1.4.12.PACKAGES

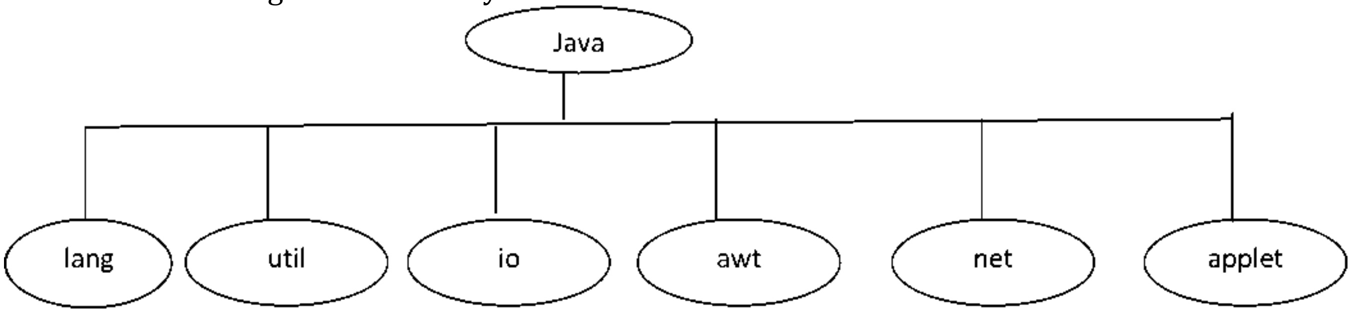
- Grouping mixture of classes and/or interfaces collectively
- Grouping is generally done according to functionality
- Packages act as containers for classes

Types:

1. Java API packages
2. User defined packages

Java API Packages

• Java API provides a large number of classes integrated in to various packages according to functionality



Package Name	Content
Java.lang [Primitive types, Strings, Mathfunctions, Threads]	Language Support class
Java.util [vectors,hash tables,random numbers,date]	Language utility classes
Java.io [input & output]	Input/output support classes
Java.awt [windows,buttons,lists,menus]	Set of classes for implementing GUI
Java.net	Classes for networking
Java.applet	Classes for creating and implementing applets

Using System Packages

There are 2 ways of accessing the classes available in a package

1. First Approach : fully qualified name

   Java.awt.color;

• Imports the color class & class name can now be openly used in the program.

2. Second Approach: Once or when we do not want to access any other classes of the package

• bring all the classes of java.awt package.

Use a class in number of times in a program/like to use different classes enclosed in a package.

import packagename.classname;

or

import packagename.*;

Naming Conventions

Can be named using standard naming rules

1. packages start with lowercase letters.
2. class names start with uppercase letters.
3. Methods start with lowercase letters.
   Ex: double y=java.lang.Math.sqrt(x)
   java.lang : package
   Math: class name
   sqrt: method name.

Benefits of Packages

- The classes enclosed in the packages of other programs can be simply reused
- They give a way to "hide" classes thus preventing from new programs or packages
- Also supply a way for separating "design" from "coding"
- Two different classes in 2 various packages can have similar name.

Java user defined package

Creating a user defined Package

- First declare the package name using the package keyword continued by a package name
- This must be the initial statement in a java source file.
- Then you can define a class just as we usually define a class

```
package firstpackage;//package declaration
public class Firstclass//class definition
{
Body
}
```

Creating our own package or user defined packages follows the following steps

1. Declare the package at the starting of a file
   package packagename;
2. Define the class that is to be place in the package & declare it public.
3. Create a subdirectory below the directory where the main source files are stored
4. Keep the listing as the classname.java file in the subdirectory created
5. Compile the file.This generates class file in the subdirectory.

- Java also provides the concept of package hierarchy
- This is done by specifying many names in a package statement,separated by dots.
  package firstpackage.secondpackage;

Accessing a Package

- In java programming package can be accessed either using a fully qualified class name or using another shortcut method through the import statement.
- The general form of import statement
  import package1[.package2][.package3].classname;
- The system must end with a semicolon(;)
- The import statement should become visible before any class definitions in a source file.

' Ex:

Importing a particular class
import firstpackage.secondpackage.Myclass;

- After defining, all the fields of the class Myclass can be straightly accessed using the class name or its objects can be used directly without specifying the package name.

Ex:
import packagename.*;

- May denote a single package or a hierarchy of packages. * represents that the compiler should look for this whole hierarchy when it encounters a class name

Example:

```
package package1;
public class ClassA
{
public void displayA()
{
System.outprintln("Class A”);
}
}

import package.ClassA;
class Test
{
public static void main(String args[])
{
ClassA objectA=new ClassA();
objectA.display();
}
}
```

Output:

ClassA

Example

```
package package2;
public class ClassB
{
protected int m=10;
public void display()
{
System.outprmtln("Class B”);
System.outprmtln("m=”+m);
}
}
```

Example:

```
import package1.ClassA;
import package2.*;
class Test2
{
public static void main(String args[])
{
ClassA objA=new ClassA();
ClassB objB=new ClassB();
objA.displayA();
objB.displayB();
}
}
}
```

