

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 Date
```

```
{
```

```
    Date()
```

```
    {
```

```
        this.day="Monday";
```

```
        this.month="April";
```

```
        this.year="2022";
```

```
    }
```

```
    Date(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()
{
    Date d = new Date();
    String day=d.getDay();
    String month=d.getMonth();
    String year=d.getyear();
    System.out.println("stored date : " + day+ "/" + month+"/"+ year);
}

public static void swapDate()
{
    Date d = new Date();
    String day=d.getDay();
    String month=d.getMonth();
    String year=d.getyear();
    d.setDay("Monday");
    String day2=d.getDay();
    System.out.println(" before swapping = "+ day + " "+ day2);

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

}

public static void main(String[] args)
{
    printDate();
    swapDate();
}
}

```

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

```

package com.zensar;

```

```

public class ComplexNumber
{
    double real;
    double img;

    public ComplexNumber(double real, double img) // Parameterize
Constructor
    {
        this.real = real;
        this.img = img;
    }

    public static ComplexNumber add(ComplexNumber n1, ComplexNumber n2) {
        ComplexNumber temp = new ComplexNumber(0.0, 0.0);
        temp.real = n1.real + n2.real;
        temp.img = n1.img + n2.img;
        return temp;
    }

    public static ComplexNumber sub(ComplexNumber n1, ComplexNumber n2) {
        ComplexNumber temp = new ComplexNumber(0.0, 0.0);
        temp.real = n1.real - n2.real;
        temp.img = n1.img - n2.img;
        return temp;
    }

    public static ComplexNumber mul(ComplexNumber n1, ComplexNumber n2) {
        ComplexNumber temp = new ComplexNumber(0.0, 0.0);
        temp.real = n1.real * n2.real;
        temp.img = n1.img * n2.img;
        return temp;
    }

    public static ComplexNumber div(ComplexNumber n1, ComplexNumber n2) {
        ComplexNumber temp = new ComplexNumber(0.0, 0.0);
        temp.real = n1.real / n2.real;
        temp.img = n1.img / n2.img;
        return temp;
    }

    public static void main(String[] args) {
        ComplexNumber n1 = new ComplexNumber(6, 8);
        ComplexNumber n2 = new ComplexNumber(3, 2);
        ComplexNumber addition = add(n1, n2);
        ComplexNumber subtraction = sub(n1, n2);
        ComplexNumber multiplication = mul(n1, n2);
        ComplexNumber division = div(n1, n2);
    }
}

```

```

        System.out.println("Addition: " + addition.real + "i " + "+ " +
addition.img);
        System.out.println("Subtraction : " + subtraction.real + "i " +
"- " + subtraction.img);
        System.out.println("Multiplication : " + multiplication.real +
"i " * "+ " + multiplication.img);
        System.out.println("Division : " + division.real + "i " + "/" +
division.img);

    }

}

```

### OUTPUT:

Addition: 9.0i + 10.0

Subtraction : 3.0i - 6.0

Multiplication : 18.0i \* 16.0

Division : 2.0i / 4.0

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

```

package com.zensar;

public class Account {
    int ac_no;
    String name;
    int amount;

    void details(int n, String naam, int a)
    {
        ac_no = n;
        name = naam;
        amount = a;
    }

    void show()
    {
        System.out.println(ac_no + " " + name + " " + amount);
    }
}

```

```

    }

    void deposit(int a)
    {
        amount += a;
        System.out.println("Deposit Amount:" + a);
    }

    void withdraw(int a)
    {
        if (amount < a)
        {
            System.out.println("Insufficient balance");
        } else
        {
            amount -= a;
            System.out.println("Withdraw Amount:" + a);
        }
    }

    void checkbalance()
    {
        System.out.println("Balance is: " + amount);
    }

    public static void main(String[] args) {
        Account obj = new Account();
        obj.details(345512, "Sahil", 20000);
        obj.show();
        obj.deposit(20000);
        obj.withdraw(10000);
        obj.checkbalance();
    }
}

```

**OUTPUT:**

```

345512 Sahil 20000
Deposit Amount:20000
Withdraw Amount:10000
Balance is: 30000

```

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

