1. Write the class Date having attributes like day, month & year. Add default & parameterized constructors. Add getters & setters. Add method to print the date. Add method to swap two

dates.

package com.zensar;

public class Day4 {

Day4()

{this.day="Monday";

this.month="February";

this.year="2022";

}

Day4(String day,String month,String year)

{this.day=day;

this.month=month;

this.year=year;

}

private String day;

private String month;

private String year;

public void setDay(String day)

{this.day=day;

}

public void setmonth(String month)

{

this.month=month;

}

public void setyear(String year)

{

this.year=year;

}

public String getDay()

{

return day;

}

public String getMonth()

{

return month;

}

public String getyear()

{

return year;

}

public static void printDate()

{

Day4 day4 = new Day4();

String day=day4.getDay();

String month=day4.getMonth();

String year=day4.getyear();

System.out.println("stored date : " + day+ "/" + month+"/" + year);

}

public static void swapDate()

{

Day4 day4 = new Day4();

String day=day4.getDay();

String month=day4.getMonth();

String year=day4.getyear();

day4.setDay("Saturday");

String day2=day4.getDay();

System.out.println(" before swapping = "+ day + " " + day2);

System.out.println("stored date : " + day+ "/" + month+"/" + year);

String temp;

temp=day;

day=day2;

day2=temp;

System.out.println(" after swapping ="+ day + " " + day2);

System.out.println("stored date : " + day+ "/" + month+"/" + year);

}

public static void main(String[] args) {

printDate();

swapDate();

}}

1. Write a class ComplexNumber having attributes real & imaginary. Add functions like add, subtract, multiply & swap.

package com.zensar;

class calc

{

float real, img;

calc() {} // Do Nothing Constructor

calc(float r, float i)

{

real = r;

img = i;

}

void display()

{

System.out.println(real+" + i "+img);

}

calc add(calc c2)

{

calc res = new calc();

res.real = real + c2.real;

res.img = img + c2.img;

return(res);

}

calc sub(calc c2)

{

calc res = new calc();

res.real = real - c2.real;

res.img = img - c2.img;

return(res);

}

}

class Complex

{

public static void main(String args[])

{

calc c1 = new calc(12.5F, 2.5F);

calc c2 = new calc(09.5F, 0.5F);

System.out.println("C1 is: ");

c1.display();

System.out.println("C2 is: ");

c2.display();

calc c3 = new calc();

System.out.println("Addition of C1 and C2 is: ");

c3 = c1.add(c2);

c3.display();

System.out.println("Subtraction of C1 and C2 is: ");

c3 = c1.sub(c2);

c3.display();

}

}

1. Write a class Account & add methods like deposit, withdraw, print etc.

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class Account{

static int deposite(int a){

System.out.print("Deposite: ");

System.out.println(a);

return a;

}

static int withdraw(int b){

System.out.print("Withdraw: ");

System.out.println(b);

return b;

}

static int print(int z){

System.out.print("Available Balance: ");

System.out.println(z);

return z;

}

public static void main(String[] args) {

int a=35000;

int b=15000;

int z = a-b;

int c=deposite(a);

int d =withdraw(b);

int x =print(z);

}

}

1. Write a program to implement a Stack using arrays as follows-

class StackedArray {

int ary[];

push(--) { }

pop() {--) {}

}

import java.util.\*;

import java.lang.\*;

import java.io.\*;

public class Prac1 {

int size;

int arr[];

int top;

public void StackArray(int size){

size = 3;

arr=new int[size];

top = -1;

}

public boolean isFull(){

return (top==size-1);

}

public boolean isEmpty(){

return (size-1==top);

}

public int peek(){

return arr[top];

}

public void push(int num){

if(!isFull()){

arr[++top]=num;

System.out.println(" Element Pushed" +num);

}else{

System.out.println("Stack is full: Overflow");

}

}

public int pop(){

if (isFull()){

int val = arr[top];

top--;

System.out.println("Popped element is "+ val);

return val;

}else{

System.out.println("Stack is empty: Underflow ");

return -1;

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Stack st = new Stack();

st.push(3);

st.push(7);

System.out.println("Hellow world");

System.out.println(st.isEmpty());

System.out.println(st.peek());

System.out.println(st.pop());

}

}

5. Write a program to implement a Queue using arrays as follows-

class QueuedArray {

int ary[];

push(--) { }

pop() {--) {}

}

package com.zensar;

public class Queue {

int queue[] = new int[5];

int size;

int rear;

int front;

public void enQueue(int data){

queue[rear]=data;

rear=rear+1;

size=size+1;

}

public void show(){

System.out.print("Elements: ");

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

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

}

}

public int deQueue(){

int data = queue[front];

size=size-1;

return data;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Queue st = new Queue();

st.enQueue(4);

st.enQueue(9);

st.enQueue(2);

st.enQueue(8);

st.deQueue();

st.show();

}

}

6. Write a single tone class. Confirm that single tone class cannot be inherited.

**package** com.zensar.training;

**public** **class** Singletone {

**private** **static** Singletone *singletone*=**null**;

**public** **static** Singletone singletoneMethod()

{

**if**(*singletone* == **null**)

{

System.***out***.println("object is null---create object");

*singletone*=**new** Singletone();

**return** *singletone*;

}

**else**

{

System.***out***.println("object cannot be created");

**return** *singletone*;

}

}

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

Singletone singletone=Singletone.*singletoneMethod*();

Singletone singletone1=Singletone.*singletoneMethod*();

}

}

1. Write java classes to build doubly linked list. Add functionalities like add new node, insert node, delete node, count nodes & print linked list.

class Node {

Node previous;

Node next;

Int data;

}

package com.zensar;

public class DoublyLinlist {

class Node{

int data;

Node previous;

Node next;

public Node(int data) {

this.data = data;

}

}

Node head, tail = null;

public void addNode(int data) {

Node newNode = new Node(data);

if(head == null) {

head = tail = newNode;

head.previous = null;

tail.next = null;

}

else {

tail.next = newNode;

newNode.previous = tail;

tail = newNode;

tail.next = null;

}

}

public void display() {

Node current = head;

if(head == null) {

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

return;

}

System.out.println("Nodes of doubly linked list: ");

while(current != null) {

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

current = current.next;

}

}

public static void main(String[] args) {

DoublyLinlist dList = new DoublyLinlist();

dList.addNode(1);

dList.addNode(2);

dList.addNode(3);

dList.addNode(4);

dList.addNode(5);

dList.display();

}

}