

What is Java? (Popular Perception)

Monday, October 19, 2020 11:36 AM




Popular Perception/Definition

- Java is an object oriented programming language
- Class based OO language
- High level programming language
- Multi-threaded
- Write once - run anywhere
- Cross platform language
- Platform Independent
- Robust and Secured
- Interpreted language
- It is an open source

Is this claim valid?

Contradictions/Challenges

- Does Android Phone support Java?
 - NO
 - A Java class doesn't run on Android Phones out of box
 - Although majority android application is written using java language.
- Does iOS support Java?
 - NO
- Have you heard the term — "This computer doesn't have Java?"
 - We havent installed Java.
 - What have we not installed — Java Programming lanuage?
 - Do everyone needs to program in Java?
- When you try to download "java", what are your actually downloading?

Windows - Which should I choose?		
	Windows Online filesize: 1.99 MB	Instructions
	Windows Offline filesize: 69.61 MB	Instructions
	Windows Offline (64-bit) filesize: 79.19 MB	Instructions
If you use 32-bit and 64-bit browsers interchangeably, you will need to install both 32-bit and 64-bit Java in order to have the Java plug-in for both browsers. » FAQ about 64-bit Java for Windows		

Mac OS X - Mac FAQ		
--------------------	--	--

- Why request for a Java download ended up downloading JRE?
- Is JRE and object oriented programming language?
- Is JRE platform independent
- Can you write an object oriented programming language using JRE?

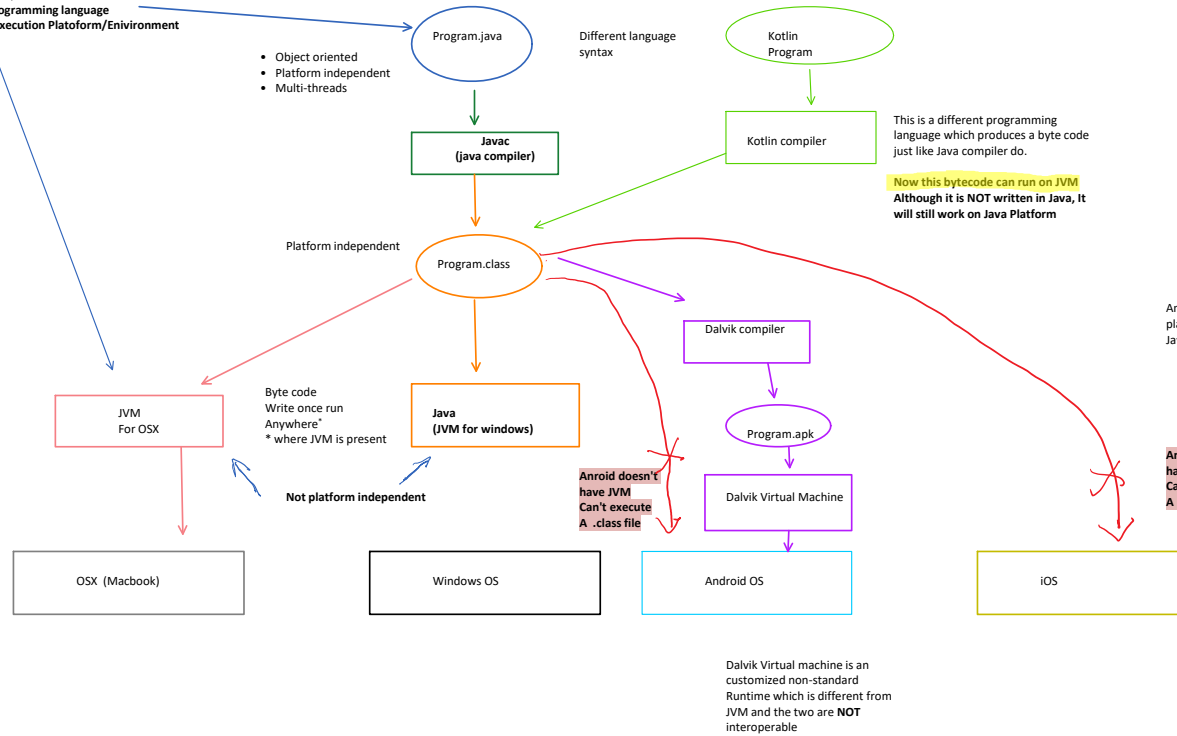


How many Java are there?