Output:

```
ClassA
ClassB
m=10
```

Example

```
import package.ClassB;
class ClassC extends ClassB
{
int n=20;
void displayC()
{
System.outprmtln("Class C”);
System.outprmtln("m=”+m);
System.outprmtln("n=”+n);
}
}

class Test2
{
public static void main(String args[])
{
ClassC objC=new ClassC();
objC.displayB();
objC.displayC();
}
}
```

Output

```
ClassB
M=10
ClassC
m=10
n=20
```

1.4.13. JAVADOC COMMENTS

TYPES

- Class Comments
- Method Comments
- Field Comments
- Package & Overview Comments

Sample tags of javadoc:

@author
@exception
@deprecated
{@link}
@param
@return

Example:

```
/**
* <h1>Hello, World!</h1>
* The HelloWorld program implements an application that
* simply displays "Hello World!" to the standard output.
* <p>
* Giving proper comments in your program makes it more
* user friendly and it is assumed as a high quality code.
*

*
* @author Zara Ali
* @version 1.0
* @since 2014-03-31
* /
public class HelloWorld {

    public static void main(String[] args) {
       /* Prints Hello, World! on standard output.
       System.out.println("Hello World!");
    }

}
```

2 MARKS QUESTIONS AND ANSWERS

1. What is Programming Paradigm?

The word paradigm to mean "any example or model". Object-oriented programming paradigm suggests new ways of thinking for finding a solution to a problem. Hence the programmers should keep their mind tuned in such a manner that they are not to be blocked by their preconceptions experienced in other programming languages such as structured programming. Proficiency in object-oriented programming requires talent, creativity, intelligence, logical thinking and the ability to build and use abstractions and experience.

2. What is Object Oriented Programming?

Object-Orientation is a set of tools and methods that enable software engineers to build reliable, user friendly, maintainable, well documented, reusable software systems that fulfills the requirements of its users.

3. What are important features in object-oriented programming and design?

- Improvement over the structured programming paradigm.

- Emphasis on data rather than algorithms.
- Procedural abstraction is complemented by data abstraction.
- Data and associated operations are unified, grouping objects with common attributes, operations and semantics.

4. **How could Java classes direct program messages to the system console, but error messages, say to a file?**

The class System has a variable out that represents the standard output, and the variable err that represents the standard error device. By default, they both point at the system console. This how the standard output could be re-directed: Stream st = new Stream(new FileOutputStream("output.txt")); System.setErr(st); System.setOut(st);

5. **What's the difference between an interface and an abstract class?**

An abstract class may contain code in method bodies, which is not allowed in an interface. With abstract classes, you have to inherit your class from it and Java does not allow multiple inheritance. On the other hand, you can implement multiple interfaces in your class.

6. **Why would you use a synchronized block vs. synchronized method?**

Synchronized blocks place locks for shorter periods than synchronized methods.

7. **Explain the usage of the keyword transient?**

This keyword indicates that the value of this member variable does not have to be serialized with the object. When the class will be de-serialized, this variable will be initialized with a default value of its data type (i.e. zero for integers).

8. **Define object and object variable ?(Apr/May 2011)(May/June 2013)**

An Objectis anything having crisply defined conceptual boundaries. Book, pen, train, employee, student, machine, etc., are examples of objects.
•Objects in java are created using the new operator
•New operator creates an object of the specified class and returns a reference to that object.
Ex:
Rectangle rect1;//declare
Rect1=new Rectangle();/instantiate
•First statement declares a variable to hold the object reference & the second one actually assigns the object reference to the variable.

9. **What is meant by private access specifier?Nov/Dec 2011**

Private members are accessible only within the class and cannot be inherited.

10. **What is the need for javadoc multiline commentsNov/Dec 2011**

To describe the program multiline comments are used. /*.... */

11. **Define Constructor.Nov/Dec 2011**

- Constructor enables the object to initialize itself.
- Constructor also can not have any return type, constructor's are automatically chained by using this keyword and super.

- Two Types.Default Constructor and Parameterized constructor

12. What is a class? (Nov/Dec 2011)
Class is a data type. It generates object. It is the prototype or model. It does not occupy memory location. It cannot be manipulated because it is not available in the memory.

13. What are the fundamental features of object-oriented programming?
- Encapsulation
- Data Abstraction
- Inheritance
- Polymorphism
- Extensibility
- Persistence
- Delegation
- Genericity
- Object Concurrency
- Event Handling
- Multiple Inheritance
- Message Passing

14. What is Encapsulation?
The process, or mechanism, by which you combine code and the data it manipulates into a single unit, is commonly referred to as *encapsulation*. Encapsulation provides a layer of security around manipulated data, protecting it from external interference and misuse.

15. What is a Type?
A type is a set of values together with one or more operations that can be applied uniformly to all these values.

16. What is a class Members?
The non-static members of a class (variables and methods) are also known as instance variables and methods while the non-static members are also known as class variables and class methods.

