrayList<String>();

System.out.println("1st Reference Variable ArrayList: ");

ref.add(1);

ref.add(2);

ref.add(3);

ref.add(4);

System.out.println("Size of List: "+ref.getSize());

System.out.println("Value 3 contains in List: "+ref.contain(3));

System.out.println("Removed Value: "+ref.remove(3));

System.out.println("Value 2 presents at "+ref.indexOff(2)+" index");

for(int i=0;i<ref.getSize();i++)

System.out.println("\tValue["+i+"]: "+ref.display(i));

System.out.println("2nd Reference Variable: ");

ref1.add("Muhammad");

ref1.add("Haris");

ref1.add("Ali");

ref1.add("Ahmed");

System.out.println("Size of List: "+ref1.getSize());

System.out.println("Haris contains in List: "+ref1.contain("Haris"));

System.out.println("Removed Value: "+ref1.remove("Ali"));

System.out.println("Shivam presents at "+ref1.indexOff("Ahmed")+" index");

for(int i=0;i<ref1.getSize();i++)

System.out.println("\t Name["+i+"]: "+ref1.display(i));

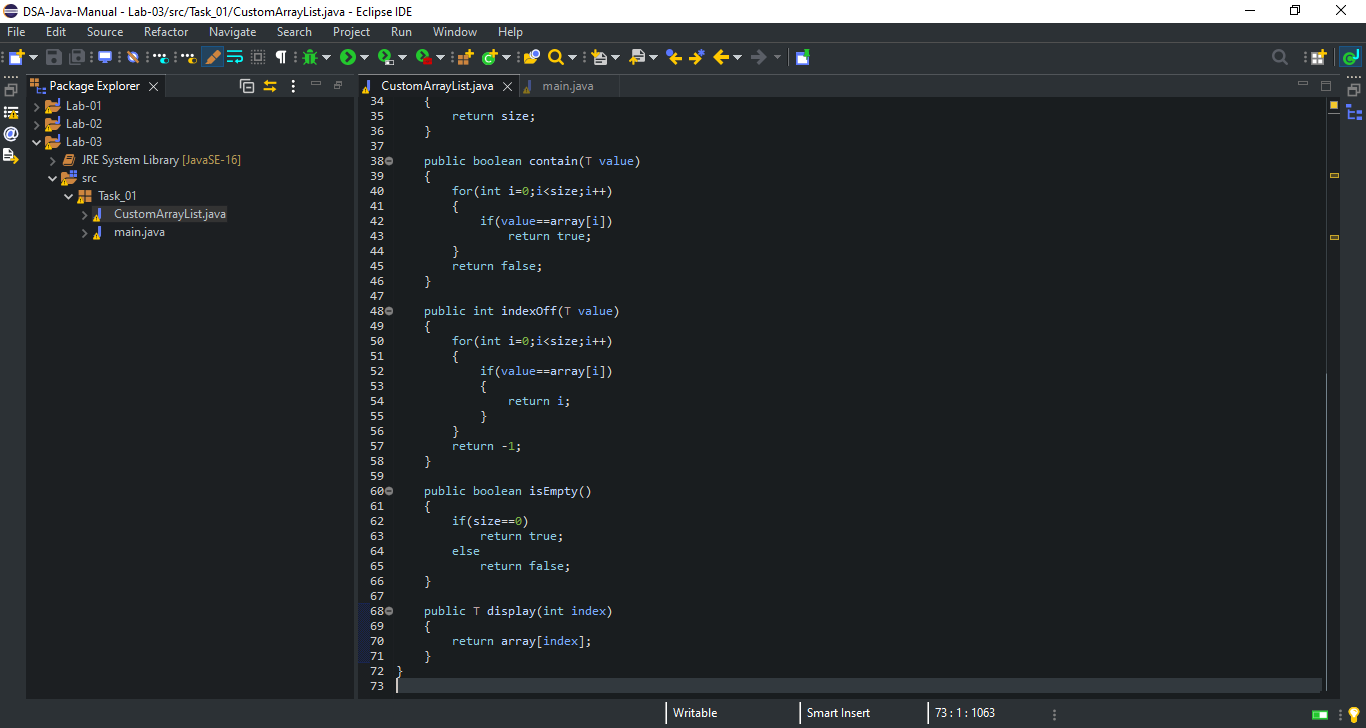
}

}

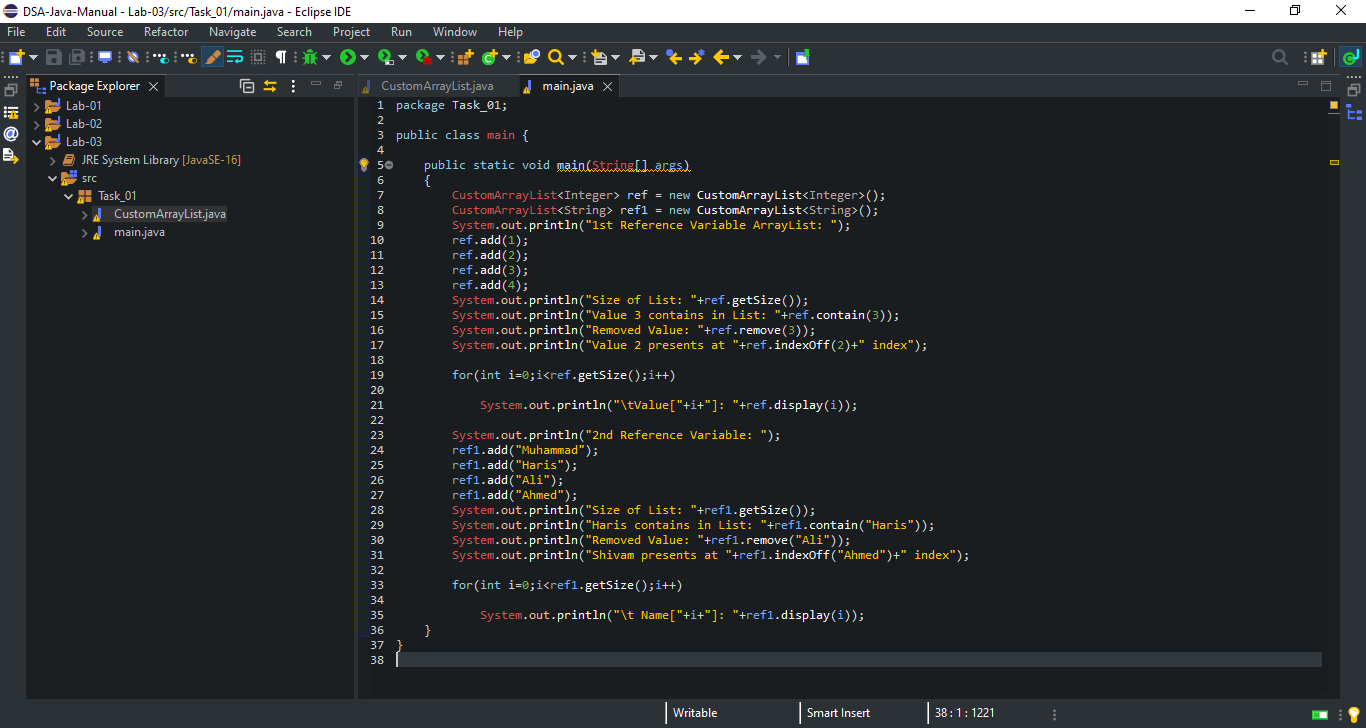
**Code Screen Shot:**

**CustomArrayList.java:**

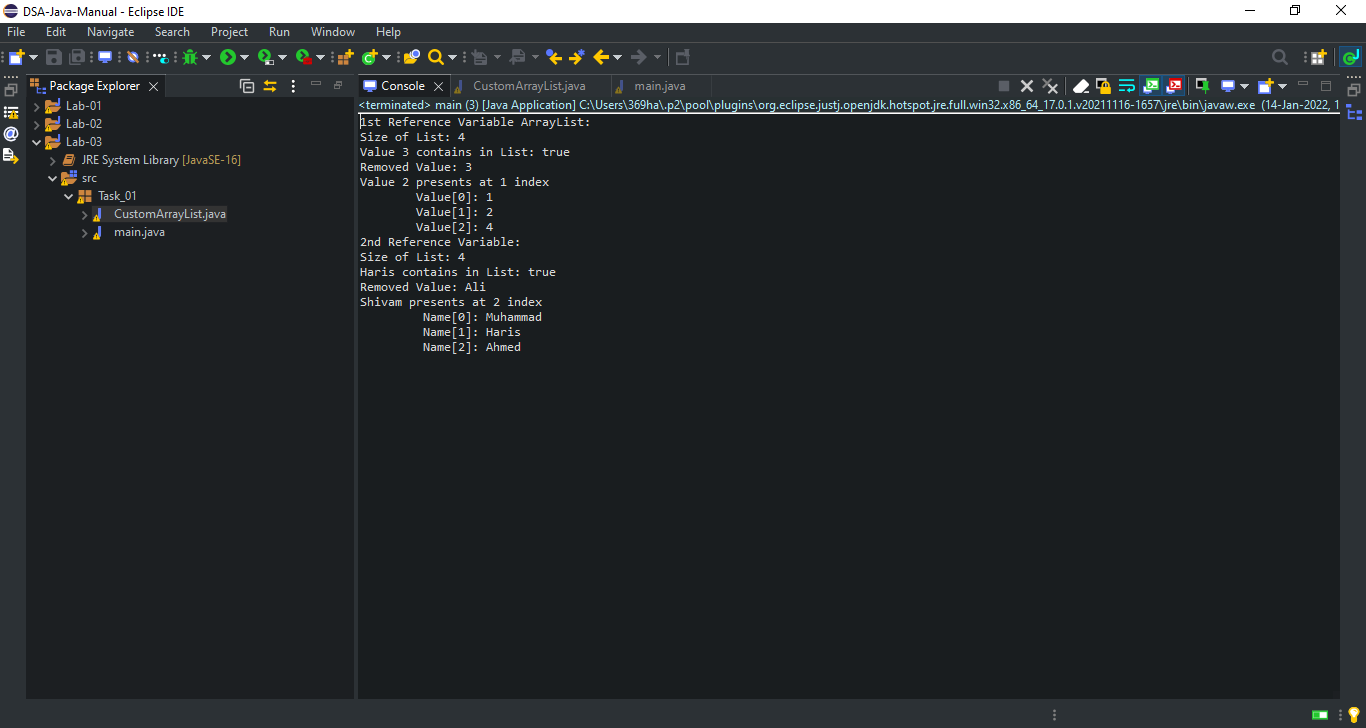




**main.java:**



**Code Output Screen Shot:**



**Lab: 04**

**Lab Tasks:**

**Write a program in java to insert a new node at the middle of Singly Linked List.**

**Code:**

**main.java:**

package Task;

public class main {

public static void main(String[] args) {

InsertNode sList = new InsertNode();

sList.addNodeAtEnd(1);

sList.addNodeAtEnd(2);

sList.addNodeAtEnd(5);

sList.display();

sList.addInMid(3);

System.out.println("Updated List: ");

sList.display();

sList.addInMid(4);

System.out.println("Updated List: ");

sList.display();

System.out.println("Updated List: ");

sList.addAtStart(0);

sList.display();

}

}

**InsertNode.java:**

package Task;

public class InsertNode {

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

public int size;

public Node head = null;

public Node tail = null;

public void addAtStart(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

tail = newNode;

}

else {

Node temp = head;

head = newNode;

head.next = temp;

}

}

public void addNodeAtEnd(int data) {

Node newNode = new Node(data);

if(head == null) {

head = newNode;

tail = newNode;

}

else {

tail.next = newNode;

tail = newNode;

}

size++;

}

public void addInMid(int data) {

Node newNode = new Node(data);

if(head == null) {

head = newNode;

tail = newNode;

}

else {

Node temp, current;

int count = (size % 2 == 0) ? (size/2) : ((size+1)/2);

temp = head;

current = null;

for(int i = 0; i < count; i++) {

current = temp;

temp = temp.next;

}

current.next = newNode;

newNode.next = temp;

}

size++;

}

public void display() {

Node current = head;

if(head == null) {

System.out.println("List is empty");

return;

}

while(current != null) {

System.out.print(current.data + " ");

current = current.next;

}

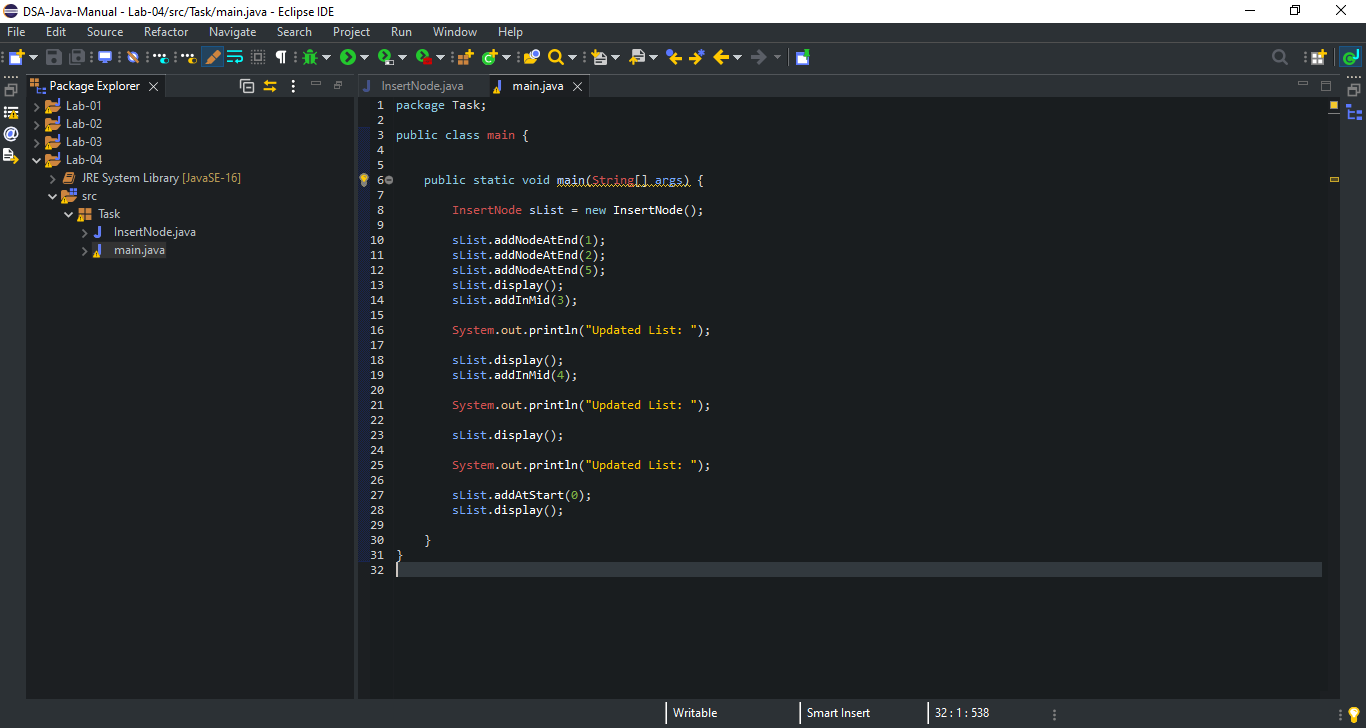
System.out.println();

