Subject Name: Object Oriented Programming using JAVA - QUESTION BANK SOLUTION
Subject Code: 2150704 Unit-5 And 9

Faculties: MS. HEMALI MOJIDRA (SHAH)

UNIT-5 PACKAGES

Use of Package, CLASSPATH, Import statement, Static import, Access control

1 Explain package in java. List out all packages with short description. (**June-12) [LJIET]** **3/4/7**
OR What is package? List various built in package used in java. (**May-15) [LJIET]**

Ans :

Java Package :

- package is a **group of similar types of classes, interfaces and sub-packages.**
- Package in java can be categorized in two form
  - **built-in package -** java, lang, awt, javax, swing, net, io, util, sql etc.
  - **user-defined package.**

Advantage of Java Package:

- Java package is used to **categorize the classes** and interfaces so that they can be easily maintained.
- Java package provides **access protection**.
- Java package **removes naming collision**. You can **create class having same name** but they should be **in different package**.

Simple example of java package:

The **package keyword** is used to create a package in java.

```
//save as LJCE.java
package ljce;
public class LJCE {
 public static void main(String args[]){
   System.out.println("Welcome to package ljce ");
  }
}
```

How to run java package program

You need to use fully qualified name e.g. **ljce.LJCE** etc to run the class.

```
Output:Welcome to package ljce
```

How to access package from another package?

There are three ways to access the package from outside the package.

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

1) Using packagename.*

If you use package.* then all the classes and interfaces of this package will be accessible but

not subpackages.

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

Example of package that import the packagename.*

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//save by B.java
package mypack;
import pack.*;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

```
Output:Hello
```

2) Using packagename.classname

If you import package.classname then only declared class of this package will be accessible.

Example of package by import package.classname

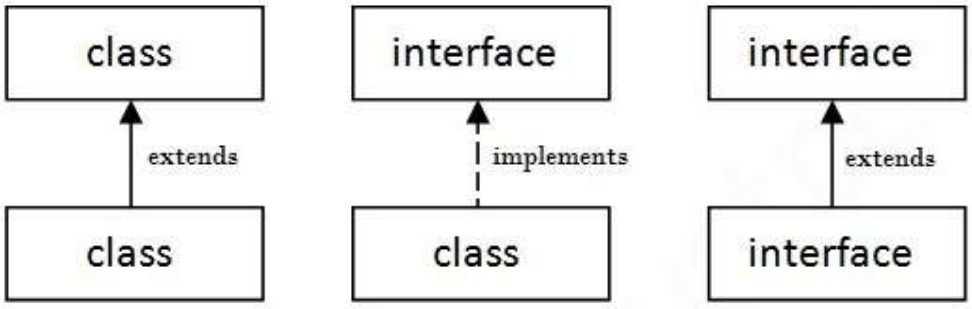
//save by A.java

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

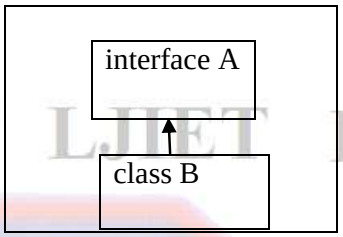
Built-in Packages

These packages consists of a large number of classes which are a part of Java **API**. For e.g, we have used **java.io** package previously which contain classes to support input / output operations in Java. Similarly, there are other packages which provides different functionality. Some of the commonly used built-in packages are shown in the table below :

Package Name	Description
java.lang	Contains language support classes (for e.g classes which defines primitive data types, System class, math operations, etc.) . This package is automatically imported.
java.io	Contains classes for supporting input / output operations.
java.util	Contains utility classes which implement data structures like Linked List, Hash Table, Vector, Dictionary, etc and support for Date / Time operations.
java.applet	Contains classes for creating Applets. –GUI programming

	java.awt	Contains classes for implementing the components of graphical user interface (like buttons, menus, etc.).	
	java.net	Contains classes for supporting networking operations and network programming	
2	What is package? Explain steps to create package with example. **(Dec-13,May-16) [LJIET]** **Ans: Ans -1**		**3/4/7**
3	Explain package and interface by giving examples. **(June-11, June-14) [LJIET] OR** (ii) Explain packages. **(Dec-15) [LJIET]** **Ans: Package – Ans-1** **Intrface:** An **interface in java** is a blueprint of a class. It is fully abstract class. It has static constants(final) and abstract methods only. Interface **fields are public, static and final** by default, and methods are **public and abstract**. The interface in java is **a mechanism to achieve fully abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java. Java Interface also **represents IS-A relationship**. It cannot be instantiated just like abstract class.(OR It is not possible to create object of interface) **Use Of Java interface:** • It is used to achieve fully abstraction. • It is used to give basic structure to the child classes. • By interface, we can support the functionality of **multiple inheritance**. • It can be used to achieve loose coupling. (Overriding and dynamic method dispatch and runtime polymorphism) ***Note : The java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members.*** **Understanding relationship between classes and interfaces**  Example:	**3/4/7**

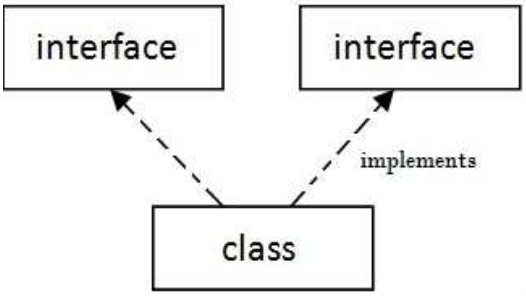
```
interface A {
        int i=10;
        void m1();
        void m1(int a);
        void m2();
}
class B implements A
{
        public void m1() {
           System.out.println("m1 called");
        }
        public void m1(int a) {
           System.out.println("m1(int) called");
        }
        public void m2() {
            System.out.println("m2 called");
        }
}
class UseInerfaceDemo
{
        public static void main(String[] args)
        {
           A a = new B();
           a.m1();
           a.m1(6);
           a.m2();
        }
}
```



Output:

```
m1 called
m1(int) called
m2 called
```

Multiple inheritance in Java by interface




```
interface A1 {
        int i=10;
        void m1();
        void m1(int a);
        void m2();
}
interface A2 {
        void m3();
```

```
}
class B implements A1,A2
{
        public void m1() {
           System.out.println("m1 called");
        }
        public void m1(int a) {
           System.out.println("m1(int) called");
        }
        public void m2() {
            System.out.println("m2 called");
        }

        public void m3() {
           System.out.println("m3 called");
        }
}
class UseInerfaceDemo
{
        public static void main(String[] args)
        {
           B b = new B();
           b.m1();
           b.m1(6);
           b.m2();
           b.m3();
        }
}
```