Question Bank for M.S.B.T.E Exam for Winter 2015
Chapter 3
Interfaces and packages

1) What is package?Why packages are needed?How to add class to a package.(8 Marks)

Ans

Package:

Packages are used for grouping a variety of classes & interfaces together.This grouping is done according to functionality. Packages acts as containers of classes. Packages are stored in hierarchical manner & are explicitly imported into new class definitions.

Need of Packages:

- **To removes class naming collision:** Two classes can have the same name, provided they are in different packages. So, different programmers don't have to be concerned about using names already used in other packages
- To **reuse already written classes** using import statement.
- To **categorize** the classes and interfaces so that they can be easily maintained.
- Java package provides **access protection**.
- Packages provide way to **hide classes** thus preventing other programs or packages from accessing classes that are meant for internal use only

To add a class to user defined package :

a)Add the package statement as the first statement before the class definition.
b)Save the file in the folder that contains other classes of that package.
c)Compile the file to generate the class file.
d)Now the new class is added into the package and can be imported into the other programs.
e) Example :

```
package MyPack;
public class Balance
{
.
}
```

Then class Balance in included inside a user defined package,MyPack.
import MyPack.*; or import MyPack.Balance will import the Balance class. ;

2) Explain any four system packages along with their use ?(4 Marks)

Ans

System packages with their use:(Any 4 can be considered)

1. java.lang - language support classes. These are classes that java compiler itself uses and therefore they are automatically imported. They include classes for primitive types, strings, math functions, threads and exceptions.
2. java.util – language utility classes such as vectors, hash tables, random numbers, date etc.
3. java.io – input/output support classes. They provide facilities for the input and output of data
4. java.awt – set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on.

5. java.net – classes for networking. They include classes for communicating with local computers as well as with internet servers.
6. java.applet – classes for creating and implementing applets

3) How do we create it and access package in java?(4 Marks)
Ans
Creating package involves following steps:
1. Declare the package at beginning of a file using the form
package packagename;
2. Define the class that is to be put in the package & declare it public
3. Create a subdirectory under directory where main source files are stored
4. Store listing as the classname.java file in the subdirectory created.
5. Compile the file. This creates .class file in the subdirectory.

Example
First declare name of package using package keyword followed by package name. This must be first statement in a java source file. Here is an example:

```
package firstPackage; //package declaration
public class FirstClass //class definition
{
…..
….. (body of class)
…..
}
```

Here the package name is firstPackage. The class FirstClass is now considered a part of this package. This listing would be saved as file called FirstClass.java & located in a director named firstPackage. When source file is complied, Java will create a .class file & store it in same directory..class must be located in a directory that has same name as the package, & this directory should be a subdirectory of the directory where classes that will import package are located.

Accessing a package
1. The import statement can be used to search a list of packages for a particular class. The general form of import statement for searching a class is as follows:
Import package1 [.package2][.package3].classname;

Here package1 is the name of the top level package, package2 is the name of package that is inside the package 1, and so on. We can have number of packages in a package hierarchy. Finally, the explicit classname is specified.

Note that the statement must end with a semicolon (;). The import statement should appear before any class definitions in a source file. Multiple import statements are allowed.

2. We can also use another approach as follows:
Import packagename.*;

Here packagename may denote a single package or a hierarchy of packages as mentioned earlier. The star(*) indicates that the compiler should search this entire package hierarchy when it encounters a class name.this implies that we can access all classes contained in the above package directly.

Example:

```
package MyPack;
public class Balance
{
String name;
double bal;
public Balance(String n, double b)
{
name = n;
bal = b;
}
public void show()
{
System.out.println(name + ": $" + bal);
}
}
```

```
import MyPack.*;
class TestBalance
{
public static void main(String args[])
{
Balance test = new Balance("ABCDE", 199.88);
test.show(); // you may also call show()
}
}
```

More information

Case is significant & therefore subdirectory name must match package name exactly. Java supports the concept of package hierarchy. This is done by specifying multiple names in a package statement, separated by dots. **Example: package firstPackage.secondPackage;**

This approach allows us to group related classes into a package & then group related packages into larger package. To store this package in subdirectory named firstPackage/secondPackage.

