Lab 2A

Truy cập trang sau:

<https://www.jdoodle.com/online-java-compiler/>

<https://www.tutorialspoint.com/compile_java_online.php>

Thực hiện kiểm thử các chương trình Java để:

* Cho biết lỗi gì xảy ra (có thể giải thích)
* Tô màu dòng lệnh lỗi (tô nền vàng)
* Sửa lỗi xảy ra, nếu có (tô chữ đỏ).

**BT1/.**

public class Main {

public static void main(String[] args) {

// Create a Thread object

Thread t = new Thread(Main::print);

// Start the thread

t.star ();

t.start ();

}

public static void print() {

for (int i = 1; i <= 5; i++) {

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

}

}

}

**BT2/.**

import java.time.LocalDate;

import java.time.LocalTime;

import java.time.LocalDateTime;

import java.time.ZonedDateTime;

import static java.time.Month.JANUARY;

public class Main {

public static void main(String[] args) {

// Get current date, time, and datetime

LocalDate dateOnly = LocalDate.now();

LocalTime timeOnly = LocalTime.now();

LocalDateTime dateTime = LocalDateTime.now();

ZonedDateTime dateTimeWithZone = ZonedDateTime.now();

System.out.println("Current Date: " + dateOnly);

System.out.println("Current Time: " + timeOnly);

System.out.println("Current Date and Time: " + dateTime);

System.out.println("Current Date, Time, and Zone: " + dateTimeWithZone);

// Construct a birth date and time from date-time components

LocalDate myBirthDate = LocalDate.of(2000, JANUARY, 31);

LocalTime myBirthTime = LocalTime.of(2, 67);

LocalTime myBirthTime = LocalTime.of(2, 59);

System.out.println("My Birth Date: " + myBirthDate);

System.out.println("My Birth Time:: " + myBirthTime);

}

}

**BT3/.**

import java.text.DateFormatSymbols;

import java.time.Clock;

import java.time.LocalDate;

import java.util.Calendar;

public class Main {

public static void main(String[] args) {

LocalDate date = LocalDate.now();

System.out.println("Current Date: " + date);

Clock clock = Clock.systemUCT();

Clock clock = Clock.systemUTC();

date = LocalDate.now(clock);

System.out.println("Date from clock: " + date);

}

}

**BT4/.**

public class StringExample

{ public static void main(String[] args)

{ String s1 = "Computer Science";

int x = 307;

String s2 = s1 + " " + x;

String s3 = s2.substring(10,25);

String s3 = s2.substring(10,17);

String s4 = "is fun";

String s5 = s2 + s4;

System.out.println("s1: " + s1);

System.out.println("s2: " + s2);

System.out.println("s3: " + s3);

System.out.println("s4: " + s4);

System.out.println("s5: " + s5);

//showing effect of precedence

x = 3;

int y = 5;

String s6 = x + y + "total";

String s7 = "total " + x + y;

String s8 = " " + x + y + "total";

System.out.println("s6: " + s6);

System.out.println("s7: " + s7);

System.out.println("s8: " + s8);

}

}

**BT5/.**

import java.util.Random;

class Consumer extends Thread {

private Buffer buffer;

public Consumer(Buffer buffer) {

this.buffer = buffer;

}

public void run() {

int data;

while (true) {

data = buffer.consume();

}

}

}

public class Main {

public static void Main(String[] args)

public static void main(String[] args) {

Buffer buffer = new Buffer();

Producer p = new Producer(buffer);

Consumer c = new Consumer(buffer);

p.start();

c.start();

}

}

class Producer extend Thread

class Producer extends Thread {

private Buffer buffer;

public Producer(Buffer buffer) {

this.buffer = buffer;

}

public void run() {

Random rand = new Random();

while (true) {

int n = rand.nextInt();

buffer.produce(n);

}

}

}

class Buffer {

private int data;

private boolean empty;

public Buffer() {

this.empty = true;

}

public synchronized void produce(int newData) {

while (!this.empty) {

try {

this.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

this.data = newData;

this.empty = false;

this.notify();

System.out.println("Produced:" + newData);

}

public synchronized int consume() {

while (this.empty) {

try {

this.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

this.empty = true;

this.notify();

System.out.println("Consumed:" + data);

return data;

}

}

**BT6/.**

public class Main extends Thread {

public Main(String name) {

super(name);

}

@Override

public void run() {

Thread t = Thread.currentThread();

String threadName = t.getName();

System.out.println("Inside run() method: " + threadName);

}

public static void Main(String[] args)

public static void main(String[] args) {

Main ct1 = new Main("First Thread");