}

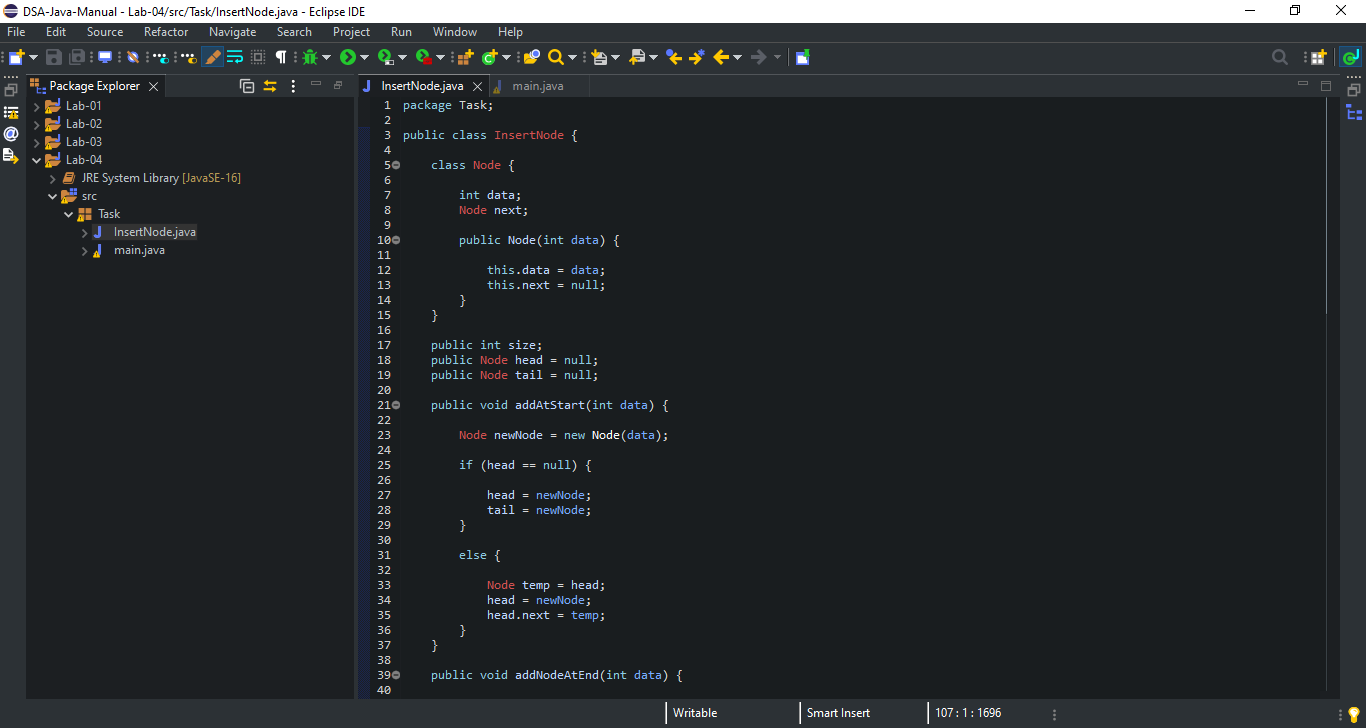
}

**Code Screen Shot:**

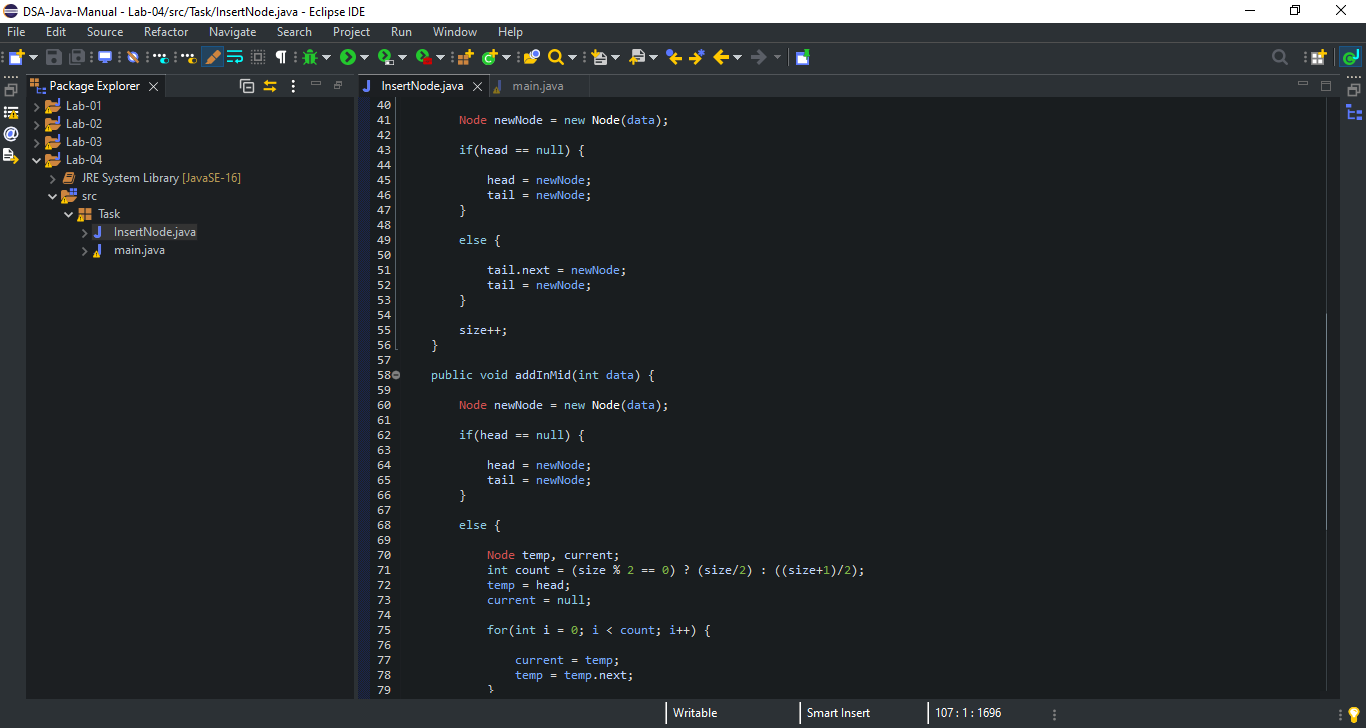
**main.java:**

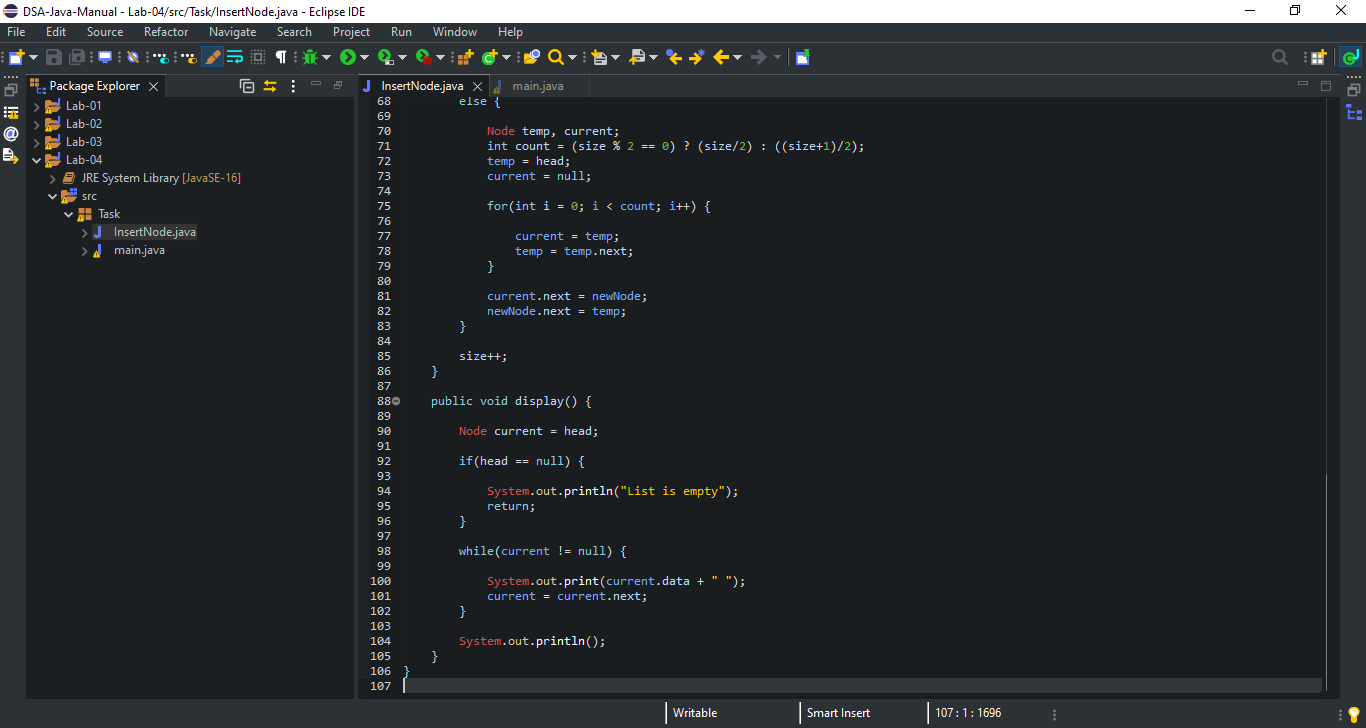


**InsertNode.java:**

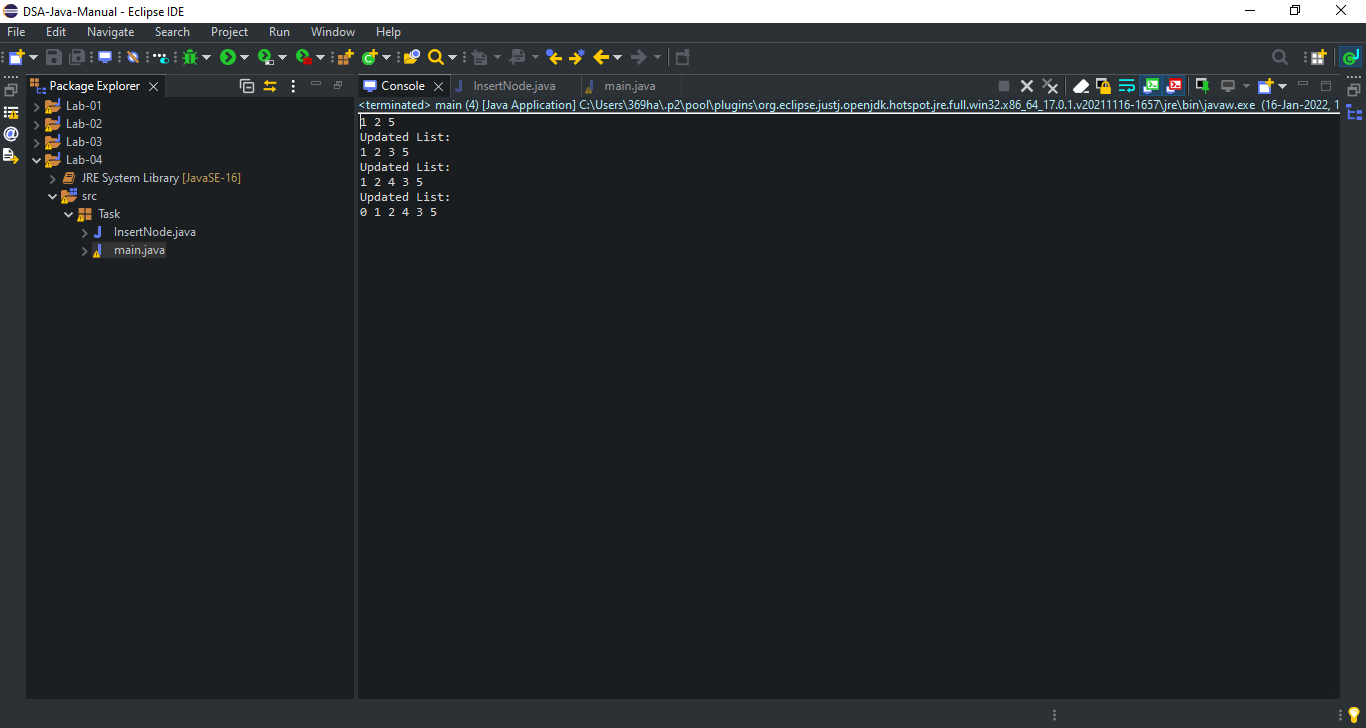


**Continue…**





**Code Output Screen Shot:**



**Lab: 05**

**Task 1:**

**Using Recursion Perform the following Tasks**

* **Calculate power of a given number in java**
* **Reverse a String using recursion in Java**

**(a). Calculate power of a given number in java**

**Code:**

**main.java:**

package Task\_a;

import java.util.Scanner;

public class main {

public static void main(String[] args) {

power pp = new power();

Scanner scan = new Scanner(System.in);

System.out.print("Enter a number:");

int num = scan.nextInt();

System.out.print("Enter power:");

int pow = scan.nextInt();

float power = pp.power(num,pow);

if(pow > 0) {

int p = (int) power;

System.out.print(num+"^"+pow+" = "+p);

}

else {

System.out.print(num+"^"+pow+" = "+power);

}

}

}

**power.java:**

package Task\_a;

public class power {

public static float power(int num, int pow) {

if(pow == 0) {

return 1;

}

else if (pow < 0) {

return 1/(power(num,-pow));

}

else {

return (num\*power(num,pow-1));

