

Chapter 6:

JAVA Classes & Libraries

Informatics Practices

Class XII

By- Rajesh Kumar Mishra

PGT (Comp.Sc.)

KV No.1, AFS, Suratgarh

e-mail : rkmalld@gmail.com

Objective

In this presentation, you will learn about the following-

- **Access Specifiers:**

How to control access to class members using private, protected, public and friendly (default) specifiers.

- **Using JAVA Class Libraries:**

How to use Java Class Libraries for various operations on Strings and Mathematical functions.

- **Concept of Packages:**

How to create and use packages to handle a large program.

What is Package ?

A group of classes is called package

- ❑ A large program can be divided into smaller modules and compiled separately. Such modules are named Package.
- ❑ Each package may contain various class definitions under the same package name and stored in a folder.
- ❑ Java offers some ready to use packages like `java.io` package. We can write our own packages also.
- ❑ We can use a package in a program by using `import` statement at top of the program.

```
//to import all members//
```

```
import mypackage.*;
```

```
// to import selected member class//
```

```
import mypackage.myclass;
```

Packages in JAVA

JAVA contains a extensive library of pre-written classes, which can be used in a program. These classes are divided in groups (package). Some packages are-

- ❑ java.applet
- ❑ java.awt
- ❑ java.io
- ❑ java.lang
- ❑ java.net
- ❑ java.util

Note: `java.lang` are imported by default i.e. no import statement is required.

How to create your own package ?

Being a collection of related classes, a package may be user-defined also. We can group related classes and interface in a group (package) by using **package** statement.

```
package myschool;  
class student  
{.....  
.....  
}  
class teacher  
{.....  
.....  
}
```

Once a package has been created, it can be used in other programs by importing it.

```
import myschool.*;
```

What is Derived class (Sub-class) ?

A class can be derived from another class. The derived class is also called a sub-class. A derived class may inherit all the data and method members from its parent. This principle is also known as Inheritance.

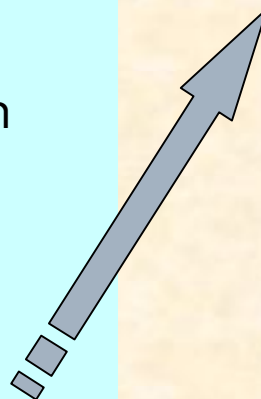
E.g. If **human** class is defined, we can derive **student** and **teacher** class by inheriting all the members of the human class, since teacher and student both are human beings.

//e.g. derivation of sub class//

```
class human
{ String name;
  int age;
  void method1()
  {.....}
}
class student extends human
{int rollno;
  .....
  void method2()
  {.....}
}
```

```
class teacher extends human
{int empno;
  .....
  void method3()
  {.....}
}
```

A sub-class may be derived in the same package or different package also.



Access Specifiers

The Access Specifiers control access to members of class from / within Java Program. Java supports various Access Specifiers to control the accessibility of class members.

❑ Private :

A variable or method declared as private, may not be accessed outside of the class. Only class member can access them, since they are private to others.

❑ Protected:

Protected members can be accessed by the class members and subclasses (derived classes) in current package or in other package.

❑ Public:

Class members declared as public, are accessible to any other class i.e. everywhere , since they are public.

❑ Package (default):

If no any specifier is mentioned, **default** or **friendly** access is assumed. Class member may be accessed by any other Class members available in the same package, but not accessible by the other classes outside the package, even subclasses.

Private Access Specifier

Members declared as private are accessible by the members of the same class, since they are private. A **private** key word is used to specify.

```
//e.g to demonstrate private specifier.//
```

```
class abc
```

```
{ private int p;
```

```
  private void method1()
```

```
  { p=10;
```

```
    system.out.print("I am Private method");
```

```
  }
```

```
}
```

```
class xyz
```

```
{.....
```

```
void method2()
```

```
{ abc x = new abc();
```

```
  x.p =10;
```

```
  x.method1() ;
```

```
}
```

```
}
```

legal use of class members.
(Only class members can
access private members)

Illegal use of members of
class **abc**, since they are
private to **abc**

Protected Access Specifier

Protected members are accessible by all the classes in the same package and sub-classes (same of different packages). A **protected** key word is used to specify.

```
Package mypackage;
class abc
{ protected int p;
  protected void method1()
  { p=10;
    system.out.print("Protected
                      method");
  }
}
class xyz
{.....
 void method2()
 { abc x = new abc();
   x.p =10;
   x.method1() ;
 }
}
```

legal

legal use

```
Lets another Package...
package yourpackage;
import mypackage.*;
class pqr extends abc
{ void method3()
  { abc x=new abc();
    pqr y=new pqr();
    x.p=10;
    x.method1();
    y.p=10;
    y.method1();
  }
}
```

illegal

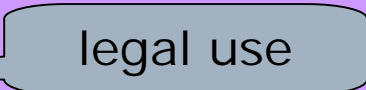
legal i.e. derived class pqr

Public Access Specifier

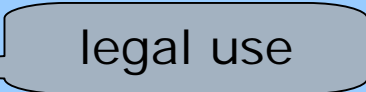
Public Members can be access at anywhere i.e. same or different package.