Main ct2 = new Main("Second Thread");

ct1.start();

ct2.start();

Thread t = Thread.currentThread();

String threadName = t.getName();

System.out.println("Inside main() method: " + threadName);

}

}

**BT7/.**

//import java.time.LocalDate;

import java.time.LocalDate;

import java.time.Month;

public class Main {

public static void main(String[] args) {

LocalDate ldt1 = LocalDate.of(2020, Month.NOVEMBER, 20);

LocalDate ldt2 = LocalDate.now();

int comparison = ldt1.compareTo(ldt2);

if (comparison > 0) {

System.out.println(ldt1 + " is after " + ldt2);

} else if (comparison < 0) {

System.out.println(ldt1 + " is before " + ldt2);

} else {//from w w w .j a v a 2s.c o m

System.out.println(ldt1 + " is equal to " + ldt2);

}

if (ldt1.isAfter(ldt2)) {

System.out.println(ldt1 + " is after " + ldt2);

} else if (ldt1.isBefore(ldt2)) {

System.out.println(ldt1 + " is before " + ldt2);

} else if (ldt1.isEqual(ldt2)) {

System.out.println(ldt1 + " is equal to " + ldt2);

}

}

}

**BT8/.**

//import java.text.DateFormatSymbols;

//import java.util.Calendar;

import java.text.DateFormatSymbols;

import java.util.Calendar;

public class Main {

public static void main(String[] args) {

Calendar gCal = Calendar.getInstance();

int month = gCal.get(Calendar.MONTH) + 1;

int day = gCal.get(Calendar.DATE);

int yr = gCal.get(Calendar.YEAR);

String dateStr = month + "/" + day + "/" + yr;

System.out.println(dateStr);

int dayOfWeek = gCal.get(Calendar.DAY\_OF\_WEEK);

System.out.println(dayOfWeek);

int hour = gCal.get(Calendar.HOUR);

int min = gCal.get(Calendar.MINUTE);

int sec = gCal.get(Calendar.SECOND);

System.out.println(hour + ":" + min + ":" + sec);

DateFormatSymbols symbols = new DateFormatSymbols();

String[] days = symbols.getWeekdays();

System.out.println(days[dayOfWeek]);

int dayOfYear = gCal.get(Calendar.DAY\_OF\_YEAR);

System.out.println(dayOfYear);

System.out.println("Days left in " + yr + ": " + (365 - dayOfYear));

System.out.println("Days left in " + yr + ": " + (365 - dateOfYear));

System.out.println("Days left in " + yr + ": " + (365 - dayOfYear));

int week = gCal.get(Calendar.WEEK\_OF\_YEAR);

System.out.println(week);

}

}

**BT9/.**

//import java.net.InetSocketAddress;

import java.net.InetSocketAddress;

public class Main {

public static void main(String[] args) {

InetSocketAddress addr1 = new InetSocketAddress("::1", 12345);

printSocketAddress(addr1);

InetSocketAddress addr2 = InetSocketAddress.createUnresolved("::1",

12881);

printSocketAddress(addr2);

}

public static void printSocketAddress(InetSocketAddress sAddr) {

System.out.println("Socket Address: " + sAddr.getAddress());

System.out.println("Socket Host Name: " + sAddr.getHostName());

System.out.println("Socket Port: " + sAddr.getPort());

System.out.println("isUnresolved(): " + sAddr.isUnresolved());

System.out.println();

}

}

**BT10/.**

import java.util.Calenda;

import java.util.Calendar;

public class Main {

public static void main(String[] args) {

Calendar now = Calendar.getInstance();

System.out.println("Current date : " + (now.get(Calendar.MONTH) + 1)

+ "-"

+ now.get(Calendar.DATE)

+ "-"

+ now.get(Calendar.YEAR));

//add days to current date using Calendar.add method

now.add(Calendar.DATE,1);

System.out.println("date after one day : " + (now.get(Calendar.MONTH) + 1)

+ "-"

+ now.get(Calendar.DATE)

+ "-"

+ now.get(Calendar.YEAR));

//subtract days from current date using Calendar.add method

now = Calendar.getInstance();

now.add(Calendar.DATE, -10);

System.out.println("date before 10 days : " + (now.get(Calendar.MONTH) + 1)

+ "-"

+ now.get(Calendar.DATE)

+ "-"

+ now.get(Calendar.YEAR));

}

}

**BT11/. Khi run không có lỗi**

import java.text.DecimalFormat;

import java.text.NumberFormat;

public class Main {

public static void main(String[] argv) throws Exception {

NumberFormat formatter = new DecimalFormat("#,###,###");