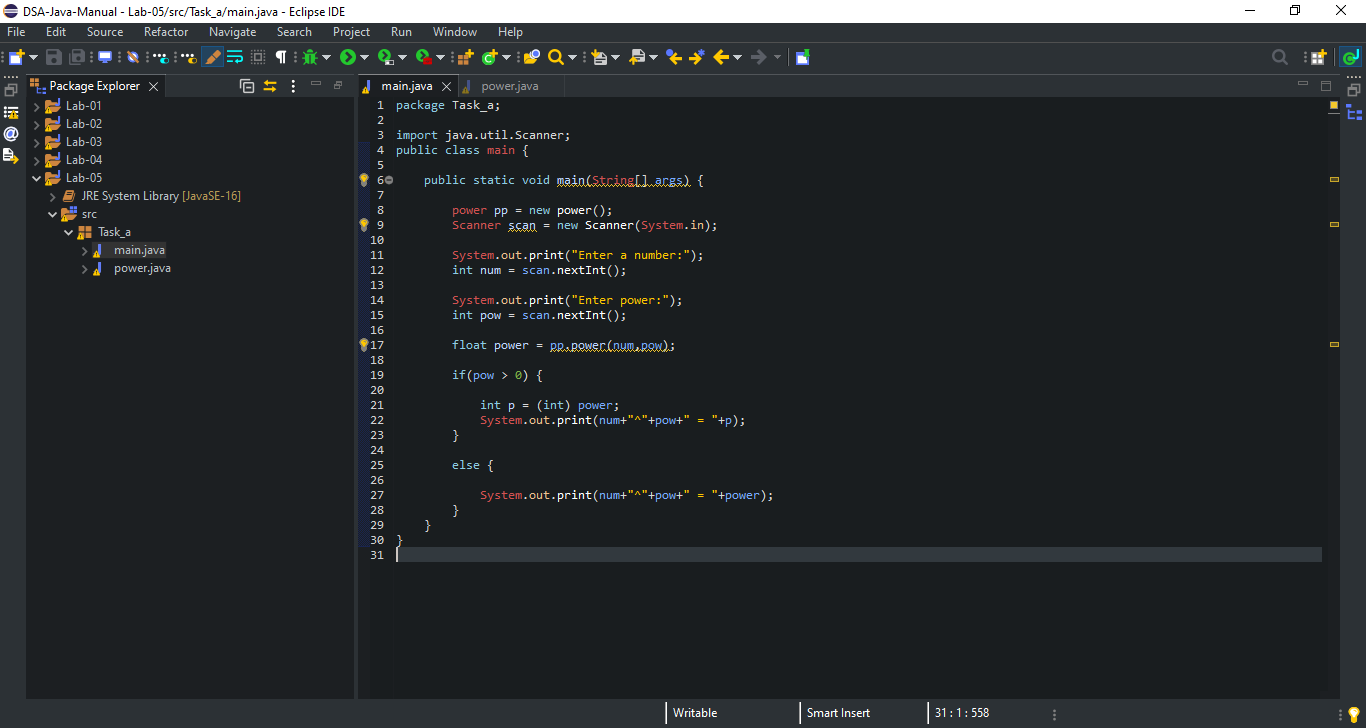
}

}

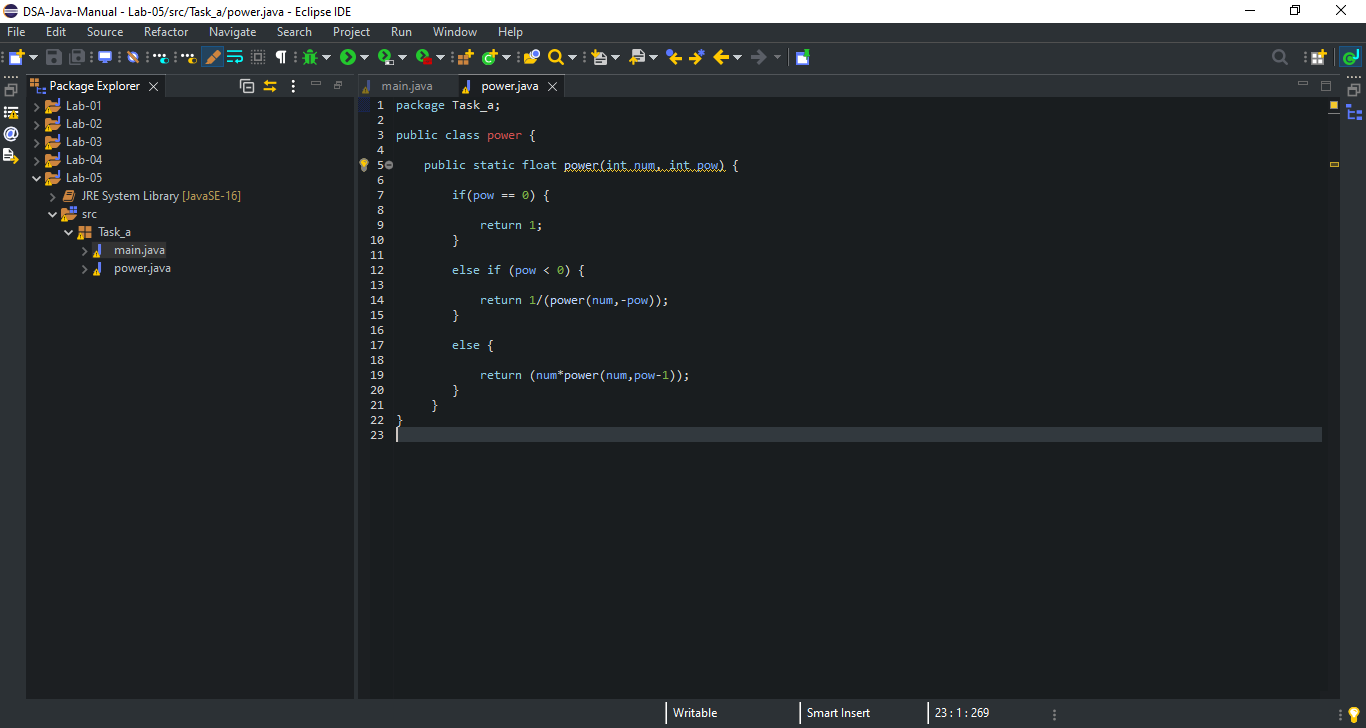
}

**Code Screen Shot:**

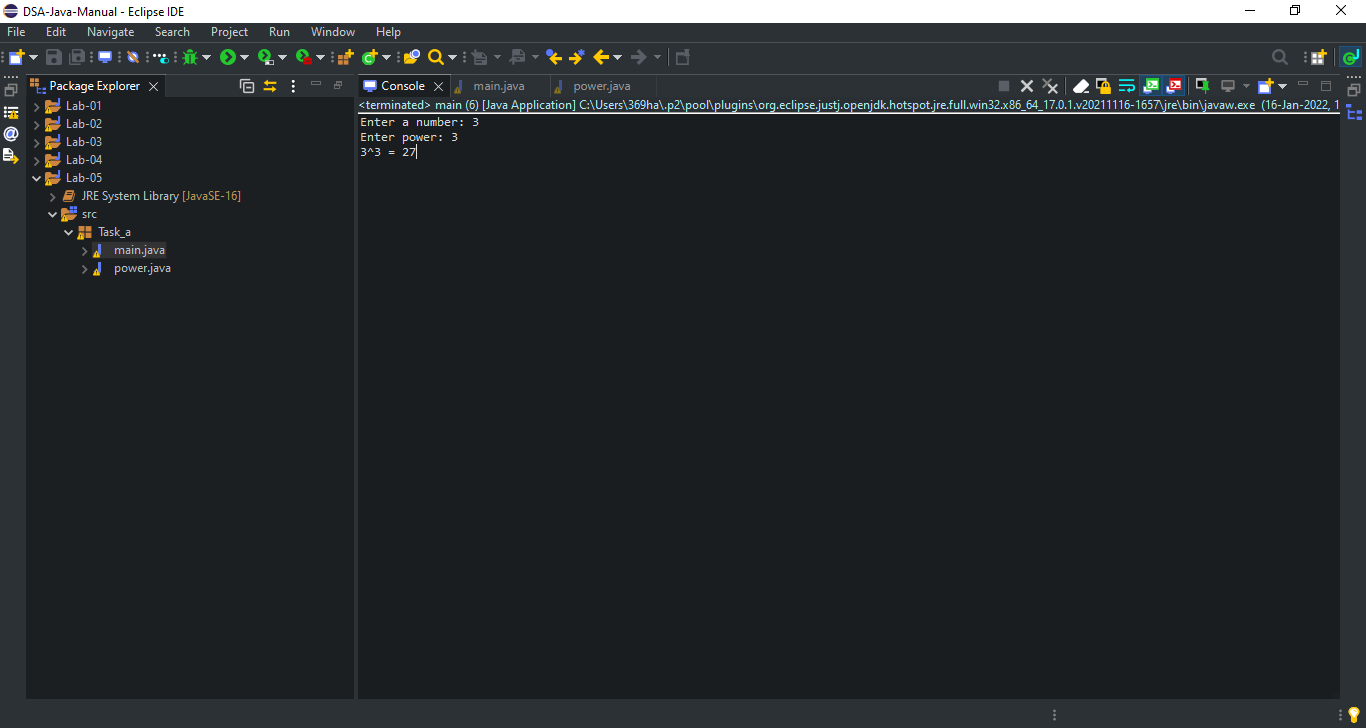
**main.java:**



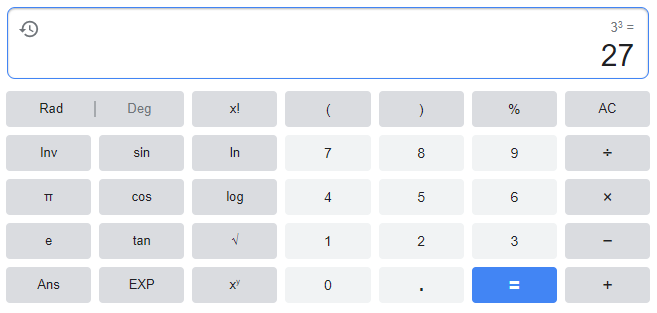
**power.java:**



**Code Output Screen Shot:**



**Calculator: (**3^3 = 27**)**



**(b). Reverse a String using recursion in Java**

**Code:**

**main.java:**

package Task\_b;

import java.util.Scanner;

public class main {

public static void main(String[] args) {

reverse r = new reverse();

Scanner scan = new Scanner(System.in);

System.out.print("Enter String:");

String str = scan.nextLine();

String rev = r.reverse(str);

System.out.print("Reverse String is: "+rev);

}

}

**reverse.java:**

package Task\_b;

public class reverse {

public static String reverse(String s) {

if (s.isEmpty())

return s;

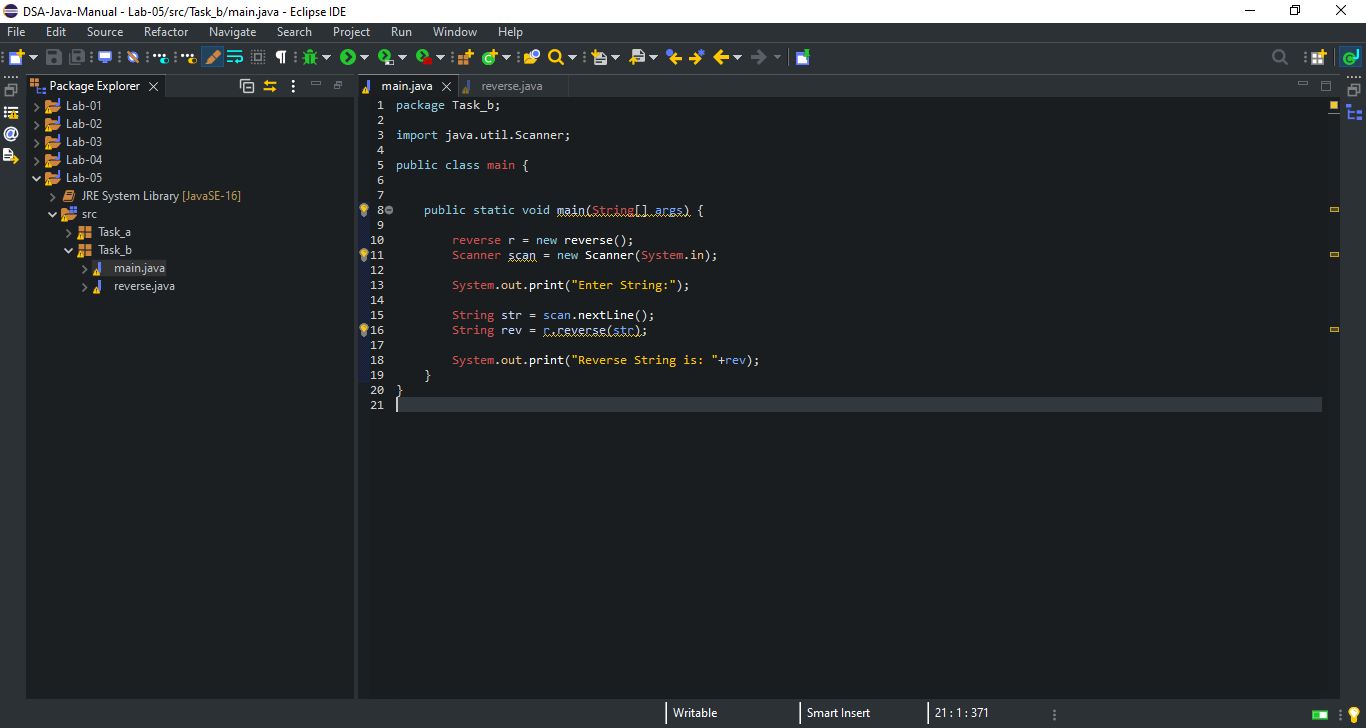
return reverse(s.substring(1)) + s.charAt(0);

}

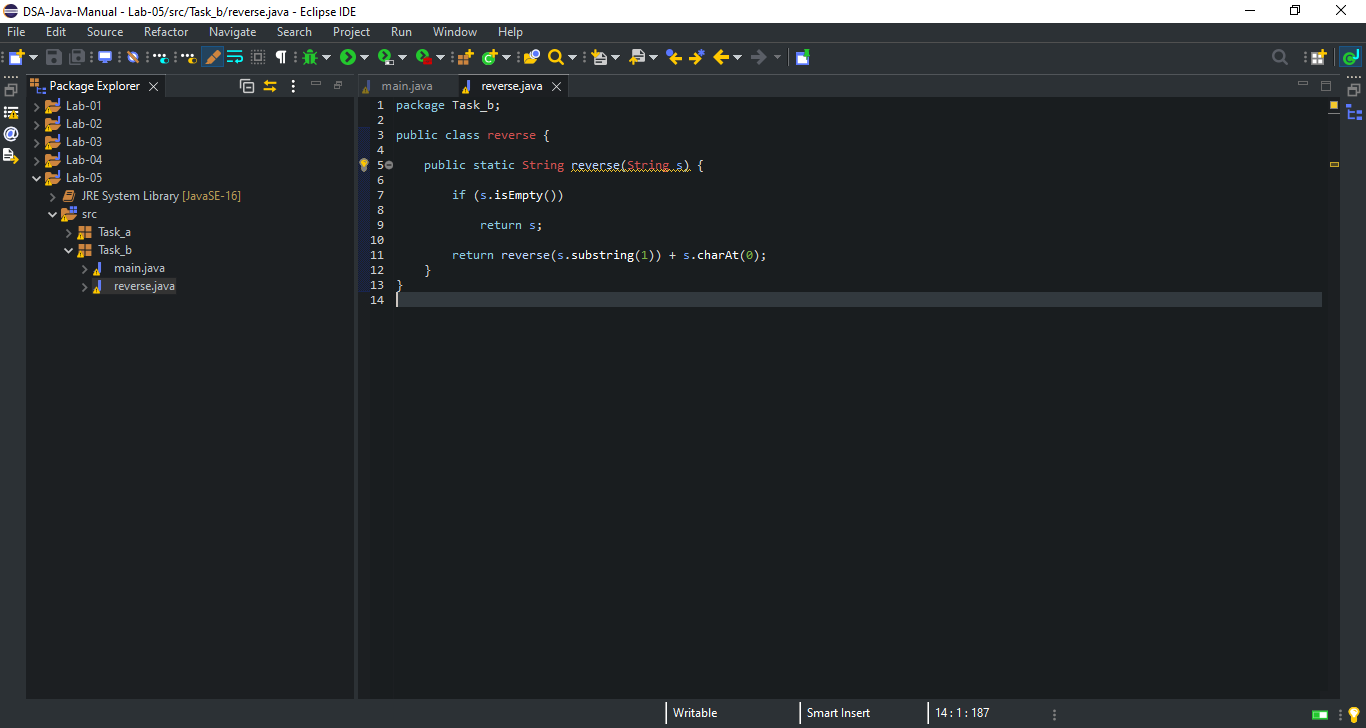
}

**Code Screen Shot:**

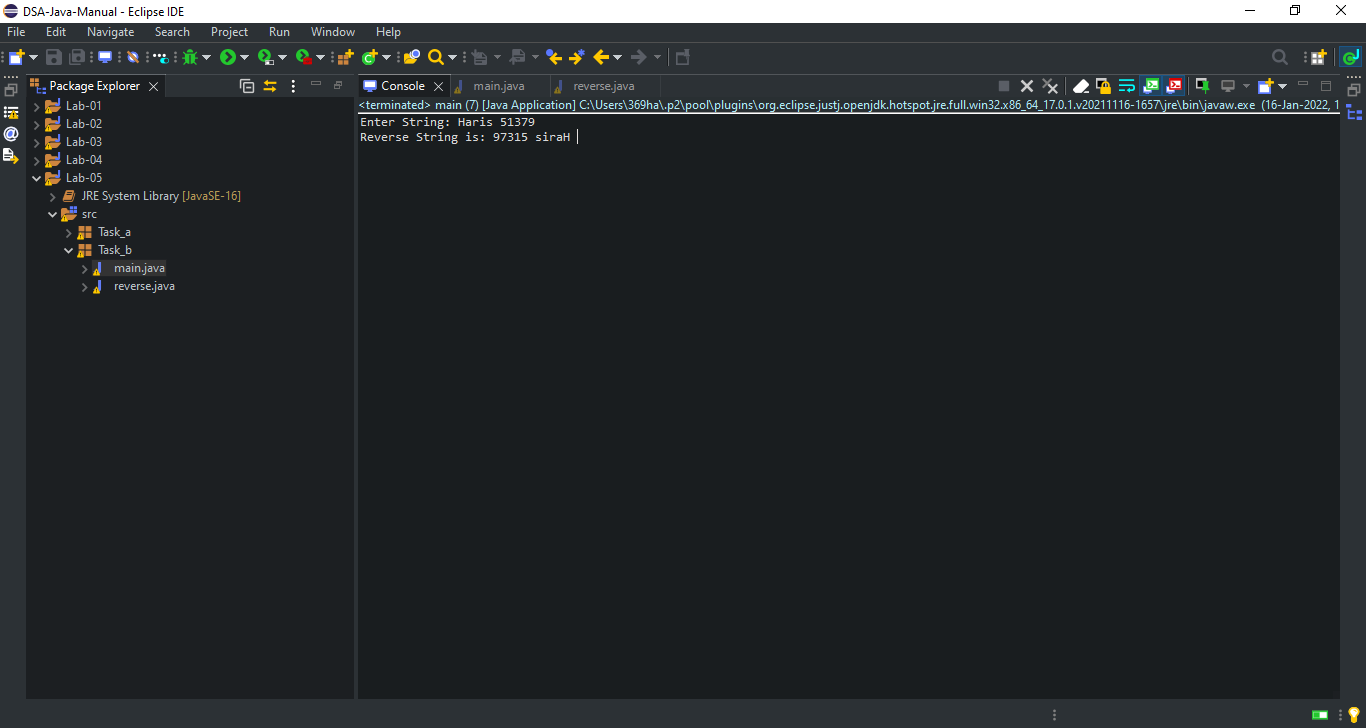
**main.java:**



**reverse.java:**



**Code Output Screen Shot:**



**Lab: 06**

**Lab Tasks:**

**Implement Stack using Linked List.**

**Code:**

**main.java:**

package Task;

public class main {

public static void main(String args[]) {

stack\_using\_linkedList lls = new stack\_using\_linkedList();

lls.push(20);

lls.push(50);

lls.push(80);

lls.push(40);

lls.push(60);

lls.push(75);

System.out.println("Element removed from LinkedList: "+lls.pop());

System.out.println("Element removed from LinkedList: "+lls.pop());

lls.push(10);

System.out.println("Element removed from LinkedList: "+lls.pop());

lls.printList(lls.head);

}

}

**stack\_using\_linkedList.java:**

package Task;

public class stack\_using\_linkedList {

public Node head;

public class Node {

int value;

Node next;

}

public stack\_using\_linkedList() {

head = null;

}

public int pop() throws LinkedListEmptyException {

if (head == null) {

throw new LinkedListEmptyException();

}

int value = head.value;

head = head.next;

return value;

}

public void push(int value) {

Node oldHead = head;

head = new Node();

head.value = value;

head.next = oldHead;

