SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23CSB101- OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Unit II – INHERITANCE, PACKAGES AND INTERFACES

Topic : PACKAGES

SNSCE/ AI&DS/ AP / Dr . N. ABIRAMI

final WITH INHERITANCE

What is final keyword in Java?

Final is a keyword or reserved word in java used for restricting some functionality. It can be applied to member variables, methods, class and local variables in Java.

- ✓ **final** keyword has three uses:
  1. For declaring variable – **to create a named constant.** A final variable cannot be changed once it is initialized.
  2. For declaring the methods – **to prevent method overriding**. A final method cannot be overridden by subclasses.
  3. For declaring the class – **to prevent a class from inheritance.** A final class cannot be inherited.

final WITH INHERITANCE

1. Final Variable:

Any variable either member variable or local variable (declared inside method or block) modified by final keyword is called final variable.

✓The final variables is equivalent **#define** directive in C.

✓**Syntax**:

```
final   data_type   variable_name = value;
```

Example:

```
final  int   MAXMARKS=100;
final  int   PI=3.14;
```

NOTE: Final variables are by default read-only.

final WITH INHERITANCE

2. Final Methods:

- ✓ Final keyword in java can also be applied to methods.
- ✓ A java method with final keyword is called **final method** and it **cannot be overridden in sub-class.**

Syntax:

```
final  return_type  function_name(parameter_list)
{
// method body
}
```

Example:

```
class Bike
{
final void run()
{
System.out.println("running");
}
 }
```

final WITH INHERITANCE

3. Final Classes:

- ✓ Java class with final modifier is called final class in Java and they cannot be sub-classed or inherited.
- ✓ **Syntax:**

```
final class class_name
{
   // body of the class
}
```

Example:

```
final class Bike
{  }
class Honda1 extends Bike
{  void run()
{  ....  }
```

✓ Several classes in Java are final e.g. String, Integer and other wrapper classes.

Packages

A Package can be defined as a collection of classes, interfaces, enumerations and annotations, providing access protection and name space management.

Package can be categorized in two form:

1. Built-in package
2. User-defined package.

Packages

Packages	Description
Java.lang	It is a default package which contain primitive data type, displaying result on console screen, obtaining garbage collector etc.
java.io	It used for developing file handling applications, such as, opening the file in read or write mode, reading or writing the data, etc.
java.awt	This package is used for developing GUI (Graphic User Interface) components such as buttons, check boxes, scroll boxes, etc.
Java. applet	This package is used for developing browser oriented applications.
java.net	This package is used for developing client server applications.
java.util	Contains utility classes which implement data structures like Hash Table, Dictionary, etc.
java.sql	This package is used for retrieving the data from data base and performing various operations on data base.

Table: List of Built-in Packages

STUCOR APP

Packages

CREATING USER DEFINED PACKAGES:

- Java package created by user to categorize their project's classes and interface are known as user-defined packages.
- Before creating a package, name for the package has to be chosen.
- Put a package statement with that name at the top of every source file that contains the classes and interfaces.
- The package statement should be the first line in the source file.
- There can be only one package statement in each source file

Syntax:

```
package package_name.[sub_package_name];
public class classname
{ …….. …….. }
```

Packages

Example:

```
package pack;
public class class1 {
public static void greet()
{
System.out.println(“Hello”);
}
}
```

To create the above package,

1. Create a directory called pack.
2. Open a new file and enter the code given above.
3. Save the file as class1.java in the directory.
4. A package called pack has now been created which contains one class class1
5. Compile the package
6. Using the Package: use the import statement to access classes in the package.

Packages

Accessing / using a package (using “import” keyword)

The import keyword is used to make the classes and interface of another package accessible to the current package.