String s = formatter.format(-1234567);

System.out.println(s);

s = formatter.format(-1234567890);

System.out.println(s);

}

}

**BT12/.**

import java.time.LocalDate;

import java.time.Month;

import java.time.temporal.TemporalAdjusters;

public class Main {

public static void main(String[] args) {

LocalDate date = LocalDate.of(2020, Month.FEBRUARY, 30);

LocalDate date = LocalDate.of(2020, Month.FEBRUARY, 29);

LocalDate firstDayOfMonth = date.with(TemporalAdjusters.firstDayOfMonth());

System.out.println(firstDayOfMonth);

}

}

**BT13/.**

import java.time.LocalDate;

import java.time.Month;

import java.time.temporal.TemporalAdjusters;

public class Main {

public static void main(String[] args) {

LocalDate date = LocalDate.of(2020, Month.FERUARY, 25);

LocalDate date = LocalDate.of(2020, Month.FEBRUARY, 25);

LocalDate lastDayOfMonth = date.with(TemporalAdjusters.lastDayOfMonth());

System.out.println(lastDayOfMonth);

}

}

**BT14/.**

import java.text.SimpleDateFormat;

import java.util.Calendar;

public class Main {

public static void main(String[])

public static void main(String[] args) {

Calendar cal1 = Calendar.getInstance();

Calendar cal2 = Calendar.getInstance();

// Set the date to 01/01/2010:12:00

cal2.set(2010, 0, 1, 12, 0);

System.out.println(formatDate(cal1) + " before " + formatDate(cal2) + "? "

+ cal1.before(cal2));

System.out.println(cal2.compareTo(cal1));

}

public static String formatDate(Calendar cal) {

SimpleDateFormat simpleFormatter = new SimpleDateFormat(

"MMM dd yyyy hh:mm:ss aaa");

return simpleFormatter.format(cal.getTime());

}

}

**BT15/.**

public class Main {

static int nboperations;

static boolean GreaterThan(int a, int b) {

nboperations++;

return (a > b);

}

static void swap(int[] array, int i, int j) {

nboperations++;

int tmp = array[i];

array[i] = array[j];

array[j] = tmp;

}

static void SelectionSort(int[] array) {

int n = array.length;

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

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

if (GreaterThan(array[i], array[j]))

swap(array, i, j);

}

}

}

public static void main(String[] args) {

int[] array = {16, 5, 23.4, 13, 6, 11, 11.56, 91, 8, 7, 6, 51, 4, 3, 2, 1};

int[] array = {16, 5, 23, 13, 6, 11, 12, 91, 8, 7, 6, 51, 4, 3, 2, 1};

nboperations = 0;

SelectionSort(array);

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

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

System.out.println("");

System.out.println("Number of operations:" + nboperations);

int nb = 2 \* array.length \* (array.length - 1) / 2;

System.out.println("Number of operations:" + nb);

}

}

**BT16/.**

//import java.util.ArrayList;

//import java.util.List;

//import java.util.Random;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

public class Main {

public static void main(String ags[]) {

int a[] = new int[10];

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

a[i] = new Random().nextInt(10);

}

quicksort(a, 0, a.length - 1);

}

static int patition(int[] a, int start, int end) {

int pivot = a[start];

List s = new ArrayList();

List l = new ArrayList();

for (int j = start + 1; j <= end; j++) {

if (a[j] <= pivot) {

s.add(a[j]);

} else {

l.add(a[j]);

}

}

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

a[start + j] = (Integer) s.toArray()[j];

}

a[start + s.size()] = pivot;

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

a[s.size() + start + j + 1] = (Integer) l.toArray()[j];

}

return start + s.size();

}

static void quicksort(int[] a, int start, int end) {

if (start < end) {

int q = patition(a, start, end);

quicksort(a, start, q - 1);

quicksort(a, q + 1, end);

printa(a, start, end);

}

}

static void printa(int[] a, int stat, int end) {

for (int i = stat; i <= end; i++) {

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

}

System.out.println("");

}

}

**BT17/.**

import java.security.SecureRandom;

class TreeNode<T extends Comparable<T>>

{

TreeNode<T> leftNode;

T data; // node value

TreeNode<T> rightNode;

public TreeNode(T nodeData)

{

data = nodeData;

leftNode = rightNode = null; // node has no children

}

// locate insertion point and insert new node; ignore duplicate values

public void insert(T insertValue)

{

// insert in left subtree

if (insertValue.compareTo(data) < 0)