}

public static void printList(Node head) {

Node temp = head;

while (temp != null) {

System.out.format("%d ", temp.value);

temp = temp.next;

}

System.out.println();

}

}

**LinkedListisEmptyException.java:**

package Task;

public class LinkedListEmptyException extends RuntimeException {

private static final long serialVersionUID = 1L;

public LinkedListEmptyException() {

super();

}

public LinkedListEmptyException(String message) {

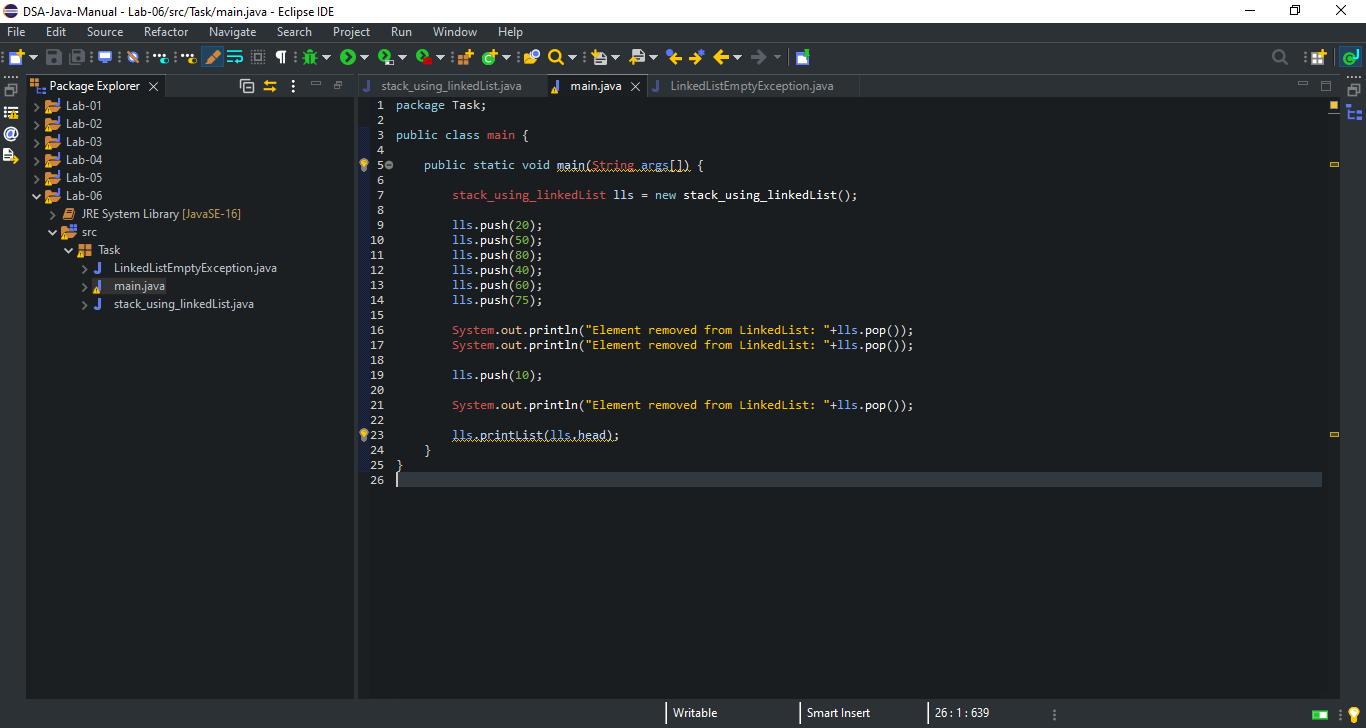
super(message);

}

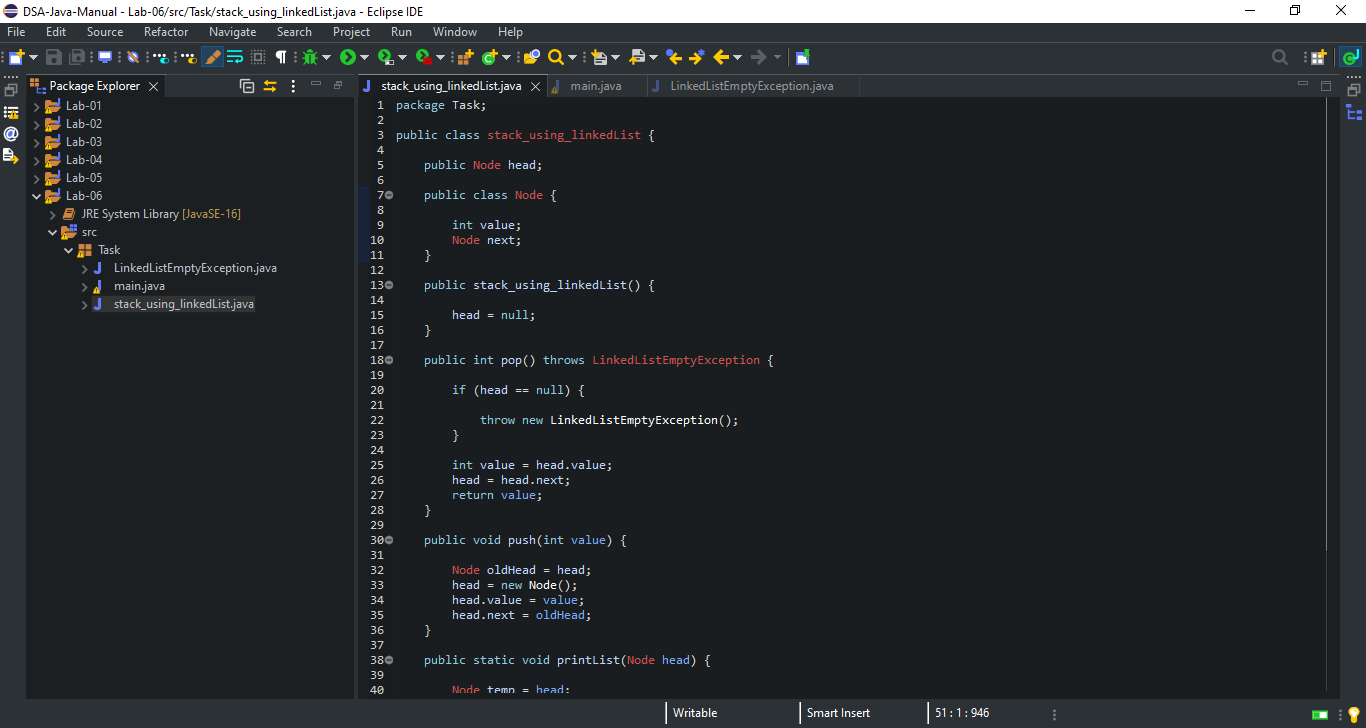
}

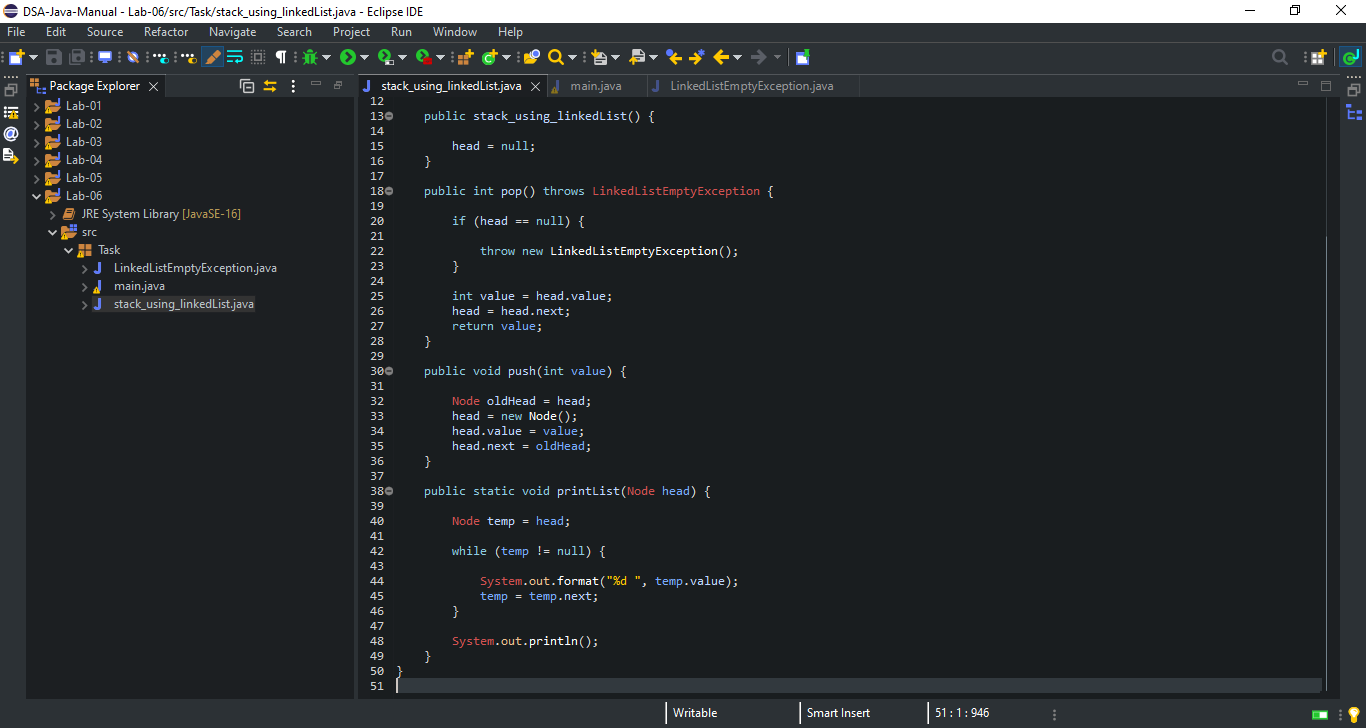
**Code Screen Shot:**

**main.java:**

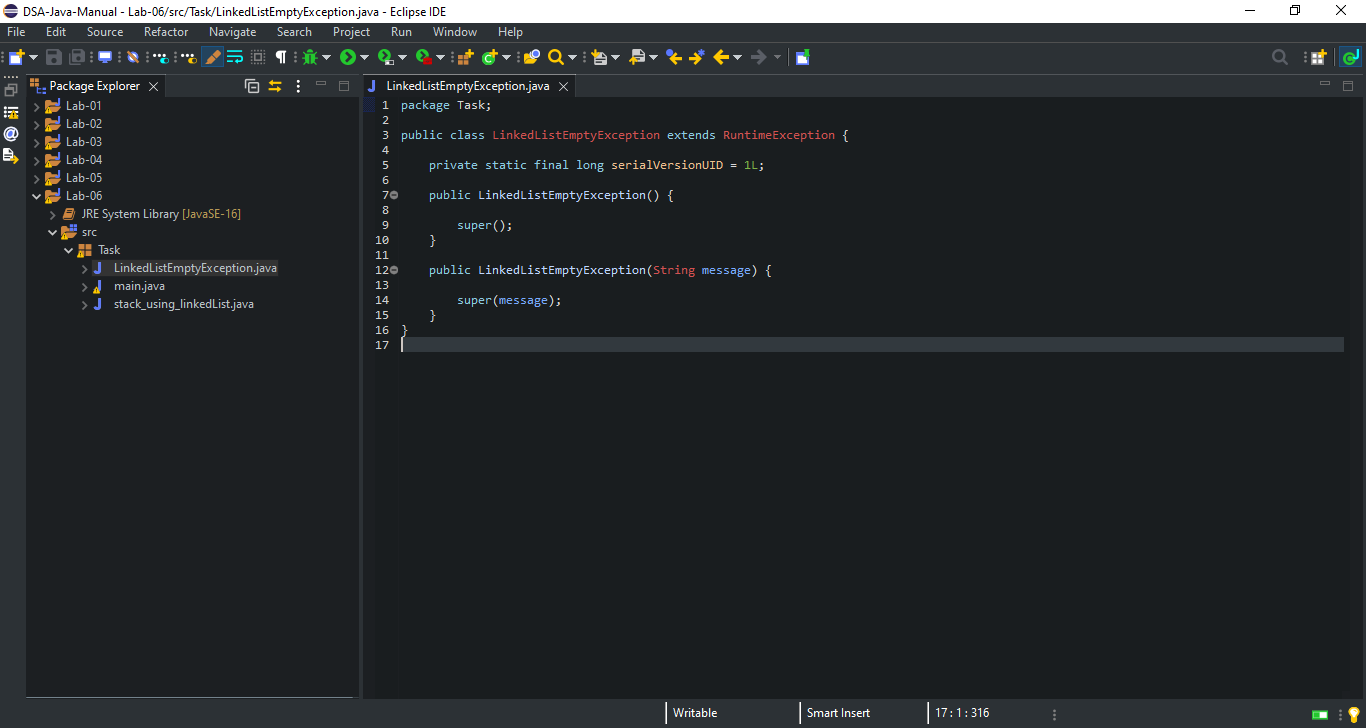


**Stack\_using\_linkedList.java:**

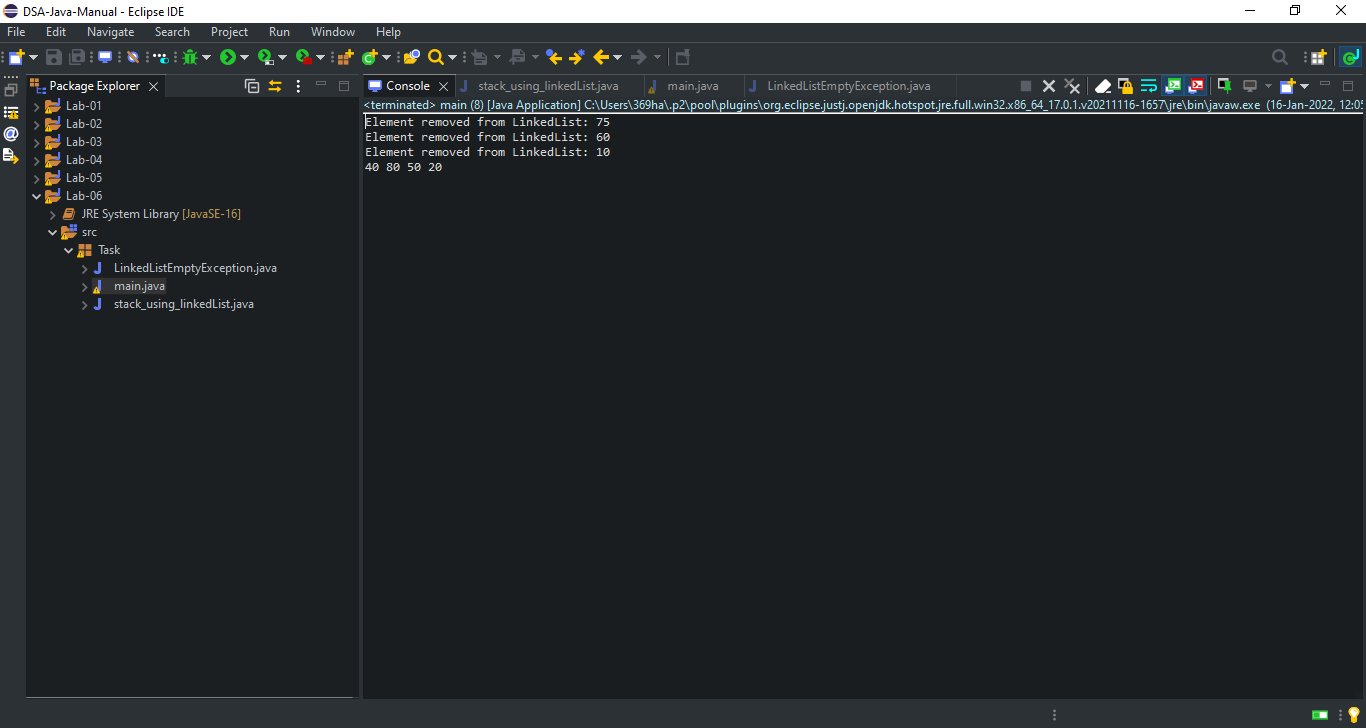




**LinkedListEmptyException.java:**



**Code Output Screen Shot:**



**Lab: 07**

**Task 1:**

**Write a java program to sort the array by using bubble sort and selection sort.**

**Code:**

package Task;

public class main {

public static void main(String[] args) {

BubbleSort bs = new BubbleSort();

int arr[] ={3,60,35,2,45,320,5};

System.out.println("\nArray Before Bubble Sort\n");

for(int i=0; i < arr.length; i++) {

System.out.print(arr[i] + " ");

}

System.out.println();

bs.bubbleSort(arr);

System.out.println("\nArray After Bubble Sort\n");

for(int i=0; i < arr.length; i++) {

System.out.print(arr[i] + " ");