```

class StackedArray {

    int ary[];

```

```
push(--){}

pop(){--} {}
```

```
}
```

```
package com.zensar;
```

```
public class StackedArray
{
```

```
    int size;
    int arr[];
    int top;
    public StackedArray(int size)
    {
        this.size=size;
        this.arr=new int[size];
        this.top=-1;
    }
    public boolean isFull()
    {
        return(size-1==top);
    }
    public boolean isEmpty()
    {
        return(top== -1);
    }
    public int peek()
    {
        return arr[top];
    }
    public void push(int num)
    {
        if(!isFull())
        {
            arr[++top]=num;
            System.out.println("Element inserted/push: "+num);
        }
        else
        {
            System.out.println("Stack is full");
        }
    }
    public int pop() {
        if(!isEmpty())
        {
            int val=arr[top];
            top--;
            System.out.println("Deleted /poped element is: "+val);
            return val;
        }
        else
        {
            System.out.println("Stack is empty");
            return -1;
        }
    }
}
```

```

    }
    public static void main(String[] args)
    {
        StackedArray stack=new StackedArray(3);
        stack.push(4);
        stack.push(3);
        stack.push(13);
        stack.push(11);
        stack.pop();
        stack.pop();
        System.out.println("Element at peek is: "+ stack.peek());

    }
}

```

#### OUTPUT:

```

Element inserted/push: 4
Element inserted/push: 3
Element inserted/push: 13
Stack is full
Deleted /popped element is: 13
Deleted /popped element is: 3
Element at peek is: 4

```

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

```

class QueuedArray {

    int ary[];

    push(--) { }

    pop() {--} {}

}

```

```

package com.zensar;
public class QueuedArray {
    int size=5;
    int [] array=new int[size];
    int rear=-1;
    int front=-1;
    public void enqueue(int data)
    {
        if(rear==size-1)
        {
            System.out.println("overflow");
        }
        if(front==-1 && rear == -1) {

```

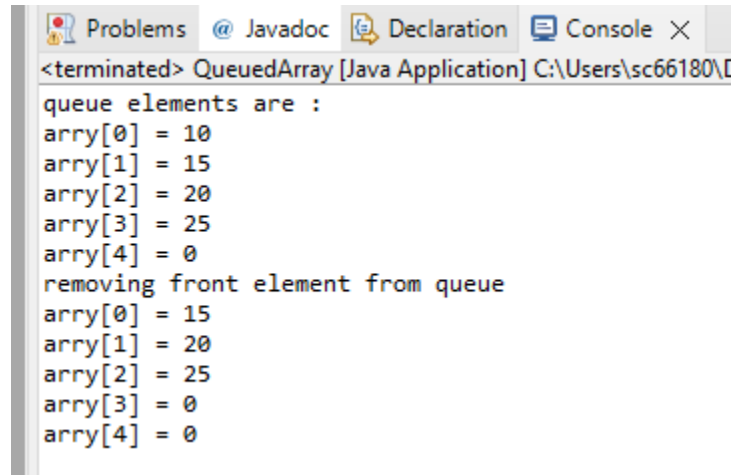
```

        front++;
        rear++;
        array[rear]=data;
    }
    else
        rear++;
        array[rear]=data;
    }
    public void show()
    {
        for(int i=0;i<array.length;i++)
        {
            System.out.println("array["+ i + "] = " +array[i]);
        }
    }
    public void deque()
    {
        if(front==-1 && rear == -1)
        {
            System.out.println("array is empty");
        }
        else if(front==rear)
        {
            array[front]=0;
            front = rear =-1;
        }
        else
        {
            for(int i=0;i<=rear;i++)
            {
                array[i]=array[i+1];
            }
            --rear;
        }
    }
}
public static void main(String[] args) {
    QueuedArray queue=new QueuedArray();
    queue.enqueue(10);
    queue.enqueue(15);
    queue.enqueue(20);
    queue.enqueue(25);
    System.out.println("queue elements are :");
    queue.show();
    System.out.println("removing front element from queue");
    queue.deque();
    queue.show();
}
}

```



OUTPUT:

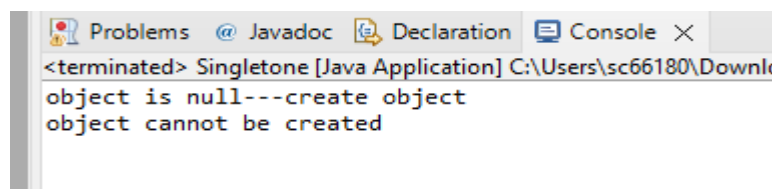


