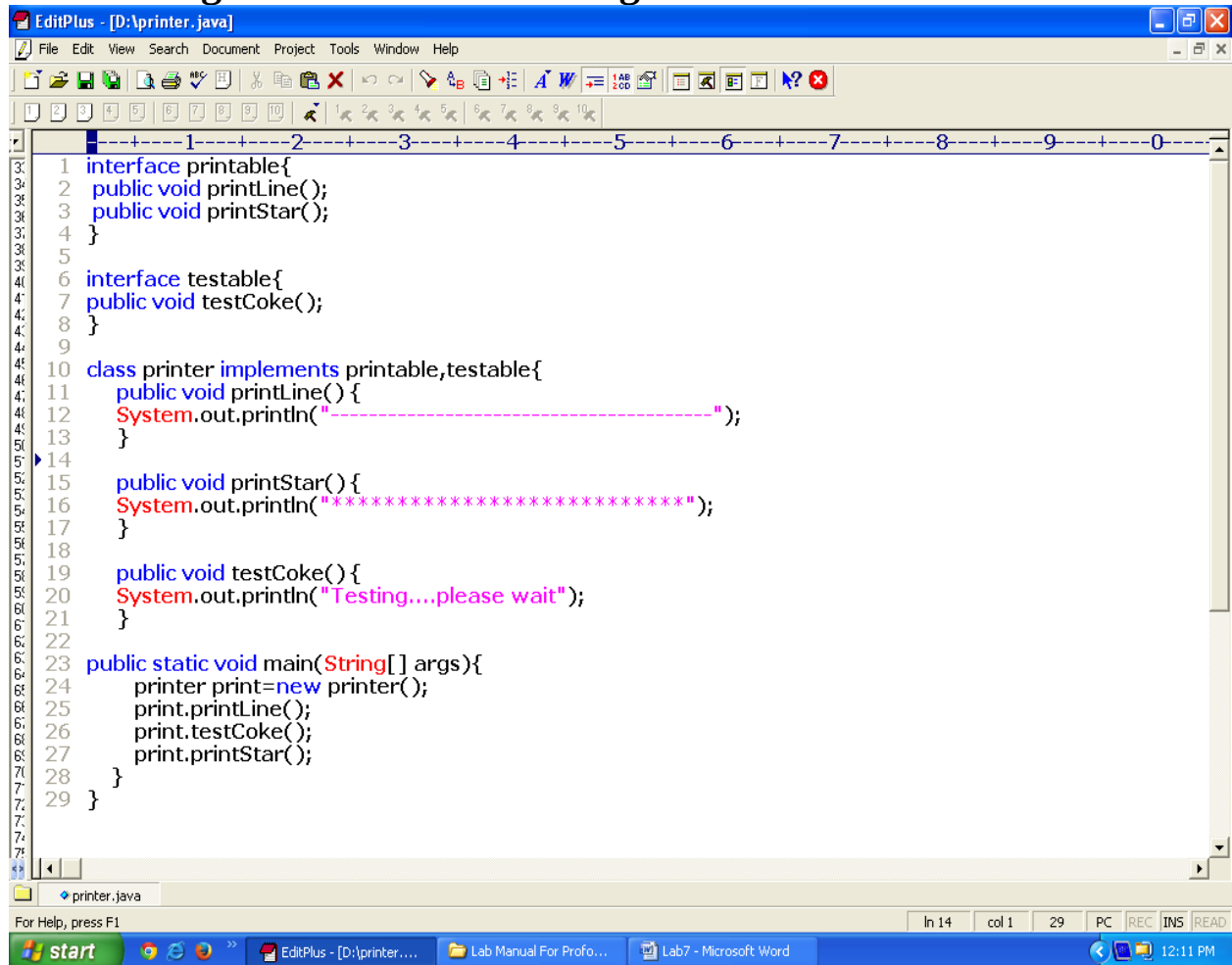


Lab-7 Assignments

Introducing Interfaces and Packages



```
1 interface printable{
2     public void printLine();
3     public void printStar();
4 }
5
6 interface testable{
7     public void testCoke();
8 }
9
10 class printer implements printable,testable{
11     public void printLine() {
12         System.out.println("-----");
13     }
14
15     public void printStar() {
16         System.out.println("*****");
17     }
18
19     public void testCoke() {
20         System.out.println("Testing....please wait");
21     }
22
23     public static void main(String[] args){
24         printer print=new printer();
25         print.printLine();
26         print.testCoke();
27         print.printStar();
28     }
29 }
```