Monday, October 19, 2020 12:08 PM

In the world of computation Java refers to two related but **distinct terms**

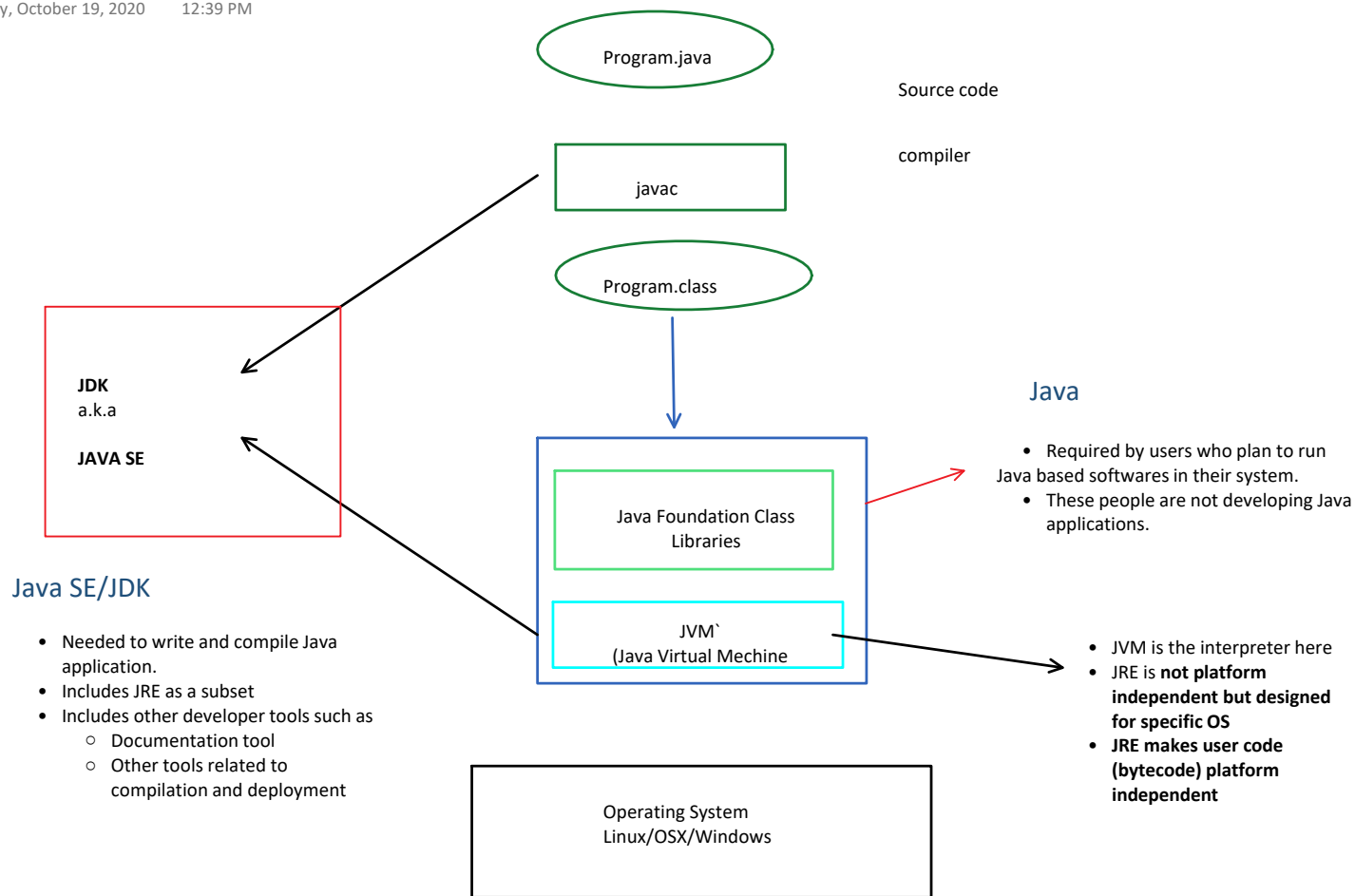
1. **Java is a Programming language**
2. **Java is an execution Platform/Environment**



Since the release of Android Studio 3.0 in October 2017, Kotlin is included as an alternative to the standard Java compiler. The Android Kotlin compiler lets the user choose between targeting Java 6 or Java 8 compatible bytecode. Kotlin has been Google's preferred language for Android app development since 7 May 2019.

