

Objectives

The structures and components of a Java class

Understanding executable Java applications

Understanding Java packages

Importing Java packages into your code

Applying access and non access modifiers

Features and components of Java

Structure of a Java class

Package statement ----- 1

Import statements ----- 2

Comments ----- 3a

Class declaration ----- 4

Variables ----- 5

Comments ----- 3b

Constructors ----- 6

Methods ----- 7

Nested classes, nested interfaces and Enum are not covered Enum

Packages

Packages (cont)

Per java naming conventions, packages names should all be in lowercase

The package and subpackage names are separated using a dot (.)

Package names follow the rules defined for valid identifiers in Java

For classes and interfaces defined in a package, the *package* statement is the first statement in a Java source file (a .java file)

There can be a maximum of one *package* statement per Java source code file (.java file)

All the classes and interfaces defined in a Java source code file are defined in the same package. They can be defined in separate packages.

The hierarchy of classes and interfaces defined in packages must match the hierarchy of the directories in which these classes and interfaces are defined in the code.

To enable the Java Runtime Environment (JRE) to find your classes, add the base directory that contains your packaged Java code to the classpath

Comments

Can appear before and after a *package*

Can appear before and after a *class* definition

Can appear before and after a *method*

Multiline `/* */`
comments

Multiline `/*`
comments `*/` Can contain special characters. The following is a coding practice (but not required):
`/*`
`* comments that span`
`* multiple lines`
`*`
`*/`

End-of-`//`
line
comments

Comments (cont)

Is it `String name = "\/* Juan`
valid? `*/ Paul";`

Javadoc comments are special comments that start with `/**` and end with `*/` (this is processed by Javadoc, a JDK tool to generate API documentation)

Class declaration

Access `public/private/protected`
modifiers

Nonaccess `static/final/abstract/synch-`
modifiers `ronized`

Class name

Name of the `if the class is extending`
base class `another class`

Class body `(class fields, methods,`
`constructors), included {}`

More on classes

`class` used to specify the attributes
definition `(variables)` and behavior
`(methods)` of an object.

A class name starts with the keyword `class`. It is cAsE-sEnS-iTiVe

The state of a class is defined using *attributes* or instance variables

It isn't compulsory to define all attributes of a class before defining its methods. But this is far from being optimal for readability

Methods often used to manipulate the instance variables

A *class method* or *static method* can be used to manipulate the *static* variables

All Java *classes* are part of a *package*

If the class has not package definition, it is classified in the **default package** (which doesn't have a name)

Must be the first in the class definition (though you can define comments above its declaration)

The *package* statement can't appear within a class declaration or after the class declaration

Must appear exactly once in a class

Classes and interfaces in the same package can use each other without prefixing their names with the package name

The use a class or an interface of another package, you must use its fully qualified name	<code>packag - eName.a - ny Sub - packag - eName.C - lassName</code>
---	--

Use **import** statement to use the simple name of a class or interface

A package is made of multiple sections that go from the more-generic(left) to the more specific(right)



By **mjorod**
cheatography.com/mjorod/

Not published yet.
Last updated 18th December, 2017.
Page 1 of 3.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

More on classes (cont)

Instance variables/- attributes Each object has its own copy of the instance variables

The instance variables are defined within a class but outside all methods in a class

Interfaces in a Java source code file

Interface in a Java source code file Types??

Specifies a contract for the classes to implement

Grouping of related methods and constants

Starting Java 8, methods in an interface can define a default implementation

Interface can also define *static* methods.

You can define either a single class or an interface in a Java source code file or multiple such entities.

-able- suffix for interfaces like Comparable? prefix for interfaces, like IComparable?

The classes and interfaces can be defined in any order of occurrence in a Java source code file

Interfaces in a Java source code file (cont)

Classes and interfaces defined in the same Java source code file can't be defined in separate packages.

```
interface Controls {
    void changeChannel(int channelNumber);
    void increaseVolume();
    void decreaseVolume();
}
```

Executable Java Applications

What is a Java class? An executable Java class, when handed over to the JVM, starts its execution at a particular point in the class-- *main* method. The JVM starts executing the code that's defined in the *main* method.

A Java application can define more than one executable class. We have to choose one when the times comes to start its execution by the JVM

main method Must be marked **public**

Must be marked as a **static** method

The name of the method must be **main**

The return type of this method must be **void**

The method must accept a method argument of a **String** array or a variable argument (varargs) of type **String**

Executable Java Applications (cont)

```
public static void main(S tring - ng... args)
```

It's valid

```
public static void main(S tring args...)
```

Won't compile

```
public static void main(S - tring[] hola)
```

acceptable

```
public static void main(S tring argume nmt - os[])
```

It's ok

```
static public void main(S tring argume nmt - os[])
```

Yes, you can exchange modifiers (ok)

set up to compile or execute from command prompr <http://docs.oracle.com/javase/tutorial/getStarted/cupojava/index.html>

D:\>java hello Execute a Java app

D:\>java hello 1 2 D:\>java hello with arguments

Java doesn't pass the name of the class as an argument to the main method (like C does)



By mjorod
cheatography.com/mjorod/

Not published yet.
Last updated 18th December, 2017.
Page 2 of 3.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>