

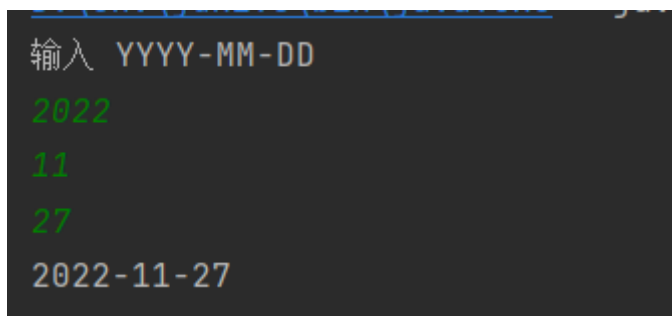
1、编写一个基于对象的程序，要求：

(1) 定义一个日期类Date，类内有私有数据成员year（年）、month（月）、day（日），公有成员函数set_date()、show_date()。

(2) set_date()作用是从键盘输入年、月、日的值，show_date()的作用是在屏幕上显示年、月、日的值。

(3) 在main()函数定义Date类的对象d1，并调用set_date()函数给日期赋值，调用show_date()函数输出日期的值。

```
public class Test {  
    static class MyDate {  
        private String year;  
        private String month;  
        private String day;  
  
        public void setDate() {  
            System.out.println("输入 YYYY-MM-DD");  
            Scanner sc = new Scanner(System.in);  
            this.year = sc.next();  
            this.month = sc.next();  
            this.day = sc.next();  
        }  
  
        public void showDate() {  
            System.out.println(year + "-" + month + "-" + day);  
        }  
    }  
  
    public static void main(String[] args) {  
        MyDate myDate = new MyDate();  
        myDate.setDate();  
        myDate.showDate();  
    }  
}
```



A screenshot of a terminal window showing the execution of the Java program. The prompt '输入 YYYY-MM-DD' is displayed. The user enters '2022' for the year, '11' for the month, and '27' for the day. The program then outputs '2022-11-27'.

2、按要求编写一个Java应用程序：

(1) 编写一个矩形类Rect，包含：矩形的宽width；矩形的高height。两个构造方法：一个带有两个参数的构造方法，用于width和height属性初始化；一个不带参数的构造方法，将矩形初始化为宽和高都为10。两个方法：一个求矩形面积的方法area()，另一个求矩形周长的方法perimeter()。

(2) 通过继承Rect类编写一个具有确定位置的矩形类PlainRect，其确定位置用矩形的左上角坐标来标识，包含：添加两个属性：矩形左上角坐标startX和startY。两个构造方法：带4个参数的构造方法，用于对startX、startY、width和height属性初始化；不带参数的构造方法，将矩形初始化为左上角坐标、长和宽都为0的矩形；添加一个方法：判断某个点是否在矩形内部的方法isInside(double x,double y)。如在矩形内，返回true, 否则，返回false。

提示：点在矩形类是指满足条件：

$$x \geq \text{startX} \ \&\& \ x \leq (\text{startX} + \text{width}) \ \&\& \ y \geq \text{startY} \ \&\& \ y \leq (\text{startY} + \text{height})$$

(3) 编写PlainRect类的测试程序，创建一个左上角坐标为（10，10），长为20，宽为10的矩形对象；计算并打印输出矩形的面积和周长；判断点(25.5，13)是否在矩形内，并打印输出相关信息。

```
public class Rect {

    public static void main(String[] args) {
        PlainRect plainRect = new PlainRect(10.0, 20.0, 10.0, 10.0);
        System.out.println("面积:" + plainRect.area());
        System.out.println("周长:" + plainRect.perimeter());
        System.out.println("(25.5, 13) 是否在矩形内?" + plainRect.isInside(25.5,
13.0));
    }

    protected Double width;
    protected Double height;

    public Rect(Double width, Double height) {
        this.width = width;
        this.height = height;
    }

    public Rect() {
        this.height = 10.0;
        this.width = 10.0;
    }

    public Double area() {
        return this.width * this.height;
    }

    public Double perimeter() {
        return 2 * (this.height + this.width);
    }
}

class PlainRect extends Rect {
    private Double startX;
    private Double startY;

    public PlainRect(Double width, Double height, Double startX, Double startY)
{
        super(width, height);
        this.startX = startX;
        this.startY = startY;
    }
}
```

```

    public boolean isInside(Double x, Double y) {
        return x >= startX && x <= (startX + super.width) && y < startY && y >=
(startY - super.height);
    }
}