```
<terminated> QueuedArray [Java Application] C:\Users\sc66180\I
queue elements are :
arry[0] = 10
arry[1] = 15
arry[2] = 20
arry[3] = 25
arry[4] = 0
removing front element from queue
arry[0] = 15
arry[1] = 20
arry[2] = 25
arry[3] = 0
arry[4] = 0
```

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

```
package com.zensar;
public class Singleton {
    private static Singleton singleton=null;
    public static Singleton singletonMethod()
    {
        if(singleton == null)
        {
            System.out.println("object is null---create object");
            singleton=new Singleton();
            return singleton;
        }
        else
        {
            System.out.println("object cannot be created");
            return singleton;
        }
    }
    public static void main(String[] args)
    {
        Singleton singleton=Singleton.singletonMethod();
        Singleton singleton1=Singleton.singletonMethod();
    }
}
```

OUTPUT:



```
<terminated> Singleton [Java Application] C:\Users\sc66180\Downl
object is null---create object
object cannot be created
```

7. 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;
```

```
class DLL  
{
```

```
    Node head,tail = null;
```

```
    class Node  
    {
```

```
        int data;  
        Node prev;  
        Node next;
```

```
        Node(int d)  
        {  
            data = d;  
        }  
    }
```

```
    void insert(int data)  
    {
```

```
        Node new_node = new Node(data);
```

```
        if(head==null)  
        {  
            head = tail = new_node;  
            head.prev = null;  
            tail.next = null;  
        }
```

```
        tail.next = new_node;  
        new_node.prev = tail;  
        tail = new_node;  
        new_node.next = null;
```

```
    }
```

```
    void delete(Node del)
```

```
    {
```

```
        if(head == null )
```

```

        {
            return;
        }
        if(head == del) {
            head = del.next;
        }

        if(del.next != null) {
            del.next.prev = del.prev;
        }

        if(del.prev != null) {
            del.prev.next = del.next;
        }
        return;
    }
    void printNodes()
    {
        Node curr = head;
        if(head == null)
        {
            System.out.println("DLL is empty");
            return;
        }

        while(curr!=null)
        {
            System.out.print(curr.data + "-> ");

            curr = curr.next;
        }
        System.out.println(" ");
    }
}

public int countNodes() {
    int counter = 0;

    Node current = head;

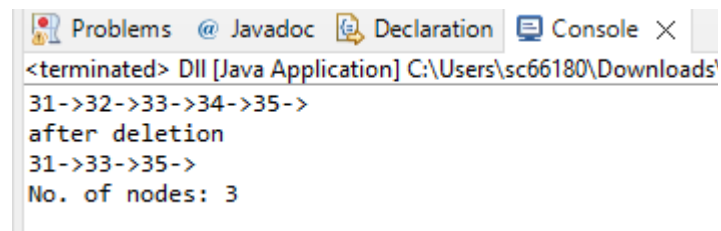
    while(current != null) {
        counter++;
        current = current.next;
    }
    return counter;
}

public static void main(String[] args) {
    DLL dl = new DLL();
    dl.insert(31);
    dl.insert(32);
    dl.insert(33);
    dl.insert(34);
    dl.insert(35);
    dl.printNodes();
}

```

```
dl.delete(dl.head.next);  
dl.delete(dl.tail.prev);  
  
System.out.println("after deletion");  
dl.printNodes();  
System.out.println("No. of nodes: "+ dl.countNodes());  
  
}  
}
```

OUTPUT:



```
<terminated> Dll [Java Application] C:\Users\sc66180\Downloads  
31->32->33->34->35->  
after deletion  
31->33->35->  
No. of nodes: 3
```