Syntax:

import package1[.package2][.package3].classname or *;

Example:

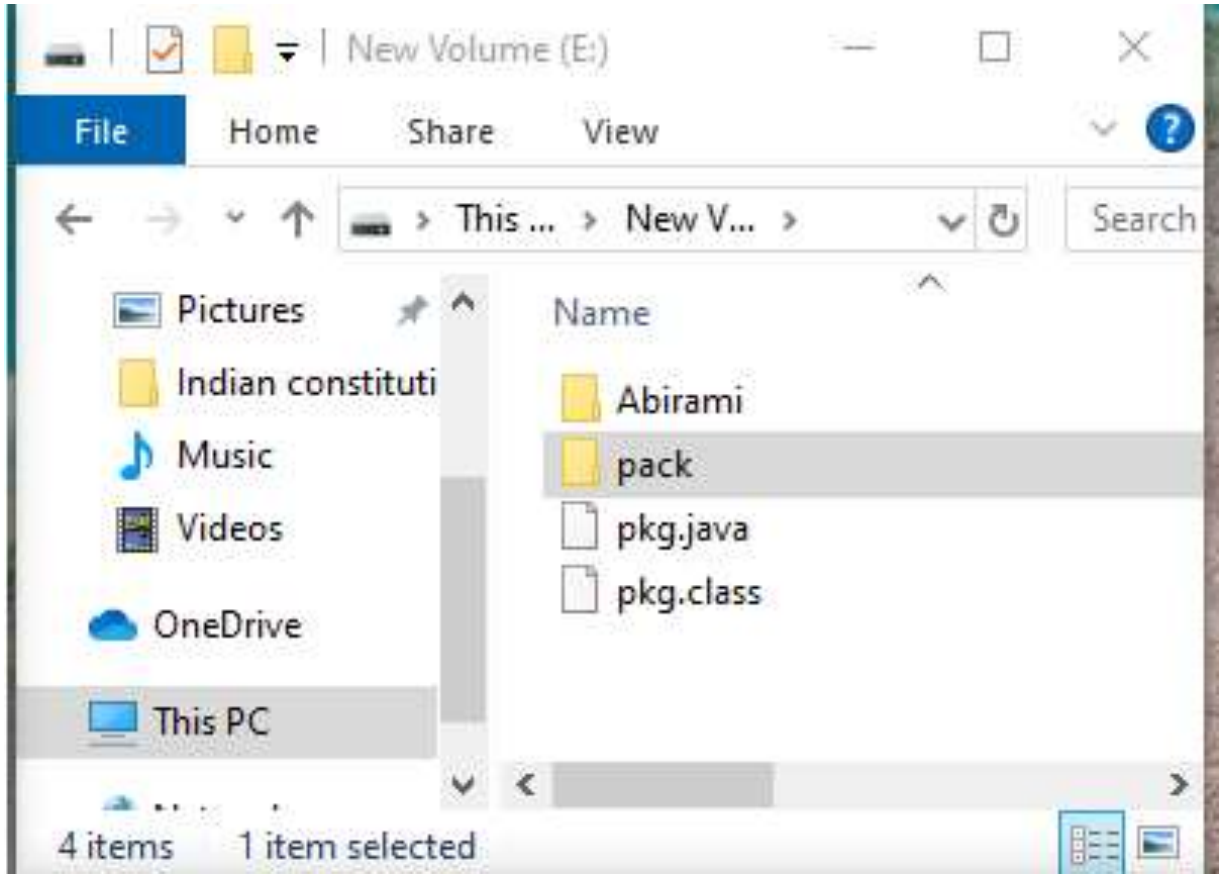
1. import package.*;
2. import package.classname;
3. fully qualified name.