}

}

}

**BubbleSort.java:**

package Task;

public class BubbleSort {

public static void bubbleSort(int[] arr) {

int n = arr.length;

int temp = 0;

for(int i=0; i < n; i++) {

for(int j=1; j < (n-i); j++) {

if(arr[j-1] > arr[j]) {

temp = arr[j-1];

arr[j-1] = arr[j];

arr[j] = temp;

}

}

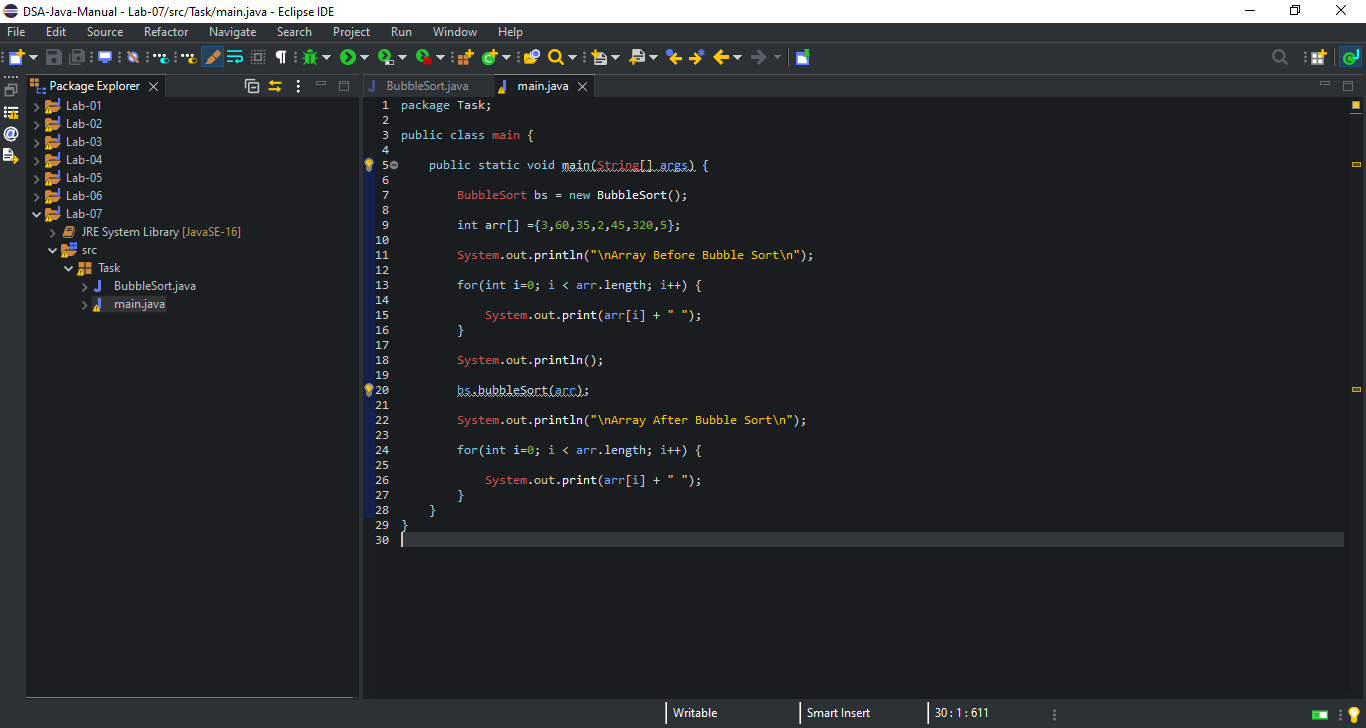
}

}

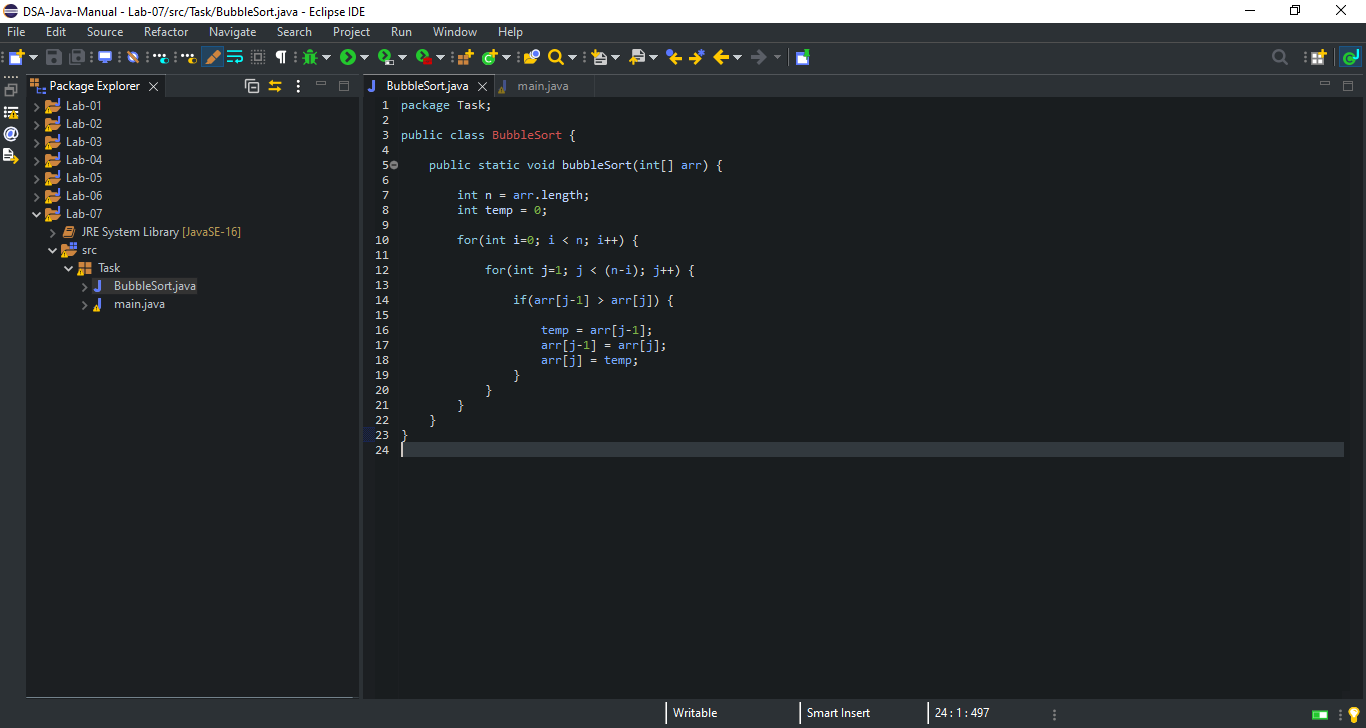
}

**Code Screen Shot:**

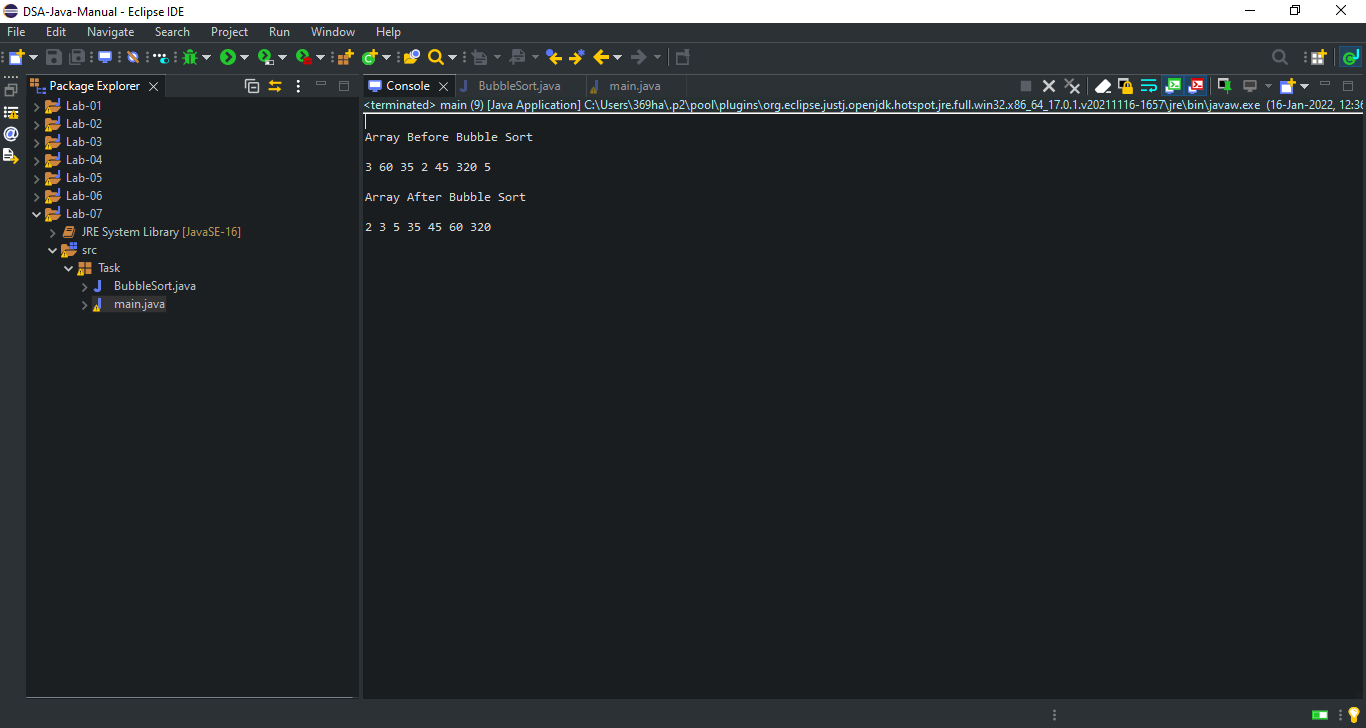
**main.java:**



**BubbleSort.java:**



**Code Output Screen Shot:**



**Lab: 08**

* **No Task & Assignment in Lab-08.**

**Lab: 09**

#### **Task # 1:**

#### **Implement Pre-order traversal in java.**

**Code:**

**main.java:**

package Task;

public class main {

public static void main(String[] args) {

Tree tree = new Tree();

tree.root = new Node(1);

tree.root.left = new Node(12);

tree.root.right = new Node(9);

tree.root.left.left = new Node(5);

tree.root.left.right = new Node(6);

System.out.println("\nPreorder traversal ");

tree.preorder(tree.root);

}

}

**Node.java:**

package Task;

public class Node {

int item;

Node left, right;

public Node(int key) {

item = key;

left = right = null;

}

}

**Tree.java:**

package Task;

public class Tree {

Node root;

Tree() {

root = null;

}

void preorder(Node node) {

if (node == null)

return;

System.out.print(node.item + "->");

preorder(node.left);

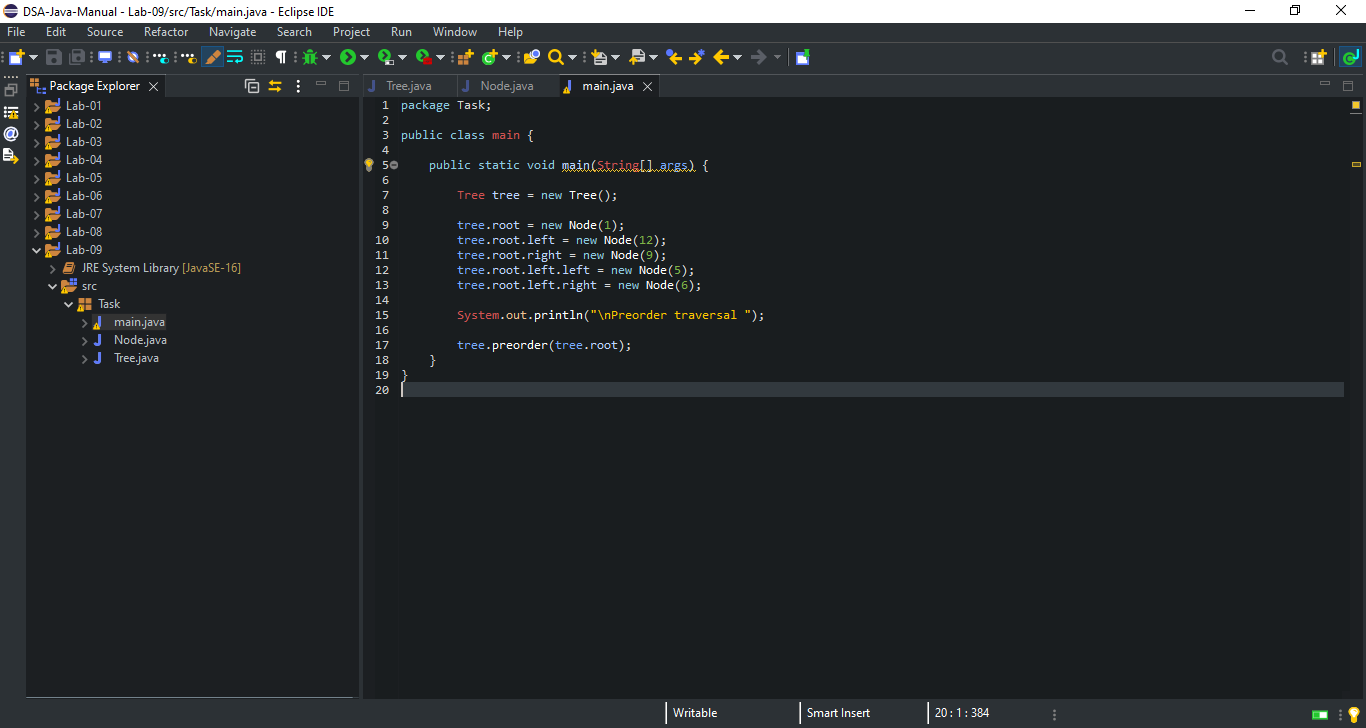
preorder(node.right);