A java package file can have more than one class definition. in such cases only one of the classes may be declared public & that class name with .java extension is the source file name.When a source file with more than one class definition is complied, java creates independent .class files for these classes.

4) What are benefits of packages(4 Marks)

Ans **Following benefits are achieved by organizing classes into packages:**

1. Packages are use to bundle classes and interfaces.
2. The classes contained in the packages of other programs can be easily reused.
3. Two classes in two different packages can have same name. They may be referred by their fully qualified name, comprising package name & class name.
3. Packages provide way to hide classes thus preventing other programs or packages from accessing classes that are meant for internal use only.
4. Packages also provide a way for separating "design" from "coding". First we can design classes & decide their relationship & then we can implement Java code needed for methods.It is possible to change implementation of any method without affecting rest of the design.

5) Write Syntax of Package?

Ans

Syntax to create package is

```
package packagename;
class <classname>
{
…….
……..
}
```

6) Example for Package?

Ans

```
package package1;
public class Box
{
int l = 5;
int b = 7;
int h = 8;
public void display()
{
System.out.println("Volume is:"+(l*b*h));
}
}
```

source file:

```
import package1.Box;
class volume
{
public static void main(String args[])
{
Box b=new Box();
b.display();
}
}
/*OUTPUT:
C:\java programsl>java volume
```

Volume is:280
*/

Note : Any other example showing use of package can also be considered.

7) Write the effect of access specifiers public, private and protected in package. (4 or 8 Marks)

8) Describe access control parameters with suitable example. (8 Marks)

9) Explain visibility control in java. (8 Marks)

Ans

Visibility control in Java is implemented using Access Modifiers. An Access Modifier is a key word in java that determines what level of access or visibility a particular java variable/method/ constructors or class has. There are 5 basic access modifiers in java. They are:

1. Visible to the package. the **default**. No modifiers are needed.
2. Visible to the class only (**private**).
3. Visible to the world (**public**).
4. Visible to the package and all subclasses (**protected**).
5. Visible to same class,sub class in same package and other package**(private protected)**

Access modifier → / **Access location** ↓	**public**	**protected**	**Friendly (default)**	**Private protected**	**private**
Same class	yes	yes	yes	yes	yes
Subclass in same package	yes	yes	yes	yes	no
Other classes in same package	yes	yes	yes	no	no
Subclass in other packages	yes	yes	no	yes	no
Non-subclasses in other packages	yes	no	no	no	no

1) Default Access Modifier - No keyword:

A variable or method declared without any access control modifier is default.It is available to any other class in the same package. Visible to all class in its package.

e.g

Variables and methods can be declared without any modifiers, as in the following

Examples:

```
String version = "1.5.1";
boolean processOrder()
{
return true;
}
```

2) Private Access Modifier - private:

Methods, Variables and Constructors that are declared private can only be accessed within the declared class itself. Private access modifier is the most restrictive access level. Class and interfaces cannot be private.

Using the private modifier is the main way that an object encapsulates itself and hide data from the outside world.

e.g.

The following class uses private access control:

```
private String format;
private void get() { }
```

c) Public Access Modifier - public:

A class, method, constructor, interface etc declared public can be accessed from any other class. Therefore fields, methods, blocks declared inside a public class can be accessed from any class belonging to the Java Universe.However if the public class we are trying to access is in a different package, then the public class still need to be imported.Because of class inheritance, all public methods and variables of a class are inherited by its subclasses.

e.g The following function uses public access control:

```
public double pi = 3.14;
public static void main(String[] arguments)
{
// ...
}
```

d) Protected Access Modifier - protected:

Variables, methods and constructors which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class.

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected, however methods and fields in a interface cannot be declared protected. Protected access gives the subclass a chance to use the helper method or variable, while preventing a nonrelated class from trying to use it.

e.g

The following parent class uses protected access control, to allow its child class override

```
protected void show( )
{ }
```

e) private protected: Variables, methods which are declared protected in a super class can be accessed only by the subclasses in same package and subclass in other package. It means visible to class and its subclasses.