```

```

面积:200.0
周长:60.0
(25.5, 13) 是否在矩形内?false

```

3、编写一个类ArraySearch

该类有一个成员变量array，它是int类型的一维数组；和一个searchArray(int)的成员方法；并有一个带参数的构造方法，该参数将在构造方法中为成员变量数组array作初始化，如果传入的数组为null，应抛出 IllegalArgumentException异常。

方法searchArray(int)接收一个整数作为参数，这个方法搜索array数组中是否存在指定的整数，如果存在，则返回该整数在数组中的位置，否则返回-1。

另有一个类TestArray，它只有一个main()方法，在该方法中，创建ArraySearch类的对象，并用一个int类型的一维数组来实例化这个对象的成员变量数组array。通过调用对象的searchArray(int)方法对数组进行搜索。

```

public class TestSearch {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5, 6};
        ArraySearch arraySearch = new ArraySearch(arr);
        System.out.println(arraySearch.searchArray(5));
    }

    public static class ArraySearch {
        private int[] arr;

        public ArraySearch(int[] arr) {
            if (Objects.isNull(arr)) {
                throw new IllegalArgumentException();
            }
            this.arr = arr;
        }

        public Integer searchArray(int elem) {
            int index = 0;
            for (int i : this.arr) {
                if (i == elem) break;
                index++;
            }
            return index;
        }
    }
}

```

```
TestSearch
D:\env\jdk1.8\bin\java.exe ...
4
```

4、按如下要求编写Java程序：

- (1) 定义接口A，里面包含值为3.14的常量PI和抽象方法double area()。
- (2) 定义接口B，里面包含抽象方法void setColor(String c)。
- (3) 定义接口C，该接口继承了接口A和B，里面包含抽象方法void volume()。
- (4) 定义圆柱体类Cylinder实现接口C，该类中包含三个成员变量：底圆半径radius、圆柱体的高度height、颜色color。
- (5) 创建主类来测试类Cylinder。

```
public class Demo {
    public static void main(String[] args) {
        Cylinder cylinder = new Cylinder(5.0, 10.0, "红色");
        System.out.println("面积为" + cylinder.area());
        System.out.println("体积为" + cylinder.volume());
        cylinder.setColor("蓝色");
        System.out.println("新颜色为" + cylinder.getColor());
    }
}

interface A {
    Double PI = 3.14;

    Double area();
}

interface B {
    void setColor(String color);
}

interface C extends A, B {
    Double volume();
}

class Cylinder implements A, B, C {

    private Double radius;
    private Double height;
    private String color;

    public Cylinder(Double radius, Double height, String color) {
        this.radius = radius;
        this.height = height;
        this.color = color;
    }

    public String getColor() {
```

```

        return color;
    }

    @Override
    public Double area() {
        return 2 * PI * radius * height;
    }

    @Override
    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public Double volume() {
        return radius * radius * PI / 2 * height;
    }
}

```

```

    面积为314.0
    体积为392.5
    新颜色为蓝色

```

5、在类List类内部建立一个内部类Node，

有成员数据 int data，成员数据 Node next和Node pre 引用变量。List类具有成员变量Node first和Node last，List类具有成员函数插入、删除结点、查找结点；两个链表的连接、结点数据的排序、两个有序链表的合并成有序链表等成员方法；构造函数、链表深复函数、链表的置空等。

```

public class MyList {
    public static void main(String[] args) {
        MyList list = new MyList(new ArrayList<Node>() {{
            add(new Node(1));
            add(new Node(2));
        }});
        list.insert(5);
        list.insert(6);
        list.insert(3);
        list.connect(new ArrayList<Node>() {{
            add(new Node(7));
            add(new Node(8));
        }});
        System.out.println(list);
    }

    @Override
    public String toString() {
        return list.toString();
    }

    static class Node {
        public int data;
    }
}