}

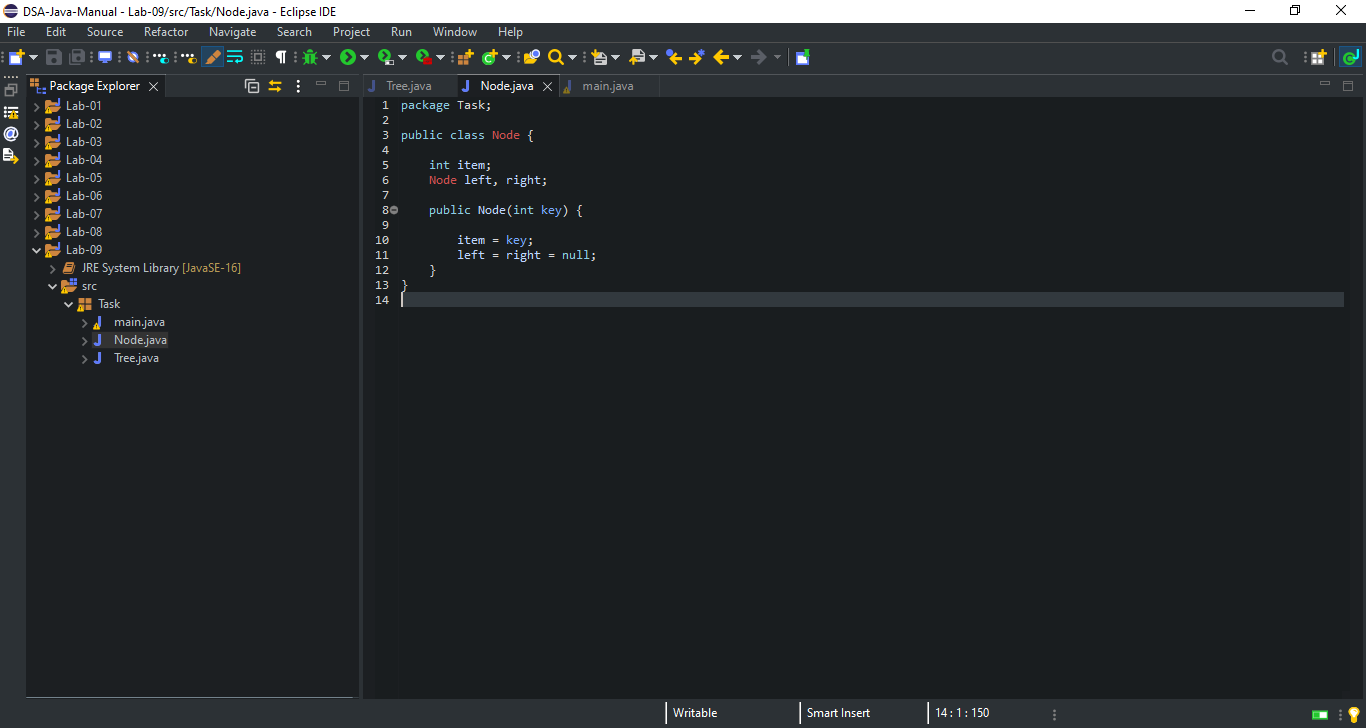
}

**Code Screen Shot:**

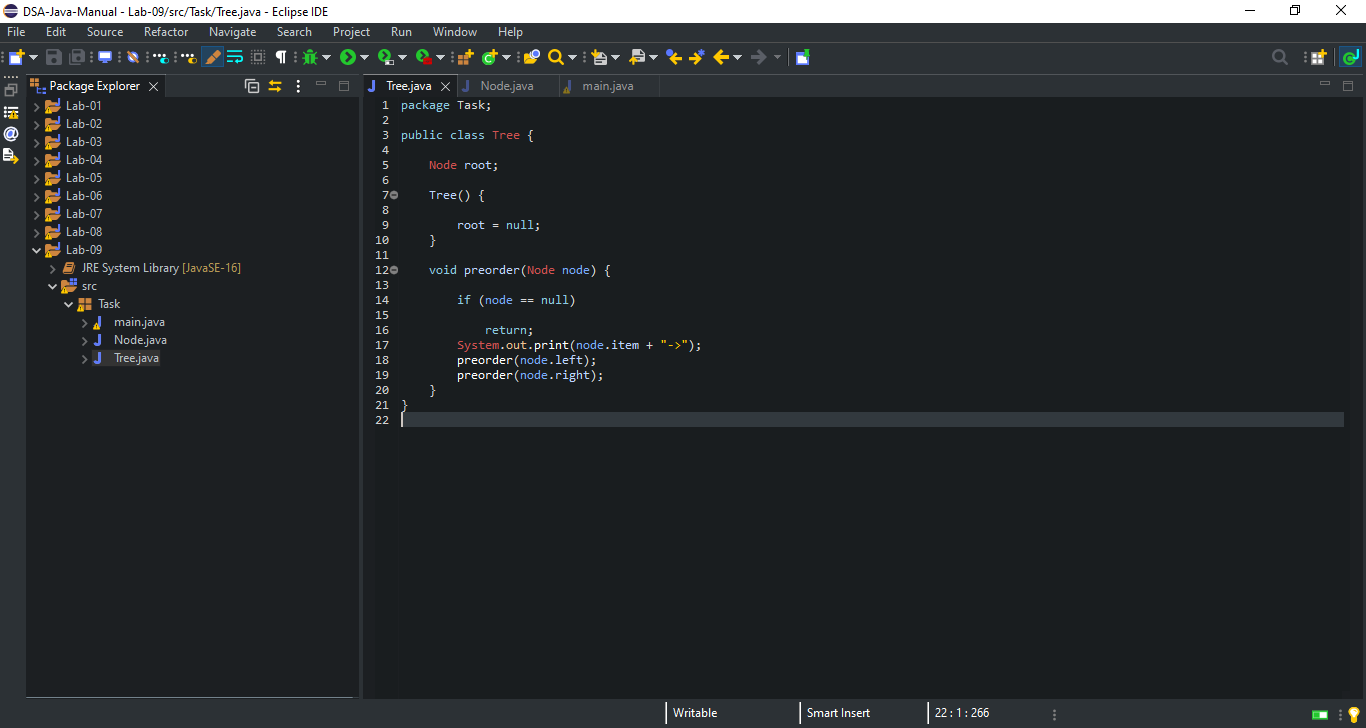
**main.java:**



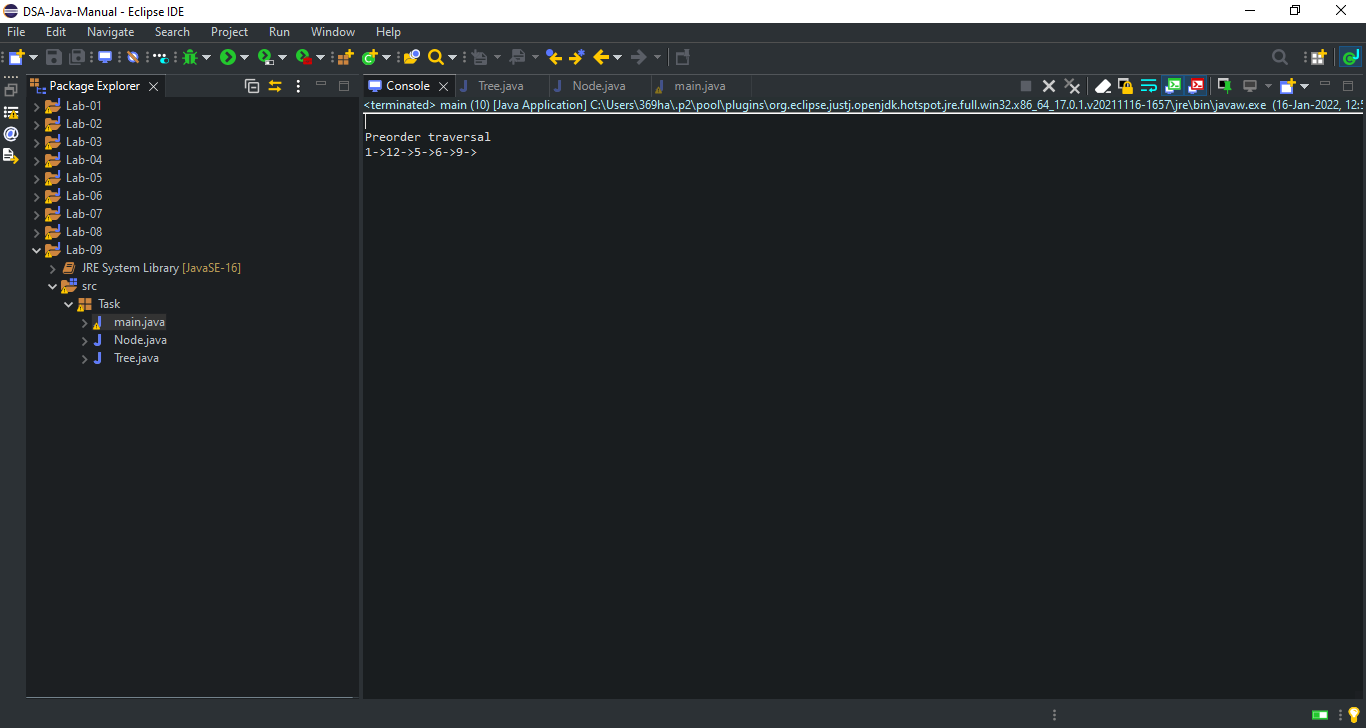
**Node.java:**



**Tree.java:**



**Code Output Screen Shot:**



**Lab: 10**

* **No Task & Assignment in Lab-10.**

**Lab: 11**

**Lab Tasks:**

**Implement a binary search tree and AVL tree from the following traversals:**

**{15, 5, 20, 70, 3, 10, 60, 90, 16}**

* **Write a method that search 15 but your tree should be BST and AVL after deletion**
* **Write a method to Reverse path of Binary Search Tree.**

**Code:**

**main.java:**

package task;

import task.ConvertBTtoBST.Node;

public class main {

public static void main(String[] args) {

ConvertBTtoBST bt = new ConvertBTtoBST();

bt.root = new Node(15);

bt.root.left = new Node(5);

bt.root.right = new Node(20);

bt.root.left.left = new Node(70);

bt.root.left.right = new Node(3);

bt.root.right.left = new Node(10);

bt.root.right.right = new Node(60);

bt.root.right.right.left = new Node(90);

bt.root.right.right.right = new Node(16);

System.out.println("Inorder representation of binary tree: ");

bt.inorderTraversal(bt.root);

Node bst = bt.convertBTBST(bt.root);

System.out.println("\nInorder representation of resulting binary search tree: ");

bt.inorderTraversal(bst);

}

}

**ConvertBTtoBST.java:**

package task;

import java.util.Arrays;

public class ConvertBTtoBST {

public static class Node{

int data;

Node left;

Node right;

public Node(int data){

this.data = data;

this.left = null;

this.right = null;

}

}

public Node root;

int[] treeArray;

int index = 0;

public ConvertBTtoBST() {

root = null;

}

public Node convertBTBST(Node node) {

int treeSize = calculateSize(node);

treeArray = new int[treeSize];

convertBTtoArray(node);

Arrays.sort(treeArray);

Node d = createBST(0, treeArray.length -1);

return d;

}

public int calculateSize(Node node) {

int size = 0;

if (node == null)

return 0;

else {

size = calculateSize (node.left) + calculateSize (node.right) + 1;

return size;

}

}

public void convertBTtoArray(Node node) {

if(root == null) {

System.out.println("Tree is empty");

return;

}

else {

if(node.left != null)

convertBTtoArray(node.left);

treeArray[index] = node.data;

index++;

if(node.right != null)

convertBTtoArray(node.right);

}

}

public Node createBST(int start, int end) {

if (start>end) {

return null;

}

int mid = (start + end) / 2;

Node node = new Node(treeArray[mid]);

node.left = createBST(start, mid - 1);

node.right = createBST(mid + 1, end);

return node;

}

public void inorderTraversal(Node node) {

if(root == null) {

System.out.println("Tree is empty");

return;

}

else {

if(node.left!= null)

inorderTraversal(node.left);

System.out.print(node.data + " ");

if(node.right!= null)

inorderTraversal(node.right);