Example.

```
private protected void show( )
{ }
```

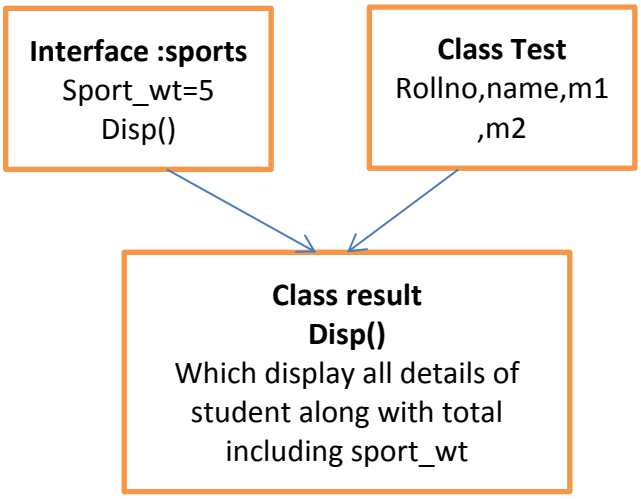
10) Explain with example how to achieve multiple inheritance with interface.(6 Marks)

11) Why interface is needed in java .Give eg explaining its use .(6 Marks)

Ans

Multiple inheritances: It is a type of inheritance where a derived class may have more than one parent class. It is not possible in case of java as you cannot have two classes at the parent level Instead there can be one class and many interface at parent level to achieve multiple

inheritance.Interface is similar to classes but can contain on final variables and abstract method. Interfaces can be implemented to a derived class.



Code :

```
interface sports
{
int sport_wt=5;
public void disp();
}
class test
{
int roll_no;
String name;
int m1,m2;
test(int r, String nm, int m11,int m12)
{
roll_no=r;
name=nm;
m1=m11;
m2=m12;
}
}
class result extends test implements sports
{
result(int r, String nm, int m11,int m12)
{
super(r,nm,m11,m12);
}
public void disp()
{
System.out.println("Roll no : "+roll_no);
System.out.println("Name : "+name);
```

```
System.out.println("sub1 : "+m1);
System.out.println("sub2 : "+m2);
System.out.println("sport_wt : "+sport_wt);
int t=m1+m2+sport_wt;
System.out.println("total : "+t);
}
public static void main(String args[])
{
result r= new result(101,"abc",75,75);
r.disp();
}
}
```

```
D:\>java result
Roll no : 101
Name : abc
sub1 : 75
sub2 : 75
sport_wt : 5

total : 155
```

12) What is interface ? How it is different from class ? With suitable program explain the use of interface.(8 Marks)

Ans

Interface is defined much like class.So interface also contains methods and variables but with major difference the interface defines only abstract method and final fields. This means that interface do not specify any code to implement those methods and data fields contains only constants. Therefore, it is the responsibility of the class that implements an interface to define the code for implementation of these methods. An interface

Java does not support multiple inheritances with only classes. Java provides an alternate approach known as interface to support concept of multiple inheritance. Data fields of interface are static final (as constants) it means that implementing class cannot change it.

Difference between Interface and class

Class	Interface
It can contain abstract and non-abstract methods.non-abstract method will have method definitions.	Methods in interface are abstract.It can contain only method declarations and not method definitions
Multiple inheritance is not possible with classes	Interface was introduced for the concept of multiple inheritance
Classes are extended by another class	Interfaces are implemented by classes
The variables of the class can be final/not	The variables of the interfaces are by default

final/static/non-static	final static
Object of class can be created	Object of interface cannot be created
A class can extend only one class	An class can implement more than one interface
Class contains executable code	Interface contains none executable code
Class has the access specifiers like public, private, and protected.	Interface has only public access specifier
The memory is allocated for the classes.	We are not allocating the memory for the interfaces.
Class contains constructor	An Interface does not contain any constructor.
A class can extend only one class (no multiple inheritance),but it can implement many interfaces.	Interface can extend one or more other interfaces.interface cannot extend a class,or implement a class or interface.
Methods of objects can be called using the object of the class.	Methods of interfaces should be defined in the class which implements the interface.

