COLLEGE OF ENGINEERING AND MANAGEMENT, KOLAGHAT

Technical Report on

NAME: SANTU JANA
UNIVERSITY ROLL: 10700121127
COLLEGE ROLL:CSE/21/L-146

Title of the Report :- PACKAGE IN JAVA

-: CONTENTS :-

ABSTRACT:-

Program comprehension is essential for code maintenance and evolution activities. It saves time and efforts of developers who want to perform any code changes. It also minimizes the chances of introducing bugs. Textual summaries for source code provide great help to code understanding activities. This paper presents an approach to automatically generate textual summaries for services implemented in java packages. The summary is generated by analyzing the source code of methods defined the package. Each method represents a service provide by the package. Each service is summarized as a natural language textual description. The generated summary for a method mainly includes the used data and the names of invoked methods. Summaries of all methods defined in a package are refined and integrated to be reported as a comprehensive summary for the services provided by the package. The generated summaries are useful in different ways. They can be used by developers in their maintenance activities. They also can be useful for the documentation purposes.

INTRODUCTION:-

PACKAGE in Java is a collection of classes, sub-packages, and interfaces. It helps organize your classes into a folder structure and make it easy to locate and use them. More importantly, it helps improve code reusability.

Each package in Java has its unique name and organizes its classes and interfaces into a separate namespace, or name group.

Although interfaces and classes with the same name cannot appear in the same package, they can appear in different packages. This is possible by assigning a separate namespace to each Java package.

Syntax Of Package: package nameOfPackage;

DISCUSSION WITH EXAMPLES:-

How to Create a package:

Creating a package is a simple task as follows:-

- Choose the name of the package
- Include the package command as the first line of code in your Java Source File.
- The Source file contains the classes, interfaces, etc you want to include in the package
- Compile to create the Java packages

Step 1) Consider the following package program in Java:

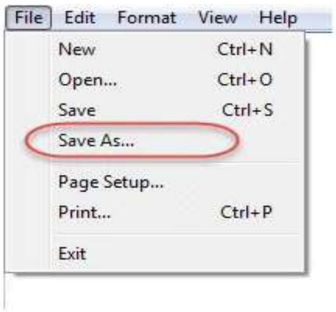
```
package p1;    //1
class c1(){      //2
public void m1(){     //3
System.out.println("m1 of c1");     //4
}
public static void main(string args[]){   //5
c1 obj = new c1();       //6
obj.m1();      //7
}
}
```

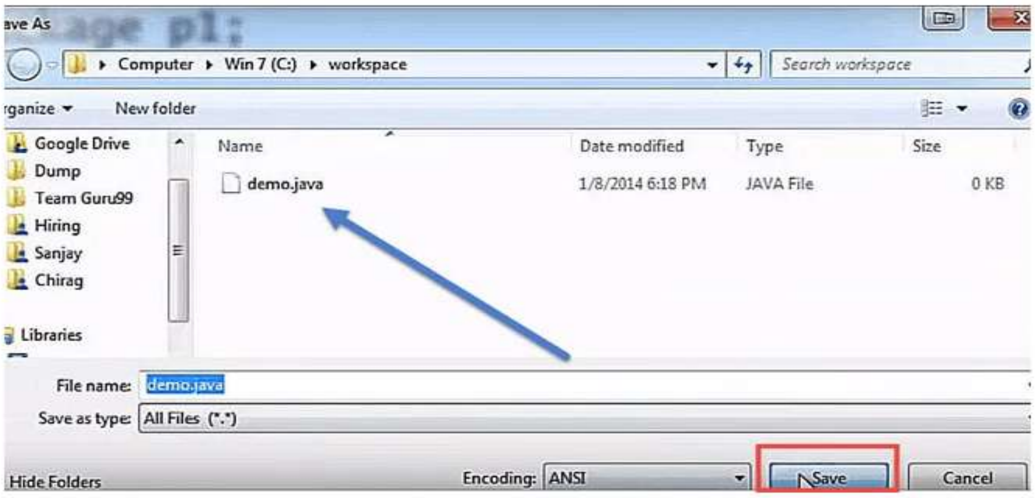
Here,

1. To put a class into a package, at the first line of code define package p1

2. Create a class c1
3. Defining a method m1 which prints a line.
4. Defining the main method
5. Creating an object of class c1
6. Calling method m1

Step 2) In next step, save this file as demo.java:

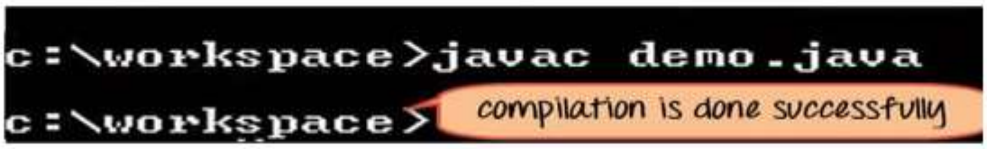




Name	Date modified	Type	Size
demo.java	1/8/2014 6:18 PM	JAVA File	0 KB

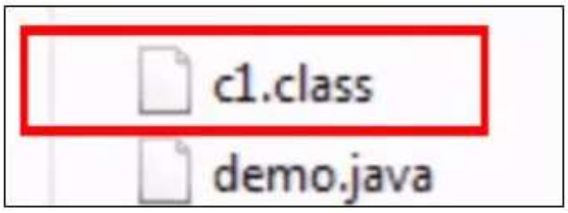
File name: demo.java
Save as type: All Files (*.*)
Encoding: ANSI

Ste



```
c:\workspace>javac demo.java
c:\workspace>
```

The compilation is completed. A class file c1 is created. However, no package is created? Next step has the solution



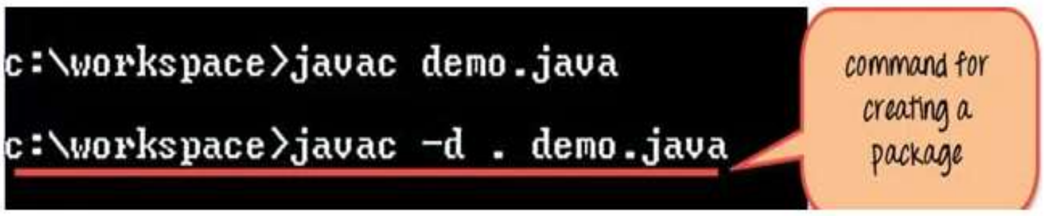
c1.class
demo.java

Step 4) Now we have to create a package, use the command:

javac –d . demo.java

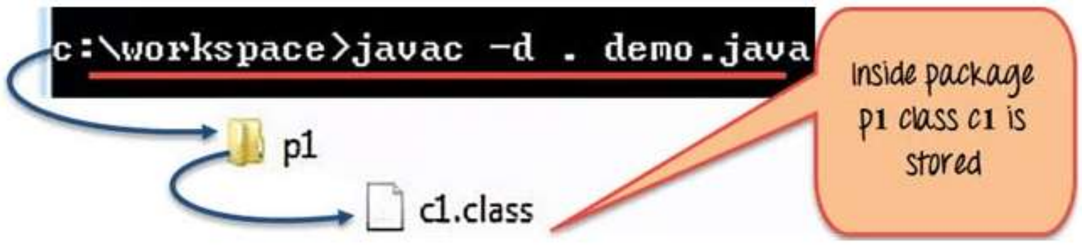
This command forces the compiler to create a package.

The **“.”** operator represents the current working directory.



```
c:\workspace>javac demo.java

c:\workspace>javac -d . demo.java
```

Step 5) When you execute the code, it creates a package p1. When you open the java package p1 inside you will see the c1.class file.

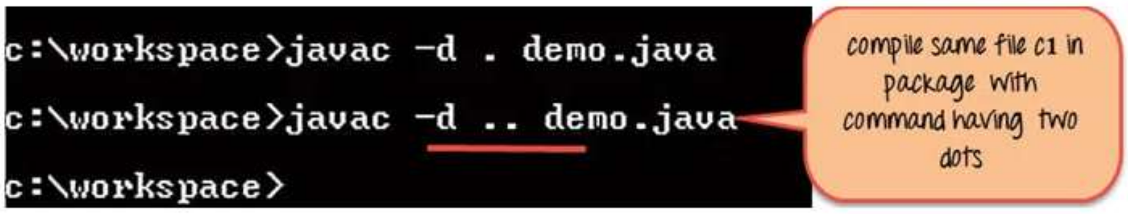


```
c:\workspace>javac -d . demo.java
```

Step 6) Compile the same file using the following code:

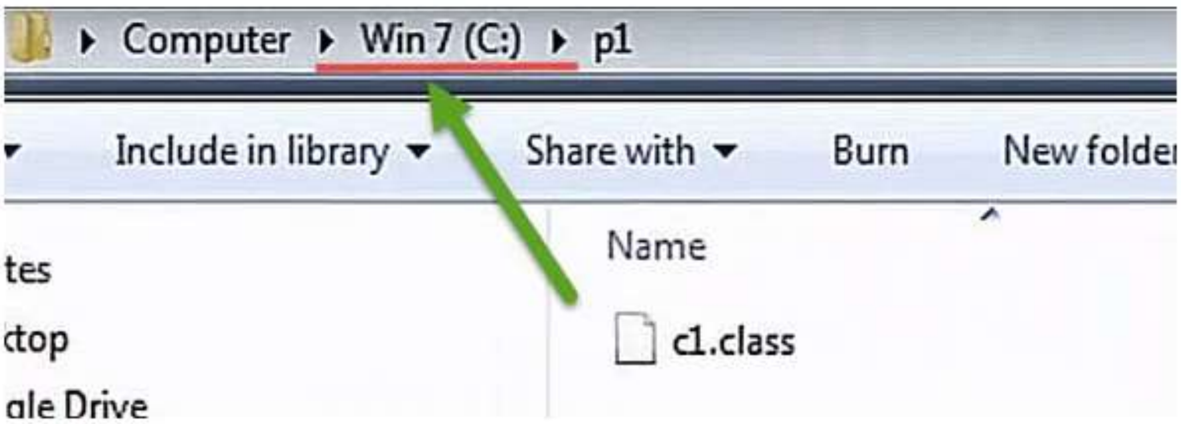
javac –d .. demo.java

Here “..” indicates the parent directory. In our case file will be saved in parent directory which is C Drive



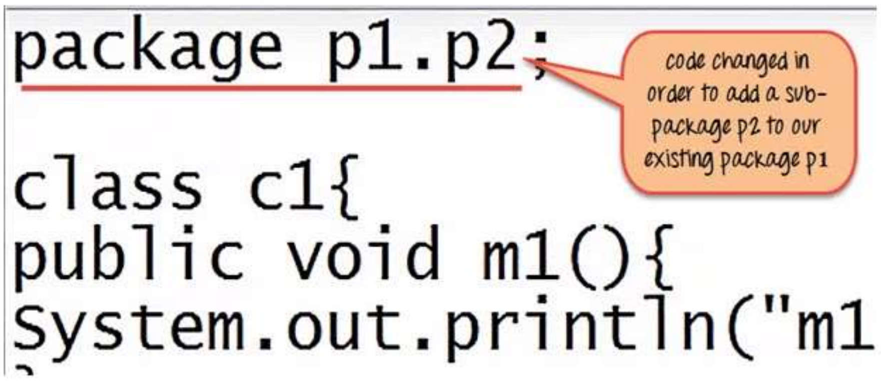
```
c:\workspace>javac -d . demo.java

c:\workspace>javac -d .. demo.java

c:\workspace>
```

compile same file c1 in package with command having two dots



Step 7) Now let’s say you want to create a sub package p2 within our existing java package p1. Then we will modify our code as:

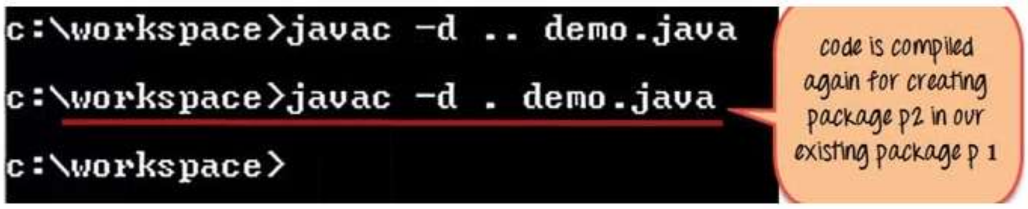
```
package p1.p2;


class c1{
public void m1() {
System.out.println("m1 of c1");
}
}
```



```
package p1.p2;

class c1{
public void m1(){
System.out.println("m1
```

code changed in order to add a sub-package p2 to our existing package p1

Step 8) Compile the file:

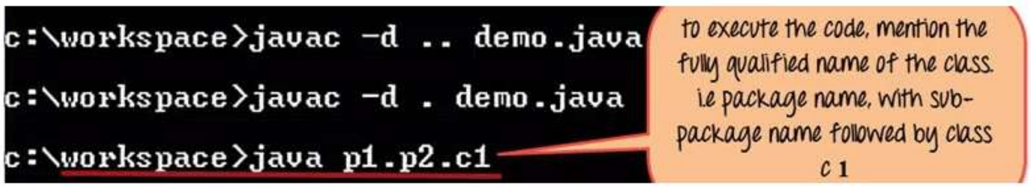


```
c:\workspace>javac -d .. demo.java

c:\workspace>javac -d . demo.java

c:\workspace>
```

code is compiled again for creating package p2 in our existing package p 1

Step 9) To execute the code mention the fully qualified name of the class i.e. the package name followed by the sub-package name followed by the class name –

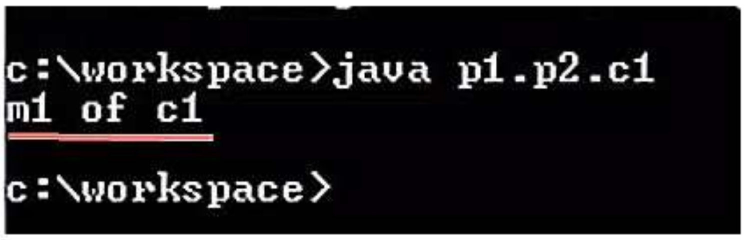
```
java p1.p2.c1
```



```
c:\workspace>javac -d .. demo.java

c:\workspace>javac -d . demo.java

c:\workspace>java p1.p2.c1
```

to execute the code, mention the fully qualified name of the class. ie package name, with sub-package name followed by class c 1

This is how the package is executed and gives the output as "m1 of c1" from the code file.



```
c:\workspace>java p1.p2.c1
m1 of c1

c:\workspace>
```

How to Import Package:

To create an object of a class (bundled in a package), in your code, you have to use its fully qualified name.

Example:

java.awt.event.actionListner object = new java.awt.event.actionListner();

But, it could become tedious to type the long dot-separated package path name for every class you want to use. Instead, it is recommended you use the import statement.

Syntax:-

```
import packageName;
```

Once imported, you can use the class without mentioning its fully qualified name.

```
import java.awt.event.*; // * signifies all classes in this package are imported
import javax.swing.JFrame // here only the JFrame class is imported
//Usage
JFrame f = new JFrame; // without fully qualified name.
```

Example: To import package

Step 1) Copy the code into an editor.

```
package p3;
import p1.*; //imports classes only in package p1 and NOT  in the sub-package
p2
class c3{
  public   void m3(){
     System.out.println("Method m3 of Class c3");
  }
  public static void main(String args[]){
    c1 obj1 = new c1();
    obj1.m1();
  }
}
```

Step 2) Save the file as Demo2.java. Compile the file using the command **javac –d . Demo2.java**

Step 3) Execute the code using the command **java p3.c3**

Packages – points to note:

- To avoid naming conflicts packages are given names of the domain name of the company in reverse Ex: com.guru99. com.microsoft, com.infosys etc.
- When a package name is not specified, a class is in the default package (the current working directory) and the package itself is given no name. Hence you were able to execute assignments earlier.
- While creating a package, care should be taken that the statement for creating package must be written before any other import statements

```
// not allowed
import package p1.*;
package p3;

//correct syntax
package p3;
import package p1.*;
```

the *java.lang package* is imported by default for any class that you create in Java.

CONCLUSION:

The approach automatically parses the source code of classes and methods inside the package. Then it extracts useful information from the parsed code and used them to generate natural language summary for the given package. The summary is focused on the services provided by the package. The approach is a light weight approach in which to complex processing is needed to generate the summary. The proposed approach helps in program comprehension tasks. It helps developers to get a brief description about the service provided by the method before examining its implementation. Another benefit is the automatic documentation of the source code, especially the behavior of methods. The summaries can be useful for educational purposes to help students in implementing functional requirements. We are currently working on realizing the approach as an Eclipse plug-in tool. We are also working on improving the generated summary by including more descriptive information about the importance of the service, how it is connected to other services and its impact on external services.

REFERENCE:-

1) https://www.javatpoint.com/java-tutorial
2) https://www.guru99.com/java-packages.html
3) OOPS IN JAVA E-balaguruswami book