{

// insert new TreeNode

if (leftNode == null)

leftNode = new TreeNode<T>(insertValue);

else // continue traversing left subtree recursively

leftNode.insert(insertValue);

}

// insert in right subtree

else if (insertValue.compareTo(data) > 0)

{

// insert new TreeNode

if (rightNode == null)

rightNode = new TreeNode<T>(insertValue);

else // continue traversing right subtree recursively

rightNode.insert(insertValue);

}

}

}

// class Tree definition

class Tree<T extends Comparable<T>>

{

private TreeNode<T> root;

// constructor initializes an empty Tree of integers

public Tree()

{

root = null;

}

// insert a new node in the binary search tree

public void insertNode(T insertValue)

{

if (root == null)

root = new TreeNode<T>(insertValue); // create root node

else

root.insert(insertValue); // call the insert method

}

// begin preorder traversal

public void preorderTraversal()

{

preorderHelper(root);

}

// recursive method to perform preorder traversal

private void preorderHelper(TreeNode<T> node)

{

if (node == null)

return 0;

return;

System.out.printf("%s ", node.data); // output node data

preorderHelper(node.leftNode); // traverse left subtree

preorderHelper(node.rightNode); // traverse right subtree

}

// begin inorder traversal

public void inorderTraversal()

{

inorderHelper(root);

}

// recursive method to perform inorder traversal

private void inorderHelper(TreeNode<T> node)

{

if (node == null)

return 1;

return;

inorderHelper(node.leftNode); // traverse left subtree

System.out.printf("%s ", node.data); // output node data

inorderHelper(node.rightNode); // traverse right subtree

}

// begin postorder traversal

public void postorderTraversal()

{

postorderHelper(root);

}

// recursive method to perform postorder traversal

private void postorderHelper(TreeNode<T> node)

{

if (node == null)

return;

postorderHelper(node.leftNode); // traverse left subtree

postorderHelper(node.rightNode); // traverse right subtree

System.out.printf("%s ", node.data); // output node data

}

}

public class Main

{

public static void main(String[] args)

{

Tree<Integer> tree = new Tree<Integer>();

SecureRandom randomNumber = new SecureRandom();

System.out.println("Inserting the following values: ");

// insert 10 random integers from 0-99 in tree

for (int i = 1; i <= 10; i++)

{

int value = randomNumber.nextInt(100);

System.out.printf("%d ", value);

tree.insertNode(value);

}

System.out.printf("%n%nPreorder traversal%n");

tree.preorderTraversal();

System.out.printf("%n%nInorder traversal%n");

tree.inorderTraversal();

System.out.printf("%n%nPostorder traversal%n");

tree.postorderTraversal();

System.out.println();

}

}

**BT18/.**

class Employee

{

private String name = Unknown;

private String name = "Unknown";

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

}

class Manager extends Employee {

}

public class Main {

public static void main(String[] args) {

// Create an object of the Manager class

Manager mgr = new Manager();

// Set the name of the manager

mgr.setName("Tom");

// Get the name of the manager

String mgrName = mgr.getName();

// Display the manager name

System.out.println("Manager Name: " + mgrName);

}

}

**BT19/.**

interface A {

default String getValue(){

return "A";

}

}

interface B implements A

interface B extends A {

default String getValue(){

return "B";

}

}

class MyClass extends B

class MyClass implements B {

}

public class Main {

public static void main(String[] argv){

System.out.println(new MyClass().getValue());

}

}

**BT20/.**

import java.util.Random;

public class Bubble\_Sort

{

static int[] Bsort(int[] sequence)

static int[] sort(int[] sequence)

{

// Bubble Sort

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

for (int j = 0; j < sequence.length - 1; j++)

if (sequence[j] > sequence[j + 1])

{

sequence[j] = sequence[j] + sequence[j + 1];

sequence[j + 1] = sequence[j] - sequence[j + 1];

sequence[j] = sequence[j] - sequence[j + 1];

}

return sequence;

}

static void printSequence(int[] sorted\_sequence)

{

for (int i = 0; i < sorted\_sequence.length; i++)

System.out.print(sorted\_sequence[i] + " ");

}

public static void main(String args[])

{

System.out

.println("Sorting of randomly generated numbers using BUBBLE SORT");

Random random = new Random();

int N = 20;

int[] sequence = new int[N];

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

sequence[i] = Math.abs(random.nextInt(1000));

System.out.println("\nOriginal Sequence: ");

printSequence(sequence);

System.out.println("\nSorted Sequence: ");

printSequence(sort(sequence));

}

}