17. What is meant by Access Specifiers?
The purpose of access specifiers is to declare which entity cannot be accessed from where. Its effect has different consequences when used on a class, class member (variable or method), constructor.

18. Explain the Static Class Method.
Static methods typically take all they data from parameters and compute something from those parameters, with no reference to variables. This is typical of methods which do some kind of generic calculation. A good example of this are the many utility methods in the predefined Math class.

19. Explain the accessing of static call method.
A common use of static variables is to define "constants". Examples from the Java library are

Math.PI or Color.RED. They are qualified with the class name, so you know they are static. Any method, static or not, can access static variables. Instance variables can be accessed only by instance methods.

20. Explain the Constructor method in Java.
When you create a new instance (a new object) of a class using the new keyword, a *constructor* for that class is called. Constructors are used to initialize the instance variables (fields) of an object.

21. How to Write a Finalize method using Java.(May/June 2013)
Before an object is garbage collected, the runtime system calls its finalize() method. The intent is for finalize() to release system resources such as open files or open sockets before getting collected. Your class can provide for its finalization simply by defining and implementing a method in your class named finalize(). Your finalize() method must be declared as follows:
protected void finalize () throws throwable

22. What is an array? How to declare an array in java.

Array is the most important thing in any programming language. By definition, array is the static memory allocation. It allocates the memory for the same data type in sequence. It contains multiple values of same types. It also store the values in memory at the fixed size. Multiple types of arrays are used in any programming language such as: one - dimensional, two - dimensional or can say multi - dimensional.
Declaration of an array:
int num[]; or int num = new int[2];

23. What is the use of Strings in Java?
The String class is commonly used for holding and manipulating strings of text in Java programs. It is found in the standard java.lang package which is automatically imported, so you don't need to do anything special to use it.

24. Explain about String Tokenizer?
*StringTokenizer*class objects may be created by one of three constructor methods depending on the parameters used. The first parameter string is the source text to be broken at the default set of *whitespace* delimiters (space, tab, newline, cr, formfeed).

25. Explain the Java Packages.
A *package* is a grouping of related types providing access protection and name space management. Note that *types* refer to classes, interfaces, enumerations, and annotation types. Enumerations and annotation types are special kinds of classes and interfaces, respectively, so *types* are often referred to in this lesson simply as *classes and interfaces*.

26. What is JavaDoc? (Nov/Dec 2011)
Javadoc is a convenient, standard way to document your Java code. Javadoc is actually a special format of comments. There are some utilities that read the comments, and then generate HTML document based on the comments. HTML files give us the convenience of hyperlinks from one document to another. Most class libraries, both commercial and open source, provide Javadoc documents.

27. Mention the types of the JavaDoc.
There are two kinds of Javadoc comments: class-level comments, and member-level comments. Class-level comments provide the description of the classes, and member-level comments describe the purposes of the members.

28. Mention the purpose of finalize method.May/June 2013
Reclaim the Object, its last chance for any object to perform cleanup activity i.e. releasing any system resources held, closing connection if open etc

29. How to access member variables from the class?
ObjectName.VariableName=Value;
ObjectName.MethodName(Parameters List);

30. Explain about Java I/O Package?

The Java I/O Package (java.io) provides a set of input and output streams used to read and write data to files or other input and output sources. The classes and interfaces defined in java.io are covered fully in Input and Output Streams.

31. Explain about Java Utility Package?
This Java package, java.util, contains a collection of utility classes. Among them are several generic data structures (Dictionary, Stack, Vector, Hashtable) a useful object for tokenizing a string and another for manipulating calendar dates. The java.util package also contains the Observer interface and Observable class, which allow objects to notify one another when they change. The java.util classes aren't covered separately in this tutorial although some examples use these classes.

32. Explain about Applet Package?
This package contains the Applet class -- the class that you must subclass if you're writing an applet. Included in this package is the AudioClip interface which provides a very high level abstraction of audio. Writing Applets explains the ins and outs of developing your own applets.

33. Explain about the Java Packages?(NOV/DEC2010)
Several packages of reusable classes are shipped as part of the Java development environment. Indeed, you have already encountered several classes that are members of these packages: **String, System, and Date**, to name a few. The classes and interfaces contained in the Java packages implement various functions ranging from networking and security to graphical user interface elements.

13 MARK QUESTIONS

1. Explain OOP Principles.
2. Explain the features of Java Language.
3. Compare and Contrast Java with C.
4. Compare and Contrast Java with C++.
5. Explain Constructors with examples.
6. Explain the methods available under String and String Buffer Class.
7. Explain the Date Class methods with examples.

8. Discuss in detail the access specifiers available in Java.
9. Explain the different visibility controls and also compare with each of them.
10. Explain the different methods in java.Util.Arrays class with example.
11. Explain Packages in detail.
12. Discuss the methods under Array Class.

PART-C (15 MARKS)

13. Discuss some of the classes available under Lang package and develop your own applications..
14. Illustrate with examples: static and final.
15. Explain method overriding with example program.
16. What is javaDoc? Explain the comments for classes, methods, fields and link.
17. Develop library application Programs using fundamental concepts in Java.