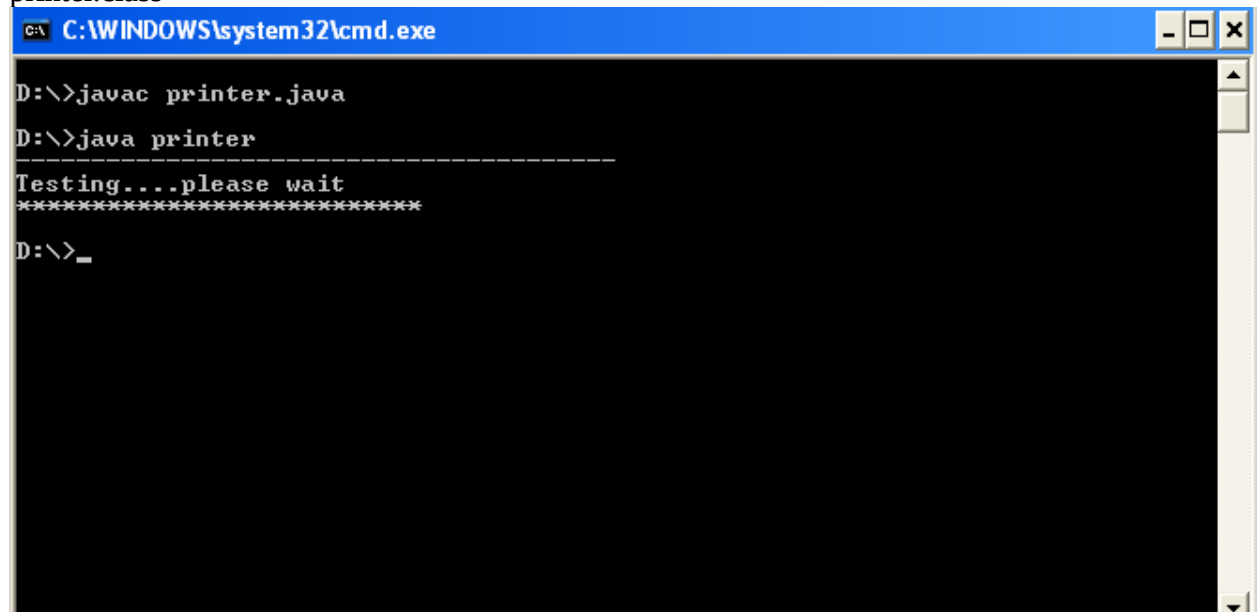
Save as printer.java

Note: three .class files get created

printable.class

testable.class

printer.class

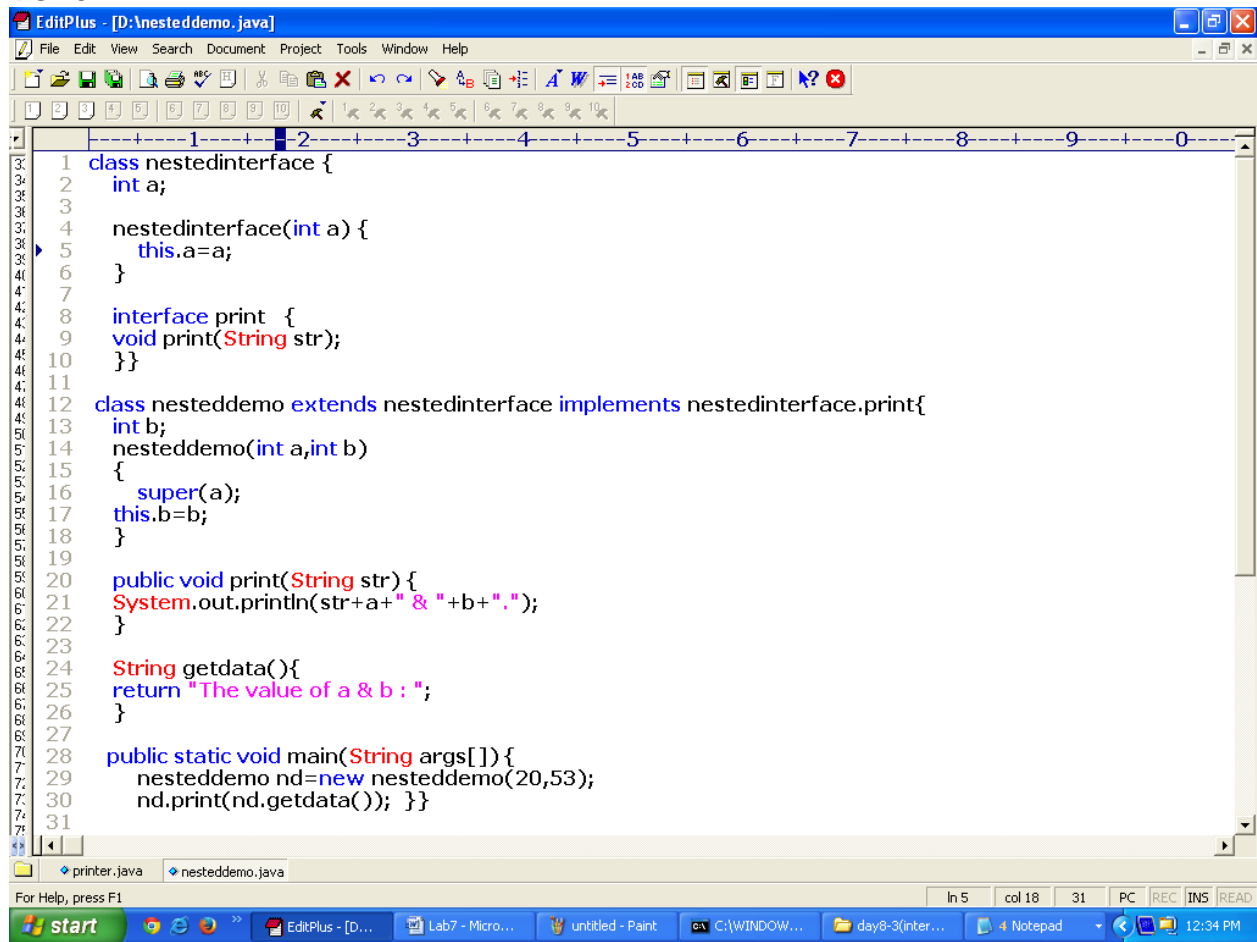