class Example {
void method1()
{
body
}
void method2()
{
body
}
}
	interface Example{
	int x =5;
	void method1();
	void method2();
	}

interface Area
{
final static float pi = 3.14F;
float compute(float x, float y);
}
class Rectangle implements Area
{
public float compute(float x, float y)
{
return(x * y);
}
}
class Circle implements Area
{
public float compute(float x,float y)
{
return(pi*x * x);
}
}

```
class InterfaceArea
{
public static void main(String args[])
{
Rectangle rect = new Rectangle();
Circle cir = new Circle();
Area area;
area = rect;
System.out.println("Area Of Rectangle = "+ area.compute(5,6));
area = cir;
System.out.println("Area Of Circle = "+ area.compute(10,0));
}
}
```

13) What is interface?Describe syntax,features and need of interface. (8 Marks)
Ans Refer ans 1

Syntax:

```
access interface Interfacename
{
returntype method-name1(parameter list);
returntype method-name2(parameter list);
type final-variable1=value1;
type final-variable2=value2;
:
returntype method-name n(parameter list);
type final-variable n=value n;
}
```

where,
access is either public or not used
interface is the keyword
Interface name is the valid java identifier

Features:

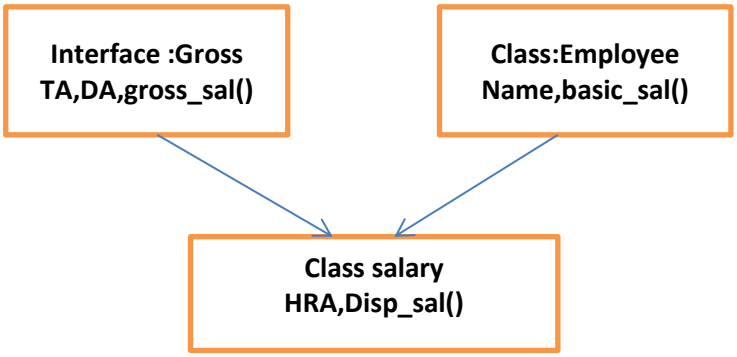
1. Variable of an interface are explicitly declared final and static (as constant) meaning that the implementing the class cannot change them they must be initialize with a constant value. All the variable are implicitly public if the interface, itself, is declared as a public.

2. Method declaration contains only a list of methods without anybody statement and ends with a semicolon.The methods are essentially abstract there can be no default implementation of any method specified within an interface.The class that implements the interface must implement all the methods.

Need:

1. To achieve multiple Inheritance. That is classes in Java cannot have more than one super class. Java allows the use of interface to implement multiple inheritance
2. We can implement more than one Interface in the one class.
3. Methods can be implemented by one or more class.

14) Write a program to make following inheritance.(4 Marks)



Ans

```
interface Gross
{
double ta=500.0;
double da=1200.0;
void gross_sal();
}
class Employee
{
String name;
float basic_sal;
Employee(String n, float b)
{
name=n;
basic_sal=b;
}
void display()
{
System.out.println("Name of Employee="+name);
System.out.println("Basic Salary of Employee="+basic_sal);
}
}

class salary extends Employee implements Gross
{
float hra;
salary(String n, float b, float h)
{
super(n,b);
hra=h;
}
void disp()
{
```

```
display();
System.out.println("HRA of Employee="+hra);
}
public void gross_sal()
{
double gross_sal=basic_sal+ta+hra;
System.out.println("TA of Employee="+ta);
System.out.println("DA of Employee="+da);
System.out.println("Gross Salary of Employee="+gross_sal);
}
}
class Empdetail
{
public static void main(String args[])
{
salary s=new salary("ABC",6000,4000);
s.disp();
s.gross_sal();
}
}
```