Topics

- Class Definition Syntax
- Methods and Attributes Syntax

Defining Classes : Class Syntax

innovate achieve lead

Class Syntax

<scope> [<abstract>/<final>] [<static>] class <class-name> [extends <super-class-name>]
[implements <interface-name-1> , <interface-name-2> , ... , <interface-name-n>]

```
{
        << Instance Fields>>
        << Methods>>
        ..................
        ..................
}
```

Body of class

- […] → Represents Optional Features
- <abstract>, <final>, <static>, <class> , <extends>, <implements> are Java Keywords
- <scope> : public, private, protected, package private
- <abstract> : Used to Define Abstract Classes
- <final> : Optional Field, If used then it Indicates that class can not have subclasses
- <static> : Used only for nested (class defined inside some other class) classes only
- <extends> : extends keyword is used for sub-classes
- <implements> : implements keyword is used when a class implements interfaces

Types of Classes

- Broad Category of Classes

1. Outer Classes
2. Nested Classes
   a. Static Nested Classes
   b. Non static Nested Classes

Outer Classes

```
// File Name: Demo1.java
class A
{
}// End of class A
class B
{
} // End of class B
class C
{
}// End of class C
```

Nested Classes

```
// File Name: Demo2.java
class A
{
        class A1
        {
        }// End of class A1

        static class A2
        {
        }// End of class A2
}// End of class A
```

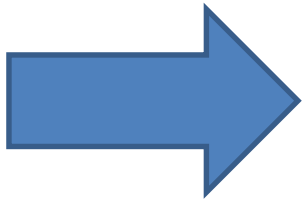
Non-Static Nested Class

Static Nested Class

Class Definition Rules : Rule 1

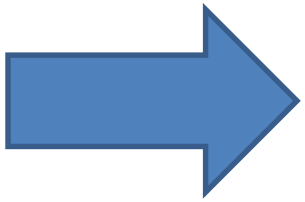
- **Rule 1: Scope of the Outer Class can be either public or package private**

```
// File Name Demo.java
private class A
{
}// End of class A
protected class B
{
}// End of class B
```



```
Demo.java:1: modifier private
not allowed here
private class A
      ^
Demo.java:4: modifier
protected not allowed here
protected class B
      ^
2 errors
```

```
// File Name Demo.java
public class Demo
{
}// End of class Demo
class B
{
}// End of class B
```



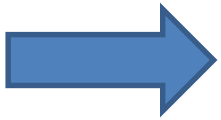
```
<< No Error>>
Compilation Successful
```

Class Definition Rules : Rule 2

innovate achieve lead

- **Rule 2: In a single source '.java' file, only one class can be defined with <public> scope.**

```
// File Name:Demo.java
public class A
{
}// End of class Demo
public class B
{
}// End of class B
public class C
{
}// End of class B
```

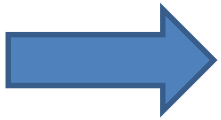


```
Demo.java:2: class A is public, should be
declared in a file named A.java
public class A
       ^
Demo.java:5: class B is public, should be
declared in a file named B.java
public class B
       ^
Demo.java:8: class C is public, should be
declared in a file named C.java
public class C
       ^
3 errors
```

Class Definition Rules : Rule 3

- **Rule 3: If a source '.java' has a class with <public> scope then file name should be named on class name**

```
// File Name:Demo.java
public class A
{
}// End of class Demo
class B
{
}// End of class B
class C
{
}// End of class B
```



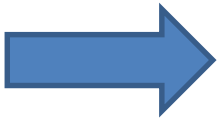
```
javac Demo.java

Demo.java:
class A is public, should be declared in a file
named A.java
public class A
       ^
1 error
```

So, In order to successfully compile, the file should be named A.java

Class Definition Rules : Rule 4

- **Rule 4 : <static> keyword can only be used for nested classes and not for outer classes**

```
// File Name: Demo.java
static class X
{
}// End of class X
static class Y
{
}// End of class Y
```

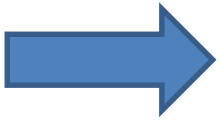


```
javac Demo.java
modifier static not allowed here
static class X
      ^
modifier static not allowed here
static class Y
      ^
2 errors
```

Class Definition Rules : Rule 5

- **Rule 5 : <final> class can not have sub-classes. However, <final> keyword can be used for both Outer and Nested Classes**

```
// File Name: Demo.java
final class X
{
}// End of class X
class Y extends X
{
        final class Y1
        {
        }// End of class Y1
}// End of class Y
```

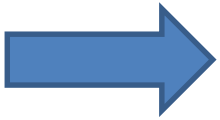


```
javac Demo.java
Demo.java:5: cannot inherit from
final X
class Y extends X
              ^
1 error
```

Class Definition Rules : Rule 6

- **Rule 6 : <final> and <abstract> keywords can not be used together for a class**

```
// File Name: Demo.java
final abstract class X
{
}// End of class X
```

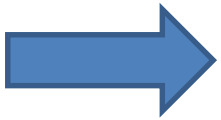


```
F:\>javac Demo.java
Demo.java:2: illegal combination of
modifiers: abstract and final
final abstract class X
                ^
1 error
```

Class Definition Rules : Rule 7

- **Rule 7 : <extends> keyword can be used only to extend one super class. [Because Java does not support multiple inheritance directly]**

```
// File Name:Demo.java
class X
{
}// End of class X
class Y extends X
{
}// End of class Y
class Z
{
}// End of class Z
class A extends X , A
{
}// End of class A
class B extends Y extends Z
{
}// End of class B
```



```
F:\>javac Demo.java
Demo.java:11: '{' expected
class A extends X , A
                  ^
1 error
```

```
F:\>javac Demo.java
Demo.java:11: '{' expected
class A extends X , A
                  ^
Demo.java:14: '{' expected
class B extends Y extends Z
                  ^
2 errors
```

Instance Field and Method Syntax

Instance Field Definition Syntax

<scope> [<static>] [<final>] <type> <variable-name> [= <value>];

- [..] are optional features
- Where <scope> can be : public, protected, private or package private
- <type> can be : primitive type, class type or an interface type

Partial Method Definition Syntax

```
<scope> [<static>] [<final>] [<abstract>] [<synchronized>] <return-type> <method-name>(<arguments>)
{

        ……..// Method Body

} // End of Method
```

Class Definition Example


```
// File Name : Complex Number
class ComplexNumber
{
    private          double    real;      // Real Part
    private          double    imag;      // Imaginary Part

    /* Method to set the Value of Real Part */
    public           void      setReal(double realValue)
    {
            real = realValue;
    }// End of Method

    /* Method to set the Value of Imaginary Part */
    public           void      setImag(double imagValue)
    {
            imag = imagValue;
    }// End of Method
}// End of class ComplexNumber
```

Thank You