```

```

    public Node(int data) {
        this.data = data;
    }

    @Override
    public String toString() {
        return "Node{" +
            "data=" + data +
            '}';
    }
}

Map<Integer, List<Node>> cache = new HashMap<>();
public List<Node> list;

public void insert(int data) {
    Node e = new Node(data);
    list.add(e);
    if (cache.get(data) == null) {
        cache.put(data, new ArrayList<>());
    } else {
        cache.get(data).add(e);
    }
}

public void delete(int data) {
    List<Node> nodes = cache.get(data);
    if (nodes == null) {
        throw new RuntimeException("删除失败");
    } else {
        list.remove(nodes.get(0));
    }
}

public Node search(int data) {
    if (cache.get(data) == null) {
        throw new RuntimeException("没有这个结点");
    } else {
        return cache.get(data).get(0);
    }
}

public void connect(List<Node> otherList) {
    this.list.addAll(otherList);
    this.list.sort(Comparator.comparingInt(n -> n.data));
}

public MyList(List<Node> list) {
    this.list = list;
}

public List<Node> deepCopy() {
    return new ArrayList<>(this.list);
}

```

```

    public void empty() {
        this.list = new ArrayList<>();
    }
}

```

```
[Node{data=1}, Node{data=2}, Node{data=3}, Node{data=5}, Node{data=6}, Node{data=7}, Node{data=8}]
```

6、

用Java实现二叉排序树的创建、插入结点、删除结点、前序、中序、后序遍历算法。定义一个名为Person的类，含有两个String类型的成员变量name和sex，一个int类型的成员变量age，它们用protect修饰符，分别实现getXXXX访问方法和setXXXX修改方法；实现构造方法Person（String name,String sex,int age）；实现成员方法display()显示输出类的成员变量的信息；并编写独立的测试文件测试Person类。

```

package cn.wjb114514;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

/**
 * Unit test for simple App.
 */
public class AppTest
{
    static class Person {
        private String name;
        private Integer age;
        private String sex;

        public Person(String name, Integer age, String sex) {
            this.name = name;
            this.age = age;
            this.sex = sex;
        }

        public void display() {
            System.out.println("name" + name + ", age" + age + ",sex" + sex);
        }
    }

    @Test
    public void test() {
        new Person("wjb", 19, "♂").display();
    }
}

```

```
namewjb, age19,sex♂
```

```

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) throws IOException,
    ClassNotFoundException {
        MyFile myFile = new MyFile("./demo.txt");
        List<Object> list = new ArrayList<>();
        Student student = new Student("小李", "男", 18, "1");
        Teacher teacher = new Teacher("王老师", "女", 25, "1");
        myFile.writeFile(student);
        myFile.writeFile(teacher);
        myFile.readFile(list);
        Student s = (Student) list.get(0);
        Teacher t = (Teacher) list.get(1);
        System.out.println("学生[" + s.getName() + "]的老师为[" + t.getName() +
        "]);
    }
}

/*
 * Student、Teacher、Course Schedule、Electivecourse
 * */
class Student extends Person {
    // 唯一标识id
    String stuId;

    public Student(String stuId) {
        this.stuId = stuId;
    }

    public Student(String name, String sex, Integer age, String stuId) {
        super(name, sex, age);
        this.stuId = stuId;
    }

    public String getStuId() {
        return stuId;
    }

    public void setStuId(String stuId) {
        this.stuId = stuId;
    }
}

class Teacher extends Person {
    String refStuId;

    public Teacher(String refStuId) {
        this.refStuId = refStuId;
    }
}

```



```

        public Teacher(String name, String sex, Integer age, String refStuId) {
            super(name, sex, age);
            this.refStuId = refStuId;
        }
    }

    public class MyFile {
        File opFile;

        public MyFile() {
        }

        public MyFile(String filename) throws IOException {
            opFile = new File(filename);
            if (!opFile.exists()) {
                if (opFile.createNewFile()) {
                    System.out.println("已为您创建好文件");
                } else {
                    System.out.println("文件创建失败");
                }
            } else {
                System.out.println("文件已存在，无需创建");
            }
        }

        public void readFile(List<Object> objList) throws IOException,
        ClassNotFoundException {
            ObjectInputStream ois = new ObjectInputStream(new
            FileInputStream(opFile));
            try {
                while (true) {
                    Object o = ois.readObject();
                    objList.add(o);
                }
            } catch (EOFException e) {
                // System.out.println("读完了");
            }
        }

        public <T> void writeFile(T obj) throws IOException, ClassNotFoundException
        {
            List<Object> list = new ArrayList<>();
            try {
                readFile(list);
            } catch (EOFException e) {
            }
            list.add(obj);
            try (ObjectOutputStream oos = new ObjectOutputStream(new
            FileOutputStream(opFile))) {
                for (Object o :
                    list) {
                    oos.writeObject(o);
                }
                oos.close();
            }
        }
    }

```

```
}  
}
```