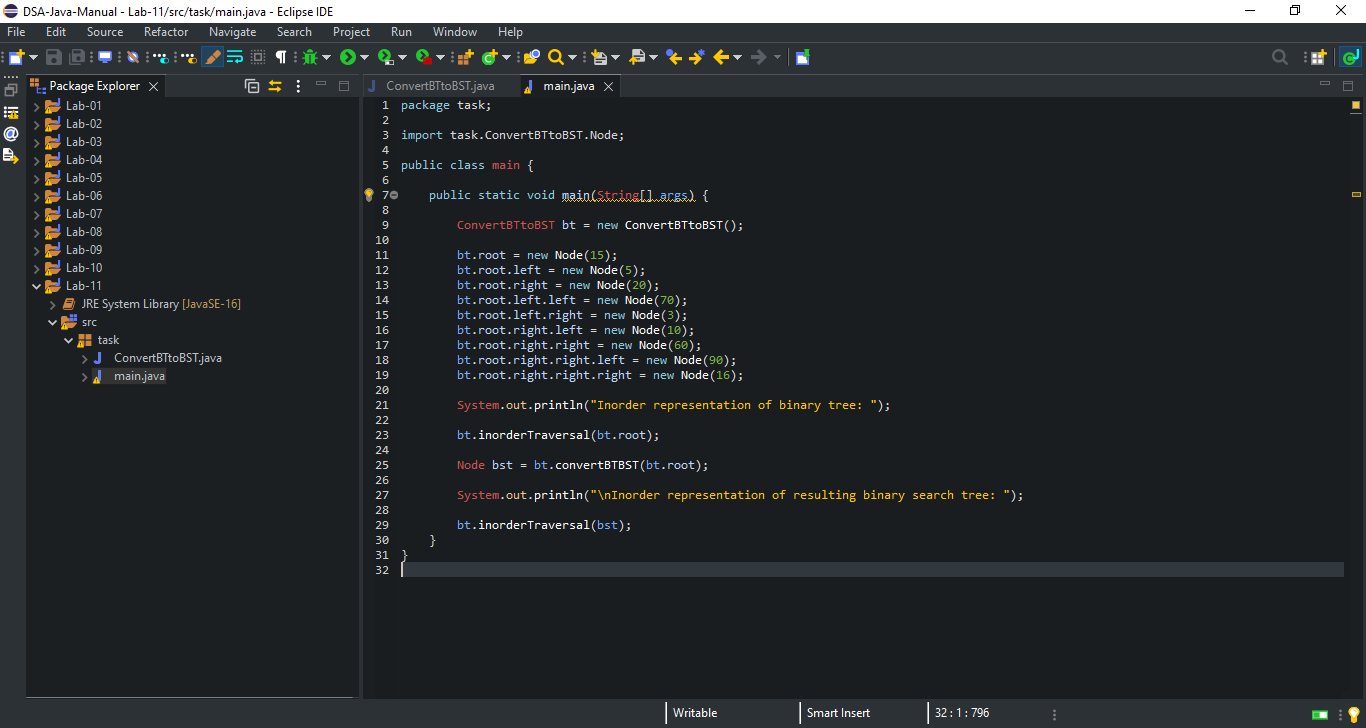
}

}

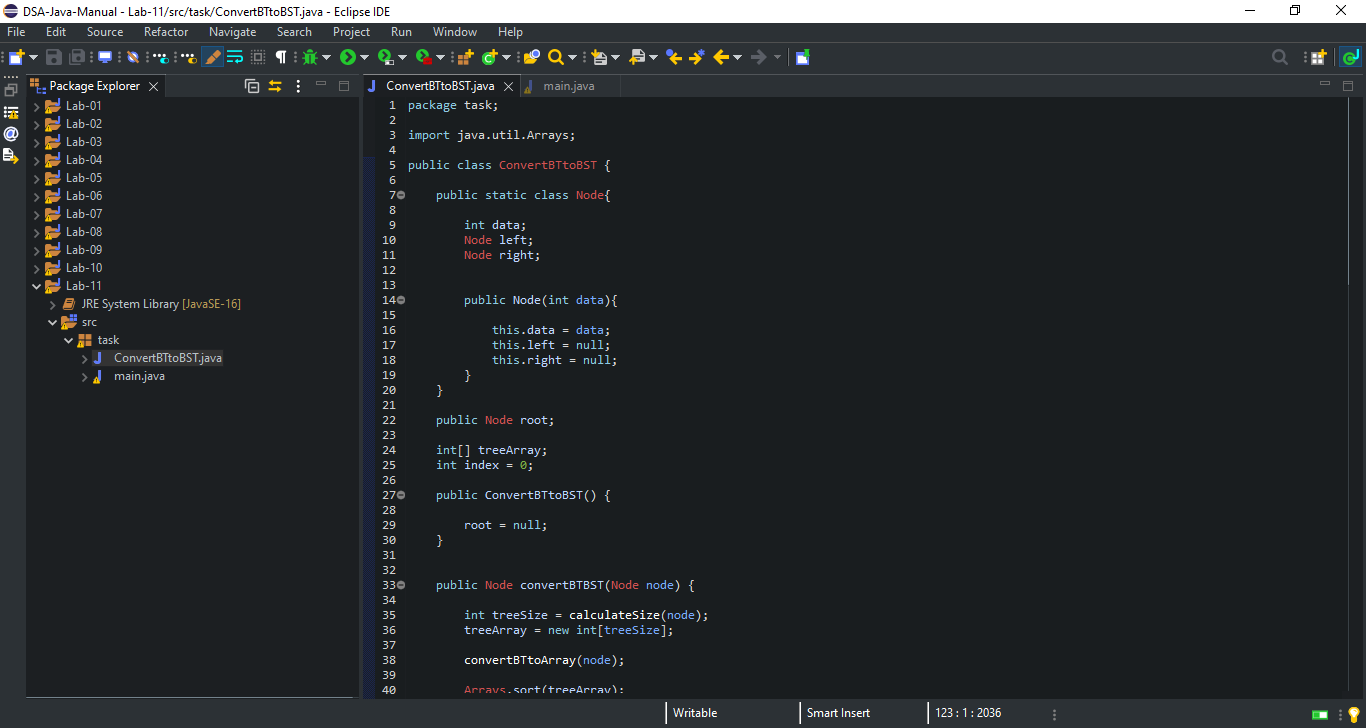
}

**Code Screen Shot:**

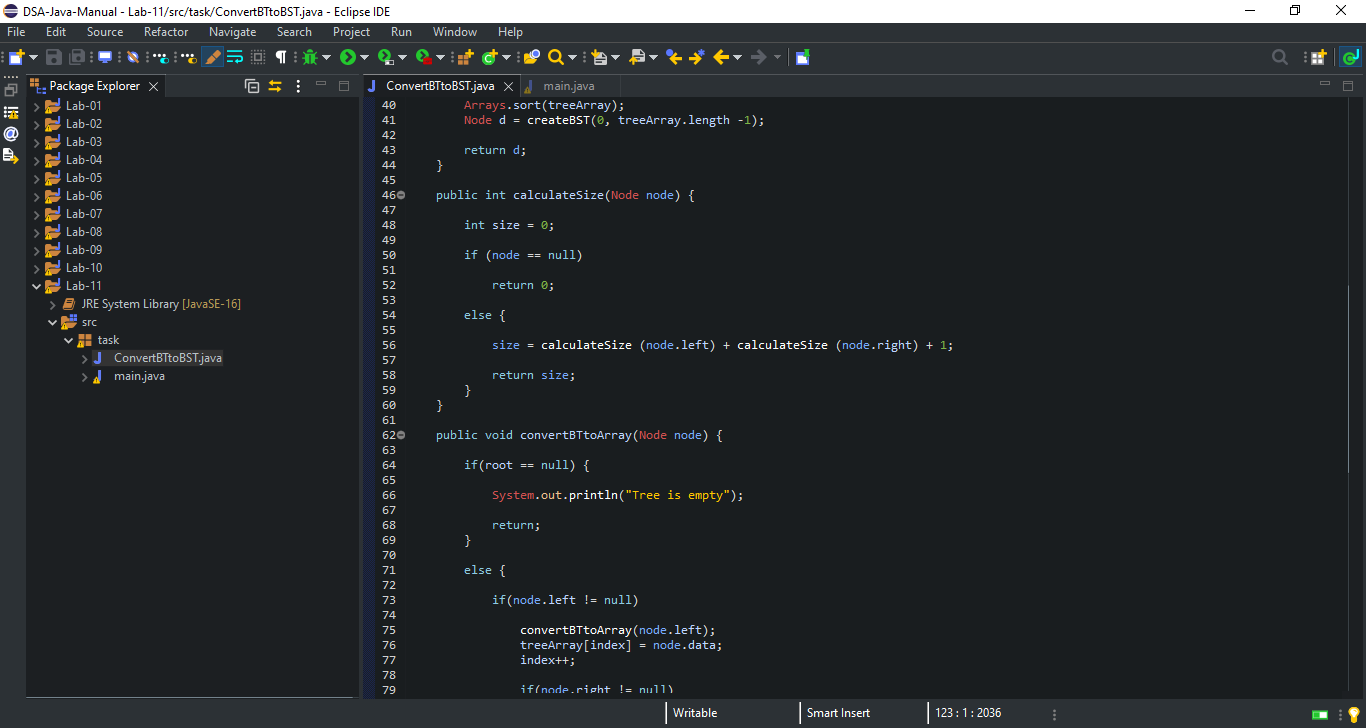
**main.java:**

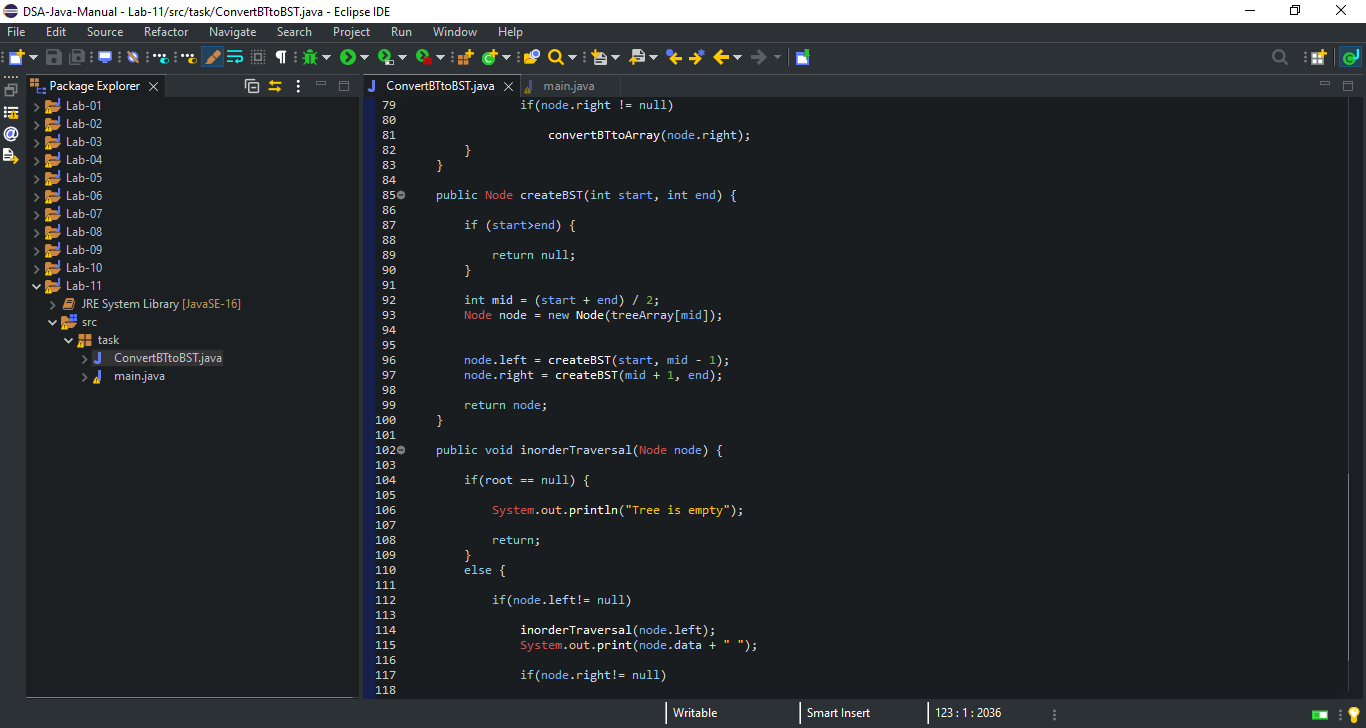


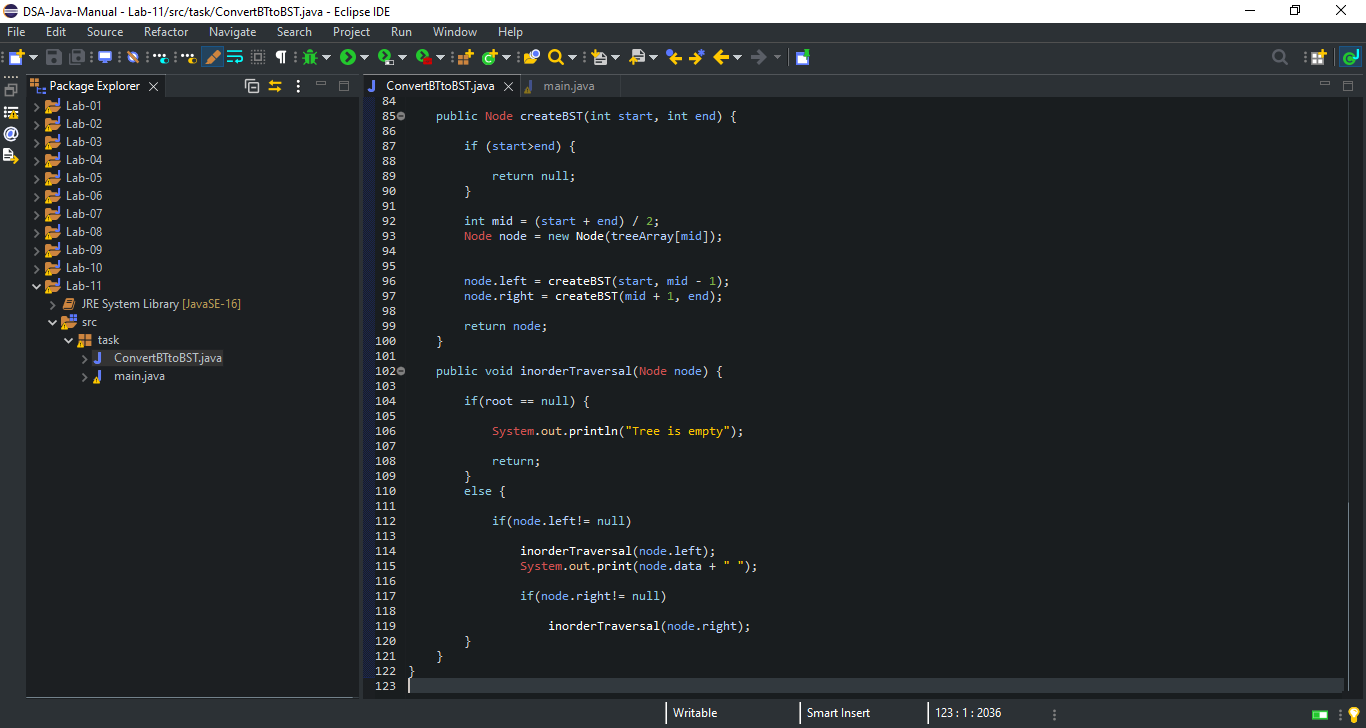
**ConvertBTtoBST.java:**



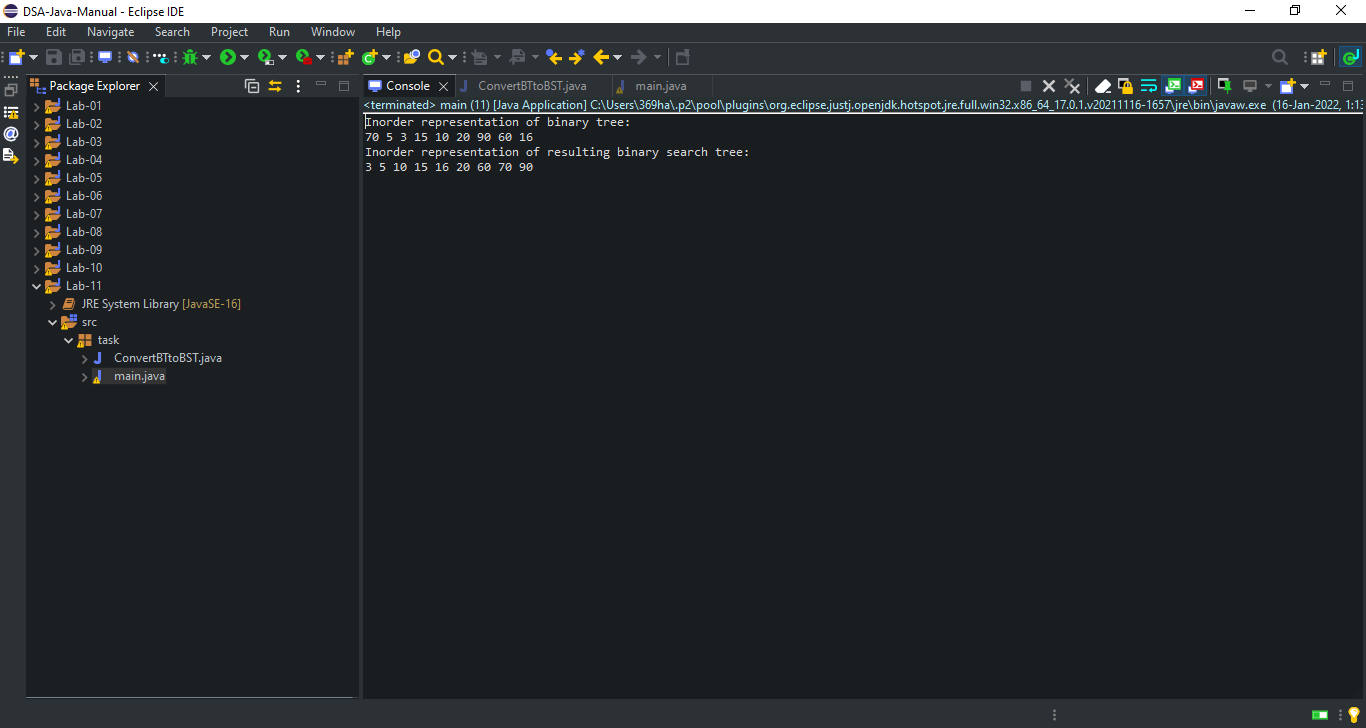
**Continue…**







**Code Output Screen Shot:**



**Lab: 12**

**Lab Tasks:**

* **Implement a method *contains(s)* that returns a boolean value that checks whether the string s is in the table.**
* **Implement a method size*()* that returns the number of strings in the table.**

**Code:**

**HashTable.java:**

package Task;

public class HashTable {

static private class ListNode {

Object key;

Object value;

ListNode next;

}

private ListNode[] table;

private int count;

public HashTable() {

table = new ListNode[64];

}

public HashTable(int initialSize) {

table = new ListNode[initialSize];

}

void dump() {

System.out.println();

for (int i = 0; i <table.length; i++) {

System.out.print(i + ":");

ListNode list = table[i];

while (list != null) {

System.out.print(" (" + list.key + "," + list.value + ")");

list = list.next;

}

System.out.println();

}

}

public void put(Object key, Object value) {

int bucket = hash(key);

ListNode list = table[bucket];

while (list != null) {

if (list.key.equals(key))

break;

list = list.next;

}

if (list != null) {

list.value = value;

}

else {

if (count >= 0.75\*table.length) {

resize();

}

ListNode newNode = new ListNode();

newNode.key = key;

newNode.value = value;

newNode.next = table[bucket];

table[bucket] = newNode;

count++;

}

}

public Object get(Object key) {

int bucket = hash(key);

ListNode list = table[bucket];

while (list != null) {

if (list.key.equals(key))

return list.value;

list = list.next;

}

return null;

}

public void remove(Object key) {

int bucket = hash(key);

if (table[bucket] == null) {

return;

}

if (table[bucket].key.equals(key)) {

table[bucket] = table[bucket].next;

count--;

return;

}

ListNode prev = table[bucket];

ListNode curr = prev.next;

while (curr != null&& ! curr.key.equals(key)) {

curr = curr.next;

prev = curr;

}

if (curr != null) {

prev.next = curr.next;

count--;

}

}

public boolean containsKey(Object key) {

int bucket = hash(key);

ListNode list = table[bucket];

while (list != null) {

if (list.key.equals(key))

return true;

list = list.next;

}

return false;

}

public int size() {

return count;

}

private int hash(Object key) {

return (Math.abs(key.hashCode())) % table.length;

}

private void resize() {

ListNode[] newtable = new ListNode[table.length\*2];

for (int i = 0; i <table.length; i++) {

ListNode list = table[i];

while (list != null) {

ListNode next = list.next;

int hash = (Math.abs(list.key.hashCode())) % newtable.length;

list.next = newtable[hash];

newtable[hash] = list;

list = next;