A **public** key word is used to specify.

```
package mypackage;
class abc
{ public int p;
  public void method1 ()
  { p=10;
    system.out.print("Public method");
  }
}
```



```
package yourpackage;
import mypackage.* ;
class xyz
{.....
  void method2()
  { abc x = new abc();
    x.p =10;
    x.method1 () ;
  }
}
```



Package (friendly) Access Specifier

If no specifier is explicitly specified, Java assumes default (friendly) access i.e. all the members are accessible in all other classes of the same package only, since they are trusted or friends. This is called **Package level** access.

No any key word is used to specify default access.

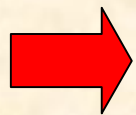
```
package mypackage;
class abc
{ int p;
  void method1 ()
    { p=10;
      system.out.print("Package method");
    }
}
class xyz
{.....
  void method2 ()
    { abc x = new abc();
      x.p =10;
      x.method1 () ;
    }
}
```

legal use

legal use

Access specifier – At a Glance

Access specifier	Classes in the same package	Classes in other packages	Accessibility by the sub-class (same package)	Accessibility by the sub-class (other package)
Public	✓	✓	✓	✓
Protected	✓	x	✓	✓
Private	x	x	x	x
Package	✓	x	✓	x



Access specifiers are applicable to both the Data and Method members either Instance or class member (Static).

JAVA Libraries

- ❑ A library is readymade and reusable component/codes that can be used in a program to perform predefined task.
- ❑ Some commonly used Java libraries are Math Library, String Library, Utility Library and IO Library etc.
- ❑ You can use import statement at the top of the program to include the Java libraries.

```
import java.io.*;
```

- ❑ The `java.lang` is the default imported library in your program without writing import statement.
-

String Library & its commonly used methods

Method Prototype	Description
<code>boolean equals(str)</code>	Compare this (current) string to given string and returns true if both are true otherwise false. e.g. <code>boolean test=str1.equals(str2);</code>
<code>int compareTo(str1,str2)</code>	Compare two strings in alphabetical order.
<code>boolean equalsIgnoreCase(str)</code>	Compare this string to given string but ignores case difference.
<code>int length(str)</code>	Returns the length of this string. e.g. <code>int x=str1.length();</code>
<code>char charAt(num)</code>	Returns the character at given position in this string. e.g. <code>char ch=str1.charAt(3);</code>
<code>String replace(char1,char2)</code>	Returns a new string after replacing all occurrences of char1 by char2.
<code>String substring(num1,num2)</code>	Returns a substring from this string from num1 to num2 position.
<code>String concat(str)</code>	Return a string after appending str into this string. e.g. <code>String name= firstname.concat(lastname);</code>
<code>String toLowerCase(str)</code>	Coverts all the characters of this string into lowercase.
<code>String toUpperCase(str)</code>	Coverts all the characters of this string into Upper case.
<code>Int indexOf (chr)</code>	Returns position of chr into this string. e.g. <code>int x= str1.indexOf('A');</code>

Math Library & its commonly used methods

- ❖ Java provides math library, which available under java.lang package.
- ❖ In order to use functions/methods of math library, you need to invoke function using **math** keywords before the function.

e.g. `x=math.abs(-7.5);`

Method Prototype	Description
<code>pow(num1,num2)</code>	It computes $\text{num1}^{\text{num2}}$, where num1 and num2 are numbers. e.g. <code>system.out.print(""+math.pow(2,3);</code>
<code>round(num1)</code>	It rounds off a given number to its nearest integer. It can take float/double as argument. e.g. <code>system.out.print(""+math.round(1.5));</code> 2 <code>system.out.print(""+math.round(-1.5));</code> -1

Using Dates & Times in JAVA

❖ Java offers two classes in java.util package to manipulate date and time.

1. java.util.Date

2. java.util.Calendar

❖ In order to use Date & calendar, you need to import java.util package. E.g. `import java.util.*;`

<code>Date d=new Date();</code>	It returns system date in the given format. Tue Jul 20 17:30:22 GMT+05:30 2010
<code>Calendar c = Calendar.getInstance();</code>	The following fields can be used to access various date and time values. c.get(Calendar.DATE) c.get(Calendar.MONTH) c.get(Calendar.YEAR) c.get(Calendar.HOUR) c.get(Calendar.MINUTE) c.get(Calendar.SECOND)