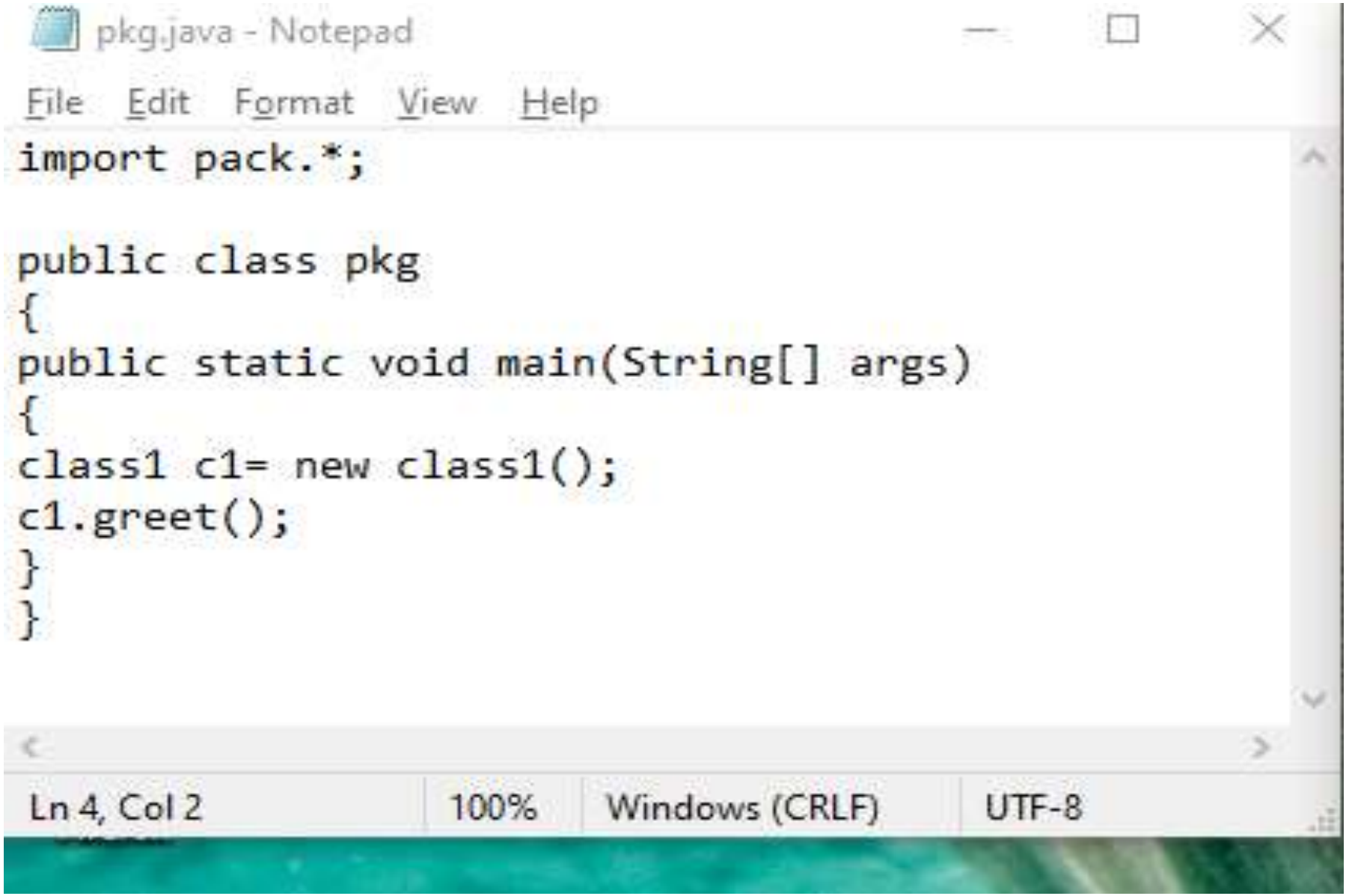
class1.java - Notepad

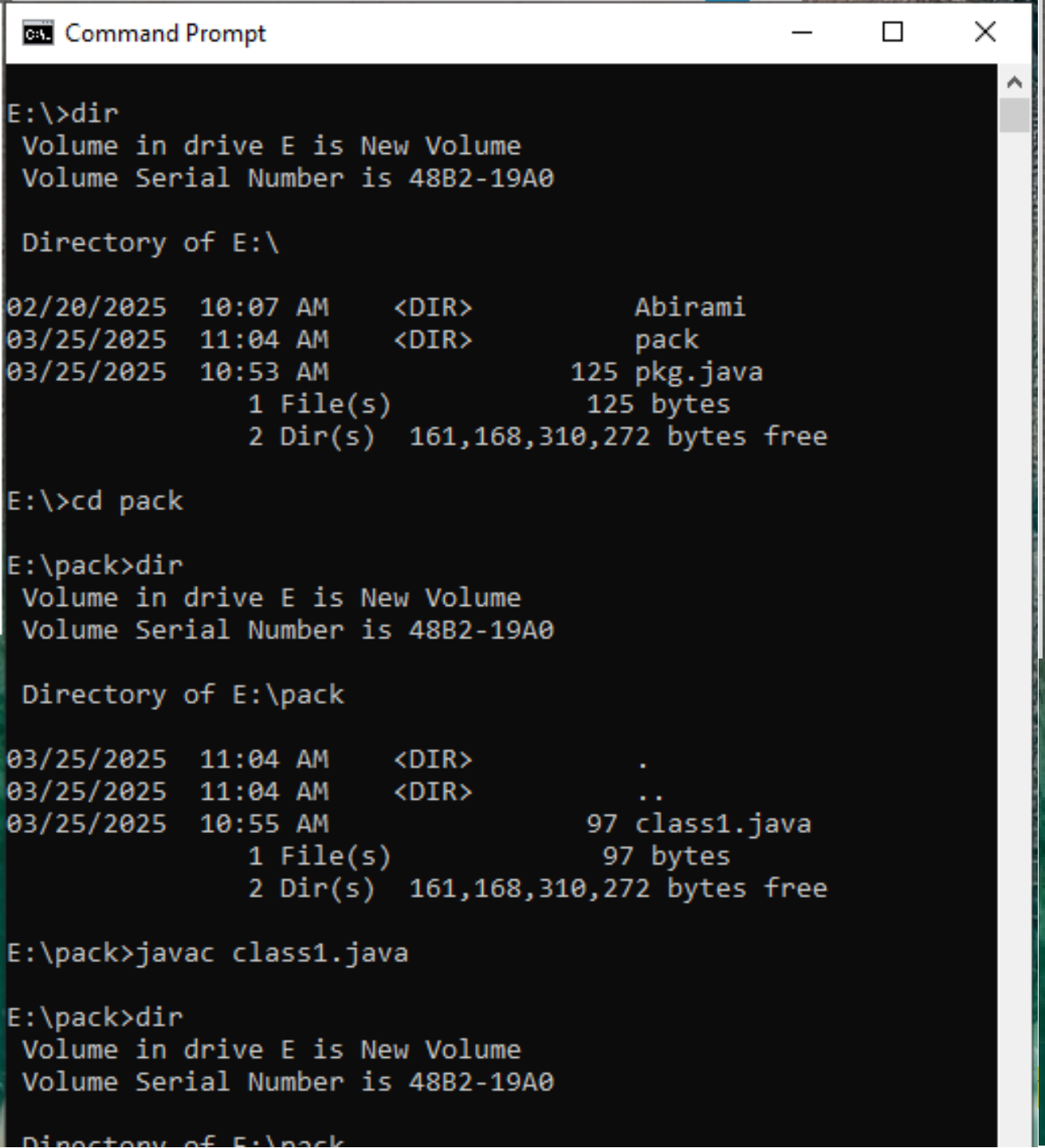
File Edit Format View Help

```
package pack;
public class class1
{
public void greet()
{
System.out.println("Hello");
}
}
```

Ln 4, Col 8 | 100% | Windows (CRLF) | UTF-8




pkg.java - Notepad
File Edit Format View Help
import pack.*;
public class pkg
{
public static void main(String[] args)
{
class1 c1= new class1();
c1.greet();
}
}
Ln 4, Col 2
100%
Windows (CRLF)
UTF-8




```
E:\pack>javac class1.java

E:\pack>dir
 Volume in drive E is New Volume
 Volume Serial Number is 48B2-19A0

 Directory of E:\pack

03/25/2025  11:05 AM    <DIR>          .
03/25/2025  11:05 AM    <DIR>          ..
03/25/2025  11:05 AM               392 class1.class
03/25/2025  10:55 AM                97 class1.java
               2 File(s)            489 bytes
               2 Dir(s)  161,168,310,272 bytes free

E:\pack>cd..

E:\>javac pkg.java

E:\>java pkg
Hello

E:\>
```

Do it yourself

Create a package mutil.

Insert the following code

```
public class MathUtils {
    public static int add(int a, int b) {
        return a + b;
    }
}
```

Create a class Main

Call the method add of package mutil

Packages

Advantage of Package:

1. Package is used to categorize the classes and interfaces so that they can be easily maintained.
2. Package provides access protection.
3. Package removes naming collision.
4. To bundle classes and interface
5. The classes of one package are isolated from the classes of another package
6. Provides reusability of code
7. We can create our own package or extend already available package


Thank
You