```
C:\WINDOWS\system32\cmd.exe

D:\>javac printer.java
D:\>java printer

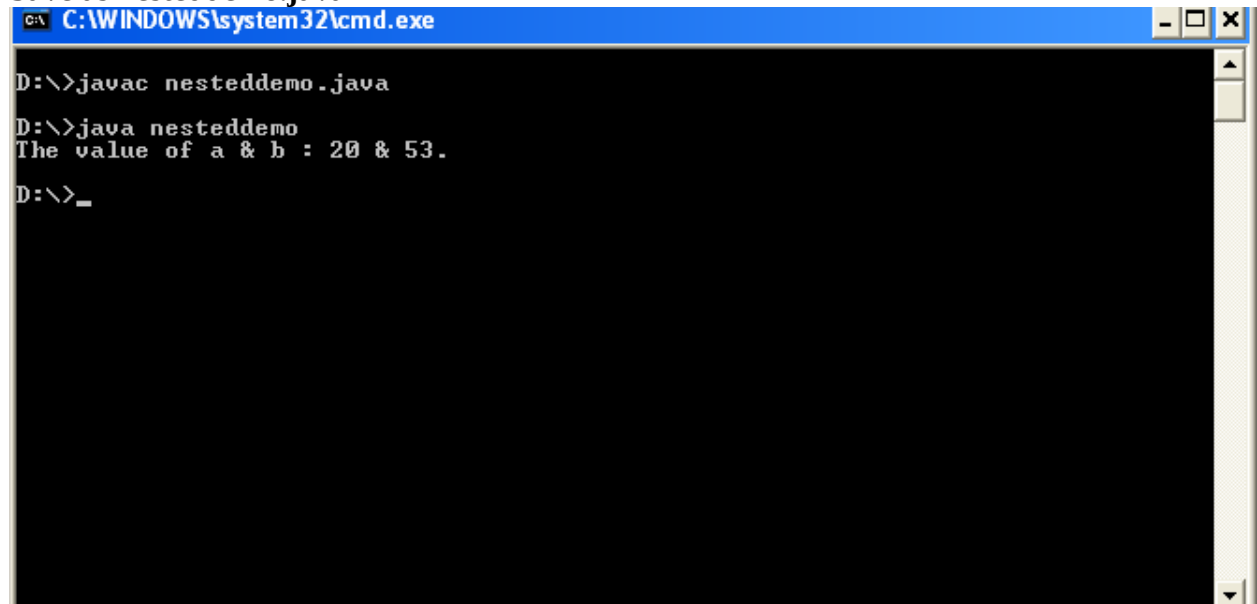
Testing....please wait
*****
D:\>_
```