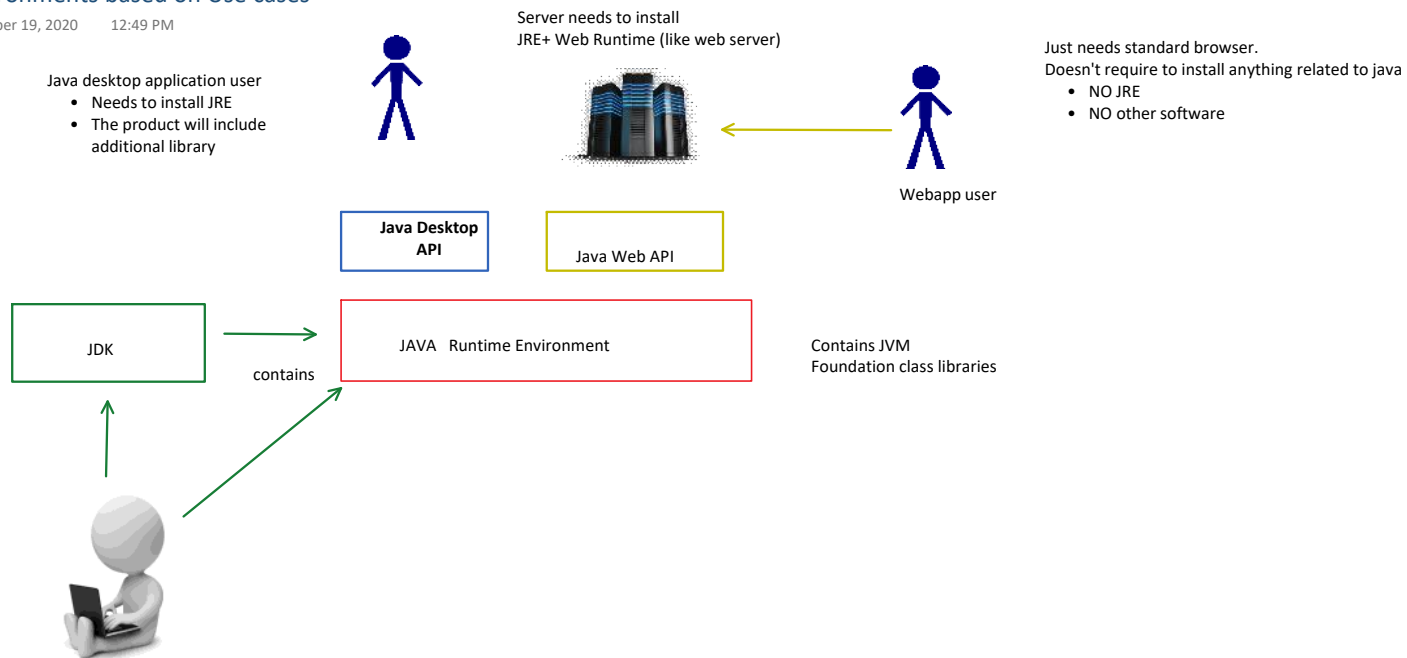
Java Framework

Monday, October 19, 2020 12:39 PM



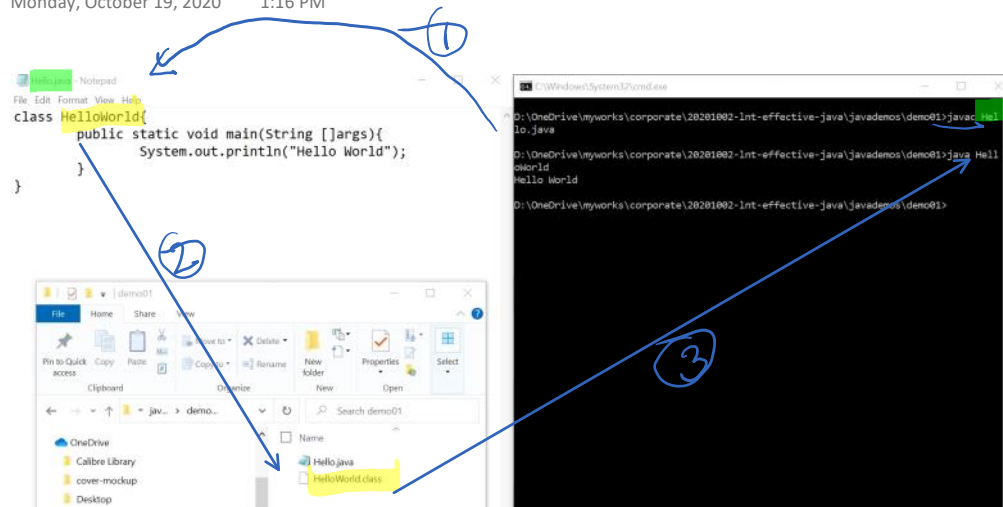
Java Environments based on Use cases

Monday, October 19, 2020 12:49 PM



Compiling and Running Java code

Monday, October 19, 2020 1:16 PM

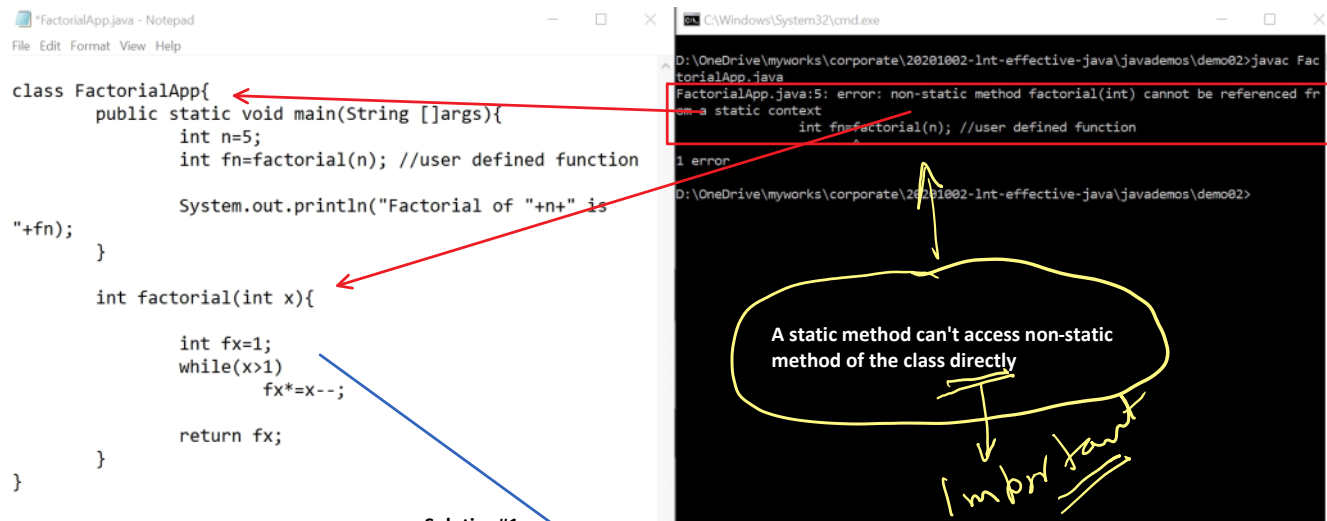


1. You compile a source file
- c:> **javac HelloWorld.java**
1. The byte code name is same as that of Class present in source code and not same as the .java file name
2. You run the byte code using java command

c:> **java HelloWorld**

Static context

Monday, October 19, 2020 1:33 PM



Solution#1

Solution

- There are multiple way to solve this problem
- We will choose different solutions depending on the context

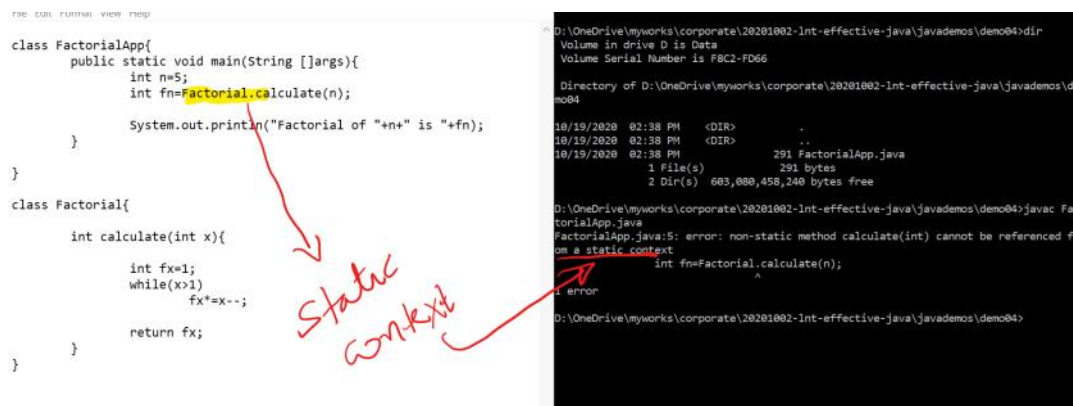
Solution 1

- Make factorial function static

```
static int factorial(int x){
    int fx=1;
    while(x>1)
        fx*=x--;

    return fx;
}
```

More on static context



A Class name is a static context. You can call only static methods using Class reference and not non-static methods

Working with a non-static context

To work with a non-static method, we need an object of the class to

Working with a non-static context

To work with a non-static method, we need an object of the class to refer and use the method.

The image shows a Java IDE with two windows. The left window, titled 'FactorialApp.java - Notepad', contains the following code:

```
class FactorialApp{
    public static void main(String []args){
        int n=5;

        //step#1: get object of the class
        Factorial fact=new Factorial();

        //step#2: call method using the object
        int fn=fact.calculate(n);

        System.out.println("Factorial of "+n+" is "+fn);
    }
}

class Factorial{
    int calculate(int x){
        int fx=1;
        while(x>1)
            fx*=x--;
        return fx;
    }
}
```

Handwritten orange annotations are present:

- An arrow points from the text 'Non Static' to the `calculate` method in the `Factorial` class.
- An arrow points from the text 'Instance of class level Method' to the `fact.calculate(n)` call in the `main` method.

The right window is a terminal titled 'C:\Windows\System32\cmd.exe' showing the execution of the program:

```
D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04>dir
Volume in drive D is Data
Volume Serial Number is F8C2-FD66

Directory of D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04

10/19/2020  02:38 PM  <DIR>          .
10/19/2020  02:38 PM  <DIR>          ..
10/19/2020  02:44 PM                370 FactorialApp.java
               1 File(s)                370 bytes
               2 Dir(s)  603,080,458,240 bytes free

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04>javac FactorialApp.java

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04>java FactorialApp
Factorial of 5 is 120

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04>dir
Volume in drive D is Data
Volume Serial Number is F8C2-FD66

Directory of D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04

10/19/2020  02:45 PM  <DIR>          .
10/19/2020  02:45 PM  <DIR>          ..
10/19/2020  02:45 PM                312 Factorial.class
10/19/2020  02:45 PM                943 FactorialApp.class
10/19/2020  02:45 PM                399 FactorialApp.java
               3 File(s)                1,654 bytes
               2 Dir(s)  603,080,450,048 bytes free

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04>
```

Multi Class Java Program

Monday, October 19, 2020 2:36 PM

The image shows a Notepad window on the left containing the source code for a multi-class Java program. The code defines two classes: `FactorialApp` and `Factorial`. `FactorialApp` has a `main` method that calls `Factorial.calculate`. `Factorial` has a static `calculate` method that computes the factorial of a number using a while loop. Blue arrows point from the `Factorial` class definition in the code to the `Factorial.class` file in the command prompt output. A yellow text box with a black border is positioned below the code, containing the text: "When compiled a separate .class is generated for every class that exists in our system". On the right, a Windows Command Prompt window shows the execution of `javac` and `java` commands. The `javac` command successfully compiles both `FactorialApp.java` and `Factorial.class`. The subsequent `java` command runs the program, outputting "Factorial of 5 is 120".

```
class FactorialApp{
    public static void main(String []args){
        int n=5;
        int fn=Factorial.calculate(n);

        System.out.println("Factorial of "+n+" is "+fn);
    }
}

class Factorial{
    static int calculate(int x){
        int fx=1;
        while(x>1)
            fx*=x--;

        return fx;
    }
}
```

When compiled a separate .class is generated for every class that exists in our system

```
D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo03>dir
Volume in drive D is Data
Volume Serial Number is F8C2-FD66

Directory of D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo03
10/19/2020  02:34 PM  <DIR>          .
10/19/2020  02:34 PM  <DIR>          ..
10/19/2020  02:34 PM                291 FactorialApp.java
               1 File(s)                291 bytes
               2 Dir(s)  603,080,462,336 bytes free

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo03>javac FactorialApp.java

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo03>java FactorialApp
Factorial of 5 is 120

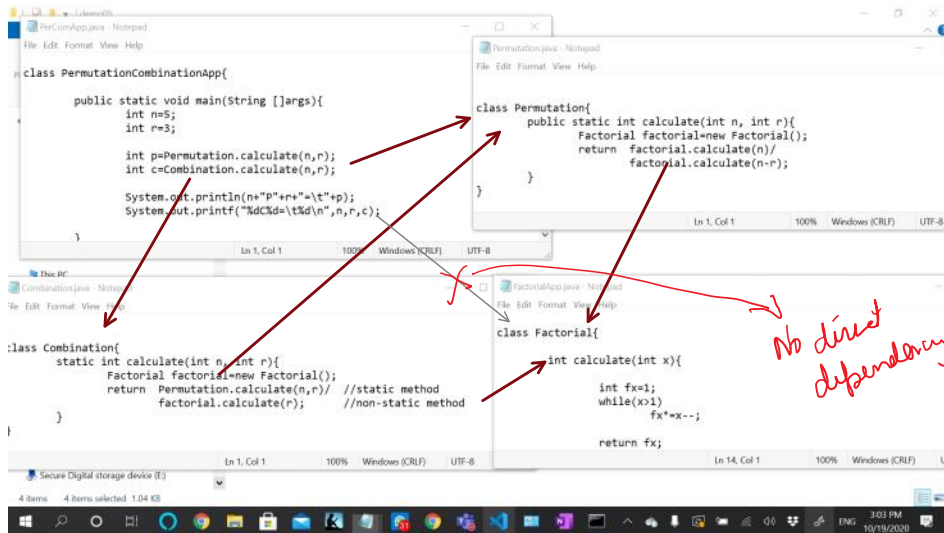
D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo03>dir
Volume in drive D is Data
Volume Serial Number is F8C2-FD66

Directory of D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo03
10/19/2020  02:35 PM  <DIR>          .
10/19/2020  02:35 PM  <DIR>          ..
10/19/2020  02:35 PM                312 Factorial.class
10/19/2020  02:35 PM                925 FactorialApp.class
10/19/2020  02:34 PM                291 FactorialApp.java
               3 File(s)                1,528 bytes
               2 Dir(s)  603,080,458,240 bytes free

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo03>
```


Class Dependencies

Monday, October 19, 2020 3:03 PM



- Unlike c/c++ (or even javascript or python) You don't need any kind of include of classes before you can use.
- An .class present in the current folder can be accessed directly.

How to compile a project with multiple files

Method #1 compile all files using * wildcard

```
c:> javac *.java
```

Method #2 compile the startup file — the one that contains main()

The steps would be as follows

1. It will try to compile PerComApp.java to create class PermutationCombinationApp.class.
 - While compiling it needs to use Permutation class
2. Javac searches for a file Permutation.class
 - It is not currently present
3. Javac searches for file Permutation.java.
 - It compiles Permutation.java to create Permutation.class
4. While compiling Permutation.class, it realizes it Needs Factorial.class
 - There is no Factorial.class present
5. Since there is not Factorial.class Present, it searches for Factorial.java
 - Factorial.java is also not present
 - Factorial class is a part of FactorialApp.java file. **It is not housed in a file name of its own**
 - **At this stage it returns an error message**

if I have both Combination.class and Combination.java present, which of them will be used by javac?

- If both files are present, then javac would compare their last modification data.
- If .class file is more recent than .java file, that means there has been no change in source code since last compilation, it would simply use the class file.
- If the class file is modified after last compilation, that means we must rebuild class file
 - It would recompile the java file

Recommendation!

- We should create one class per java file
- A class should be housed in a java file of same name
- This will help Java compiler automatically compile your java file if required.

Compilation based on Modification

Name	Status	Date modified	Type	Size
<input type="checkbox"/> Combination.class	✓	10/19/2020 3:27 PM	CLASS File	1 KB
<input type="checkbox"/> PermutationCombinationApp.class	✓	10/19/2020 3:27 PM	CLASS File	2 KB
<input checked="" type="checkbox"/> Combination.java	✓	10/19/2020 3:26 PM	JAVA File	1 KB
<input type="checkbox"/> Factorial.class	✓	10/19/2020 3:20 PM	CLASS File	1 KB
<input type="checkbox"/> Permutation.class	✓	10/19/2020 3:20 PM	CLASS File	1 KB
<input checked="" type="checkbox"/> Permutation.java	✓	10/19/2020 3:19 PM	JAVA File	1 KB
<input checked="" type="checkbox"/> FactorialApp.java	✓	10/19/2020 3:18 PM	JAVA File	1 KB
<input checked="" type="checkbox"/> Factorial.java	✓	10/19/2020 3:17 PM	JAVA File	1 KB
<input checked="" type="checkbox"/> PerComApp.java	✓	10/19/2020 2:56 PM	JAVA File	1 KB

Combination has changed after its last compile
So it is recompiled along with PerComApp.java

Factorial class has not changed after its
Last modification so it is not recompiled

```
C:\Windows\System32\cmd.exe
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo05>javac PerComApp.java
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo05>
```

①
This file will always be compiled irrespective of its date
as we are explicitly compiling

Organization Project files in multiple folders

Monday, October 19, 2020 3:32 PM



```
CAWindows\System32\cmd.exe
Folder PATH listing for volume Data
Volume serial number is F8C2-FD66
D:.
  FactorialApp.java
  PerComApp.java
  lib
    console
      ConsoleWriter.java
    maths
      Combination.java
      Factorial.java
      Permutation.java

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo06>javac PerComApp.java

PerComApp.java:8: error: cannot find symbol
    int p=Permutation.calculate(n,r);
            ^
symbol:   variable Permutation
location: class PermutationCombinationApp
PerComApp.java:9: error: cannot find symbol
    int c=Combination.calculate(n,r);
            ^
symbol:   variable Combination
location: class PermutationCombinationApp
PerComApp.java:11: error: cannot find symbol
    ConsoleWriter writer=new ConsoleWriter();
                        ^
symbol:   class ConsoleWriter
location: class PermutationCombinationApp
PerComApp.java:11: error: cannot find symbol
    ConsoleWriter writer=new ConsoleWriter();
                        ^
symbol:   class ConsoleWriter
```

Java compiler by default doesn't know where to search for the java/class files if it is not present in current folder.

It doesn't search entire file system for those files

How do I locate .java/.class files

1. CLASSPATH

- We can specify an environment variable called **CLASSPATH** listing all the folders where javac/java should search for .java/.class files
- Folders should be separated using path separator character that varies from one os to another
 - Windows using semicolon
 - Linux/osx uses colon

Our class path should look like

```
c:>set classpath=.\lib\console;.\lib\maths;.
```

Current directory

Note

- We have mentioned 3 path here
 - .\lib\console
 - .\lib\maths
 - . (current directory)
- We can't specify just lib
 - We must include right sub directory
- If you are having a classpath, then java/javac doesn't search current directory by default.
 - You must include current directory if you have files in current directory

```
C:\Windows\System32\cmd.exe
Volume serial number is F8C2-FD66
D:..
  FactorialApp.class
  FactorialApp.java
  PerComApp.java
  PermutationCombinationApp.class
lib
  console
    ConsoleWriter.class
    ConsoleWriter.java
  maths
    Combination.class
    Combination.java
    Factorial.class
    Factorial.java
    Permutation.class
    Permutation.java

D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo06>java PermutationCombinationApp
Error: Could not find or load main class PermutationCombinationApp
Caused by: java.lang.ClassNotFoundException: PermutationCombinationApp

D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo06>
```

Java is not searching for .class file in the current folder because current folder is not present in class path

Works correctly with the Right Path Set

```
C:\Windows\System32\cmd.exe
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo06>set classpath=.\lib\console;.\lib\maths;.
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo06>tree /f
Folder PATH listing for volume Data
Volume serial number is F8C2-FD66
D:..
  FactorialApp.class
  FactorialApp.java
  PerComApp.java
  PermutationCombinationApp.class
lib
  console
    ConsoleWriter.class
    ConsoleWriter.java
  maths
    Combination.class
    Combination.java
    Factorial.class
    Factorial.java
    Permutation.class
    Permutation.java

D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo06>java PermutationCombinationApp
SP3= 60
SC3= 10

D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo06>
```

Works fine with the right classpath

Note About class path

- Any classpath set at the terminal or the command window is good for current session only and is lost once you close the window.
- Classpath or any environment variable set on a terminal/command window is not available to other terminal or the command window.
- If you need a classpath everyday then you must store it in **system environment variables**
 - A good place to store classpath for common libraries.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo06>java PermutationCombinationApp
5P3=    60
5C3=    10

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo06>cd \

D:\>java PermutationCombinationApp
5P3