}

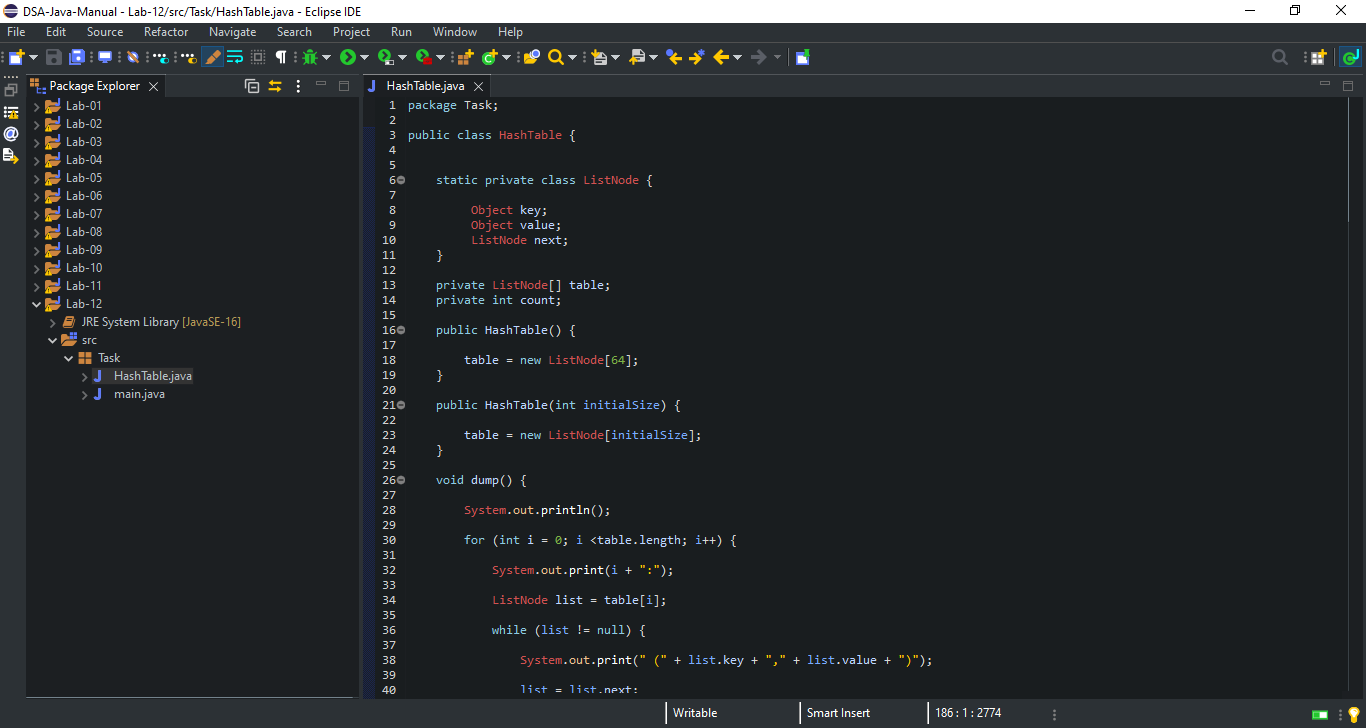
}

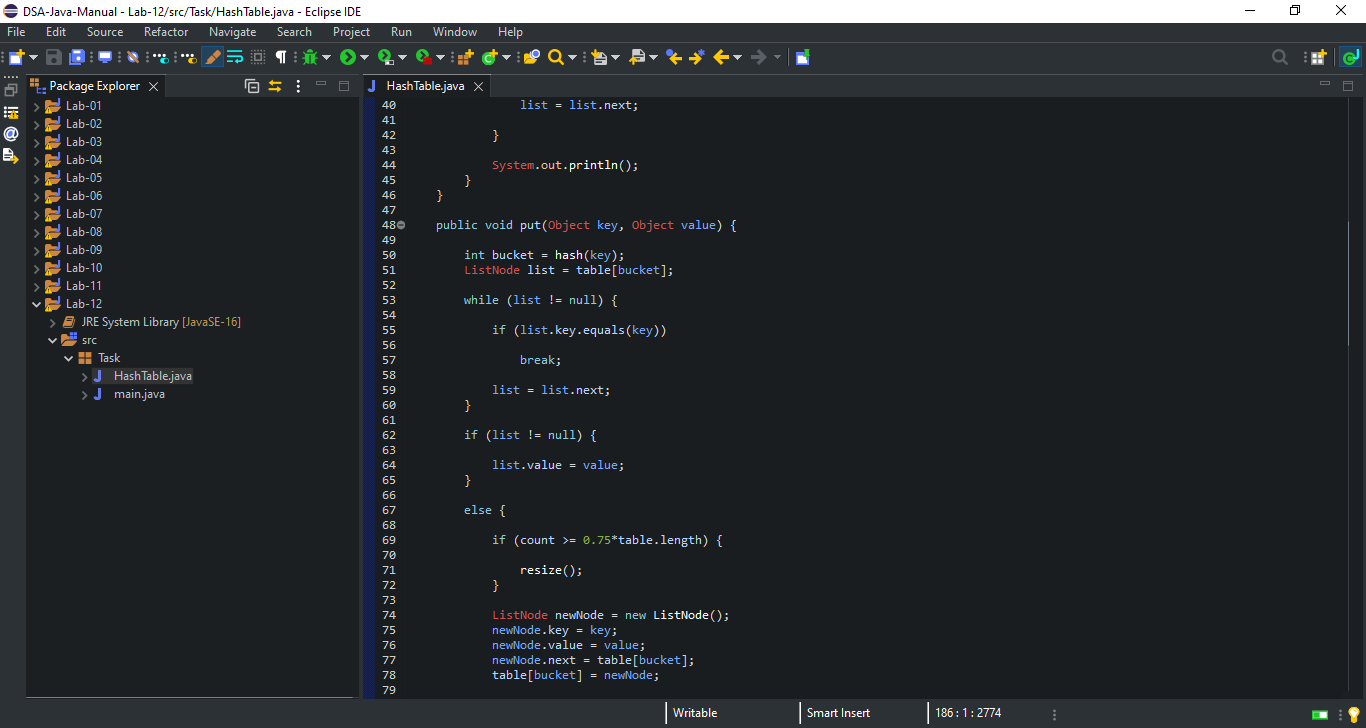
table = newtable;

}

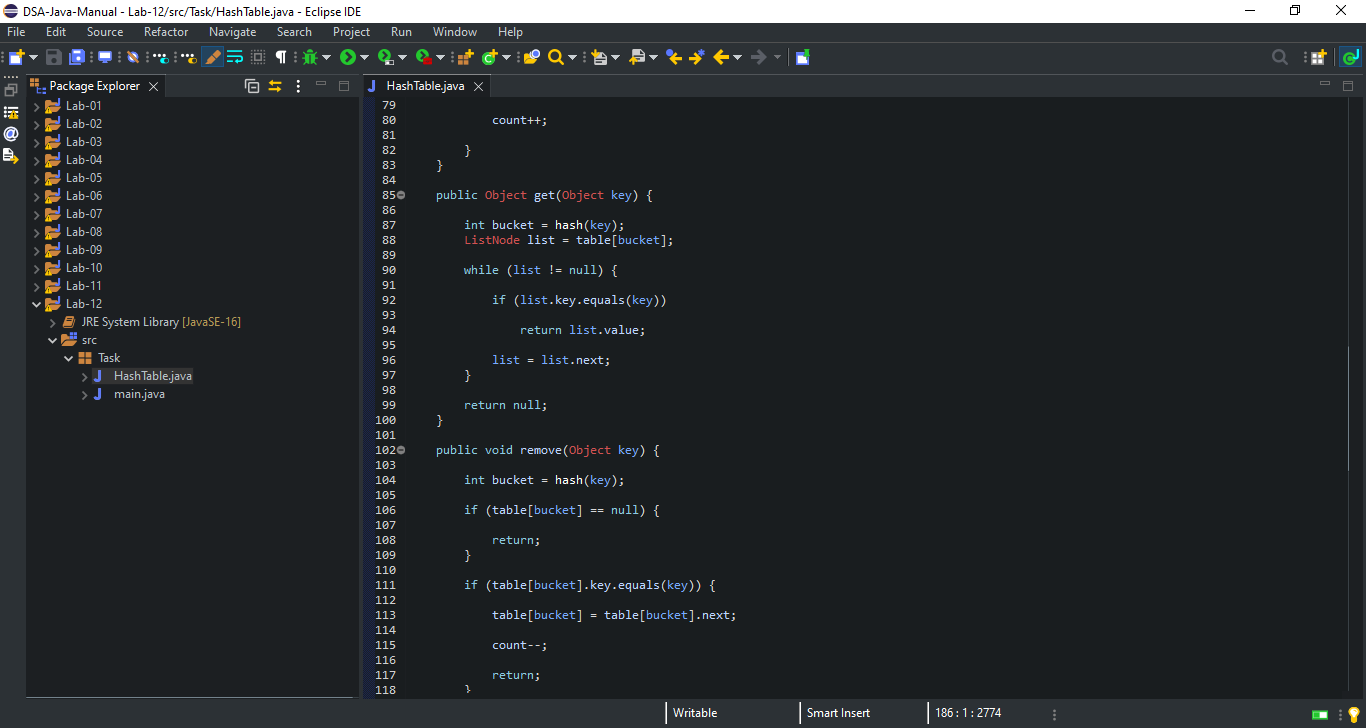
}

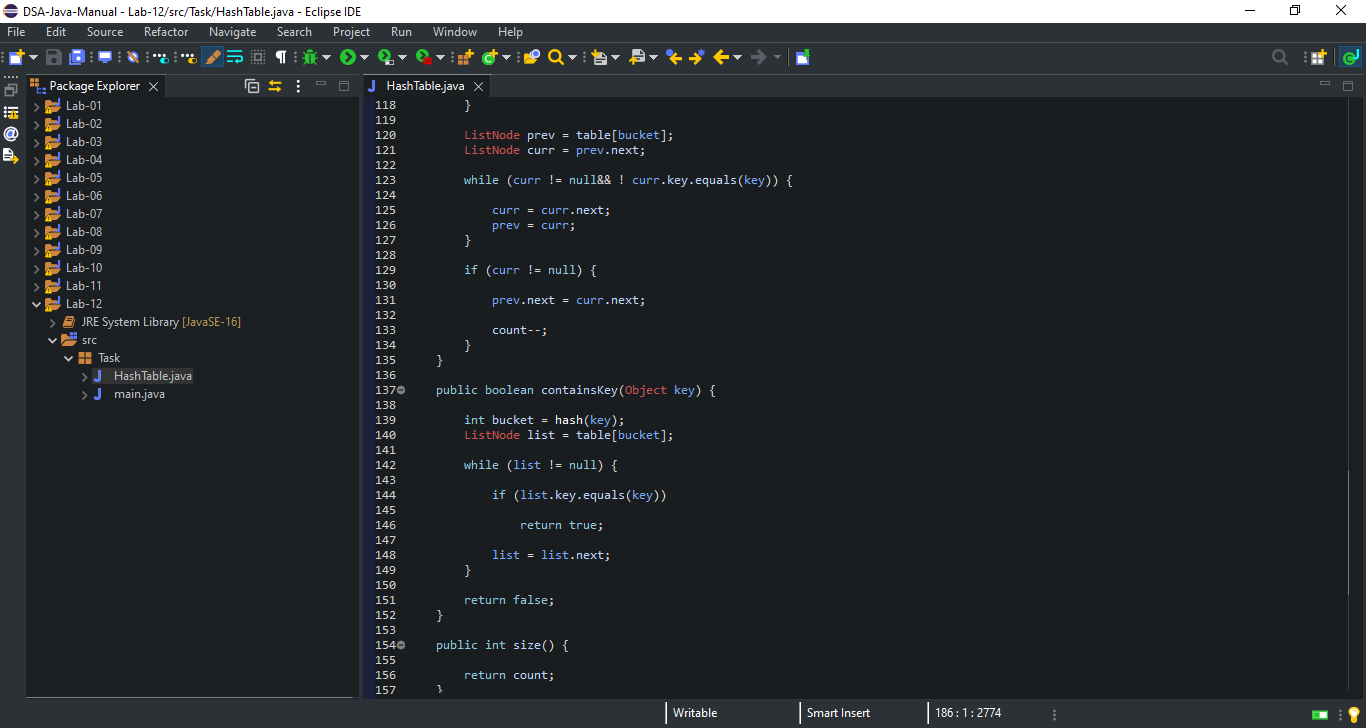
**Code Screen Shot:**

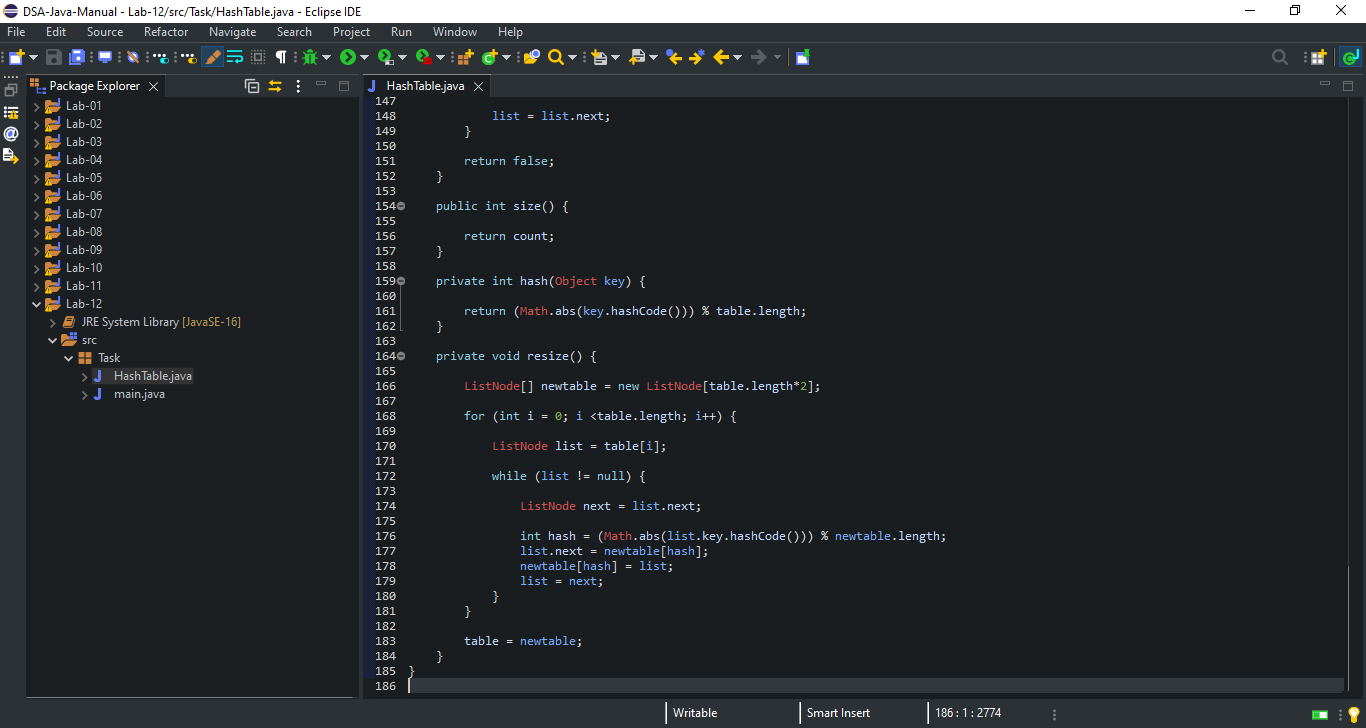




**Continue…**







**Lab: 13**

**Lab Tasks:**

* **Implement a method *remove(s)* that removes a value from hashmap.**
* **Implement a method isempty*()* that returns a boolean value a that checks whether the hash map is empty or not.**
* **Implement a method clear*()* that removes all values from the hashmap.**

**Code:**

**Hashing.java:**

package Task;

import java.util.\*;

public class Hashing {

public static void main(String[] args) {

HashSet<Integer>A = new HashSet<Integer>();

A.add(1);

A.add(2);

A.add(3);

System.out.println(A);

if(A.isEmpty()) {

System.out.println("HashSet is empty");

}

else {

System.out.println("HashSet is not empty");

}

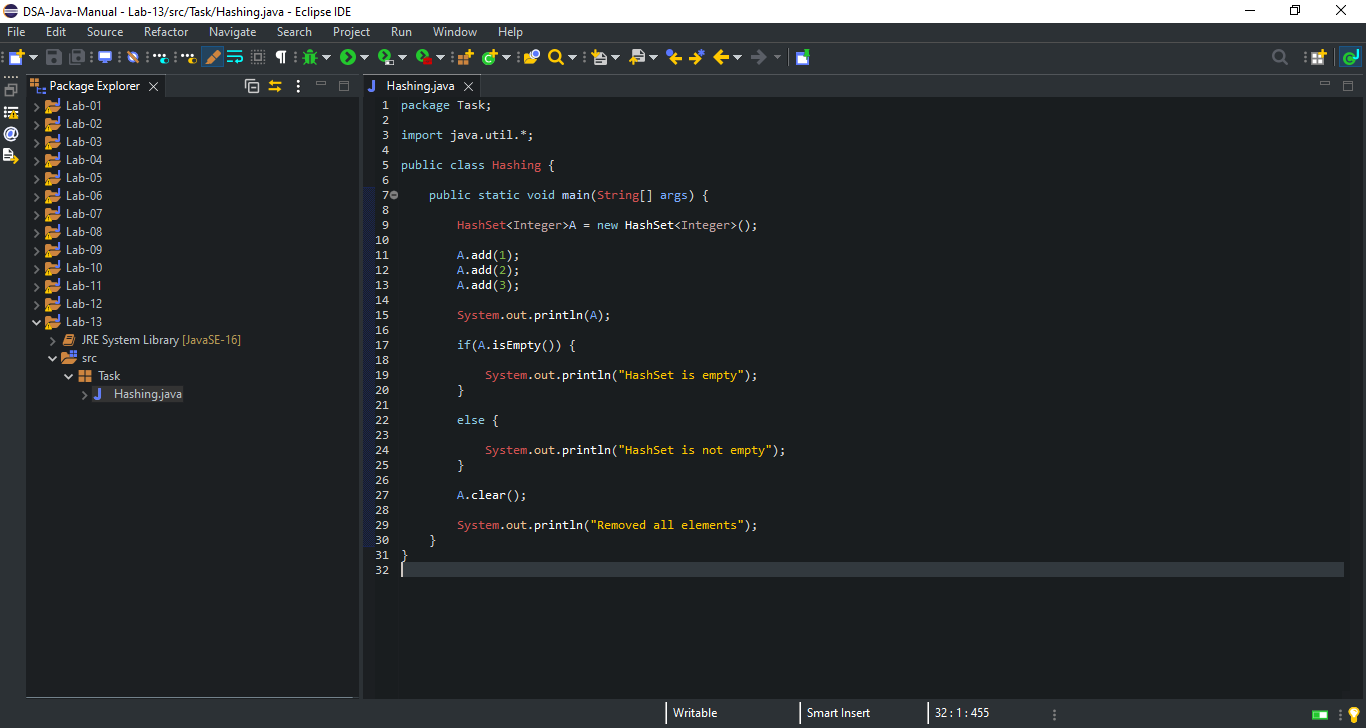
A.clear();

System.out.println("Removed all elements");

}

}

**Code Screen Shot:**



**Code Output Screen Shot:**



**Lab: 14**

* **No Task & Assignment in Lab-14.**