Demo2



```
1 class nestedinterface {
2     int a;
3
4     nestedinterface(int a) {
5         this.a=a;
6     }
7
8     interface print {
9         void print(String str);
10    }
11
12    class nesteddemo extends nestedinterface implements nestedinterface.print{
13        int b;
14        nesteddemo(int a,int b)
15        {
16            super(a);
17            this.b=b;
18        }
19
20        public void print(String str) {
21            System.out.println(str+a+ " & "+b+ ".");
22        }
23
24        String getdata(){
25            return "The value of a & b : ";
26        }
27
28        public static void main(String args[]){
29            nesteddemo nd=new nesteddemo(20,53);
30            nd.print(nd.getdata()); }
31    }
```

Save as nesteddemo.java



```
C:\WINDOWS\system32\cmd.exe

D:\>javac nesteddemo.java

D:\>java nesteddemo
The value of a & b : 20 & 53.

D:\>_
```

Assignments To Solve

1. Create an interface relational
interface relational
{
void greaterThan();
void lessThan();
void greaterThaneq();

```
void lessThanEq();
```

```
}
```

Write an implementing class implRel which implements relational

```
class implRel implements relational
```

```
{
```

```
    int a, b;
```

```
    implRel(int a,int b)
```

```
{
```

```
    this.a=a;
```

```
    this.b=b;
```

```
}
```

```
    -   - - -
```

```
    -   - - -
```

```
}
```

Create a class relationdemo .In main create an object of implRel (implementing class) and invoke all the methods ...

2. Create an interface

```
interface MyMath
```

```
{
```

```
    double sqr(double a);
```

```
    double cube(double a);
```

```
    double cosine(double a);
```

```
    double sine(double a);
```

```
}
```

Write a class implMath which implements MyMath interface.

Write another class MathDemo ..In main create a reference of MyMath pointing to implMath class and invoke the methods..Accept the values from the user..

3. Create a package com.user .

Now create a Greeter class in this package having the following features:

Attributes:

```
    name  string  //indicates name of the person to be greeted
```

Member functions:

```
    Greeter(aName)    //constructor to initialize the name of the //person to be greeted
                      by this greeter.
```

```
    sayHello()        //returns a hello message with the name of the //person
                      initialized earlier.
```

```
    sayGoodBye()//bids goodbye to the person named earlier.
```

Create another class in the same package called Advisor that has the following features:

Attributes:

```
    message  string[5]    //contains five advice messages
```

Member functions:

```
    Advisor()          //default constructor to initialize an array of //strings with
                      atleast five advice messages
```

```
    getAdvice()        //randomly selects an advice from the available //list of
                      messages and returns it to the caller of //this method
```

Outside the package, from your working directory, create a class GreeterTest that constructs Greeter objects for all command-line arguments and prints out the results of calling sayHello().

The program should then display an advice and finally bid goodbye to each of the persons/entities in reverse order of the names entered at the command line.

For e.g.,

```
java GreeterTest Mars Venus
```

then the program should print

Hello, Mars!

Hello, Venus!

Advice: Never say No

Goodbye Venus!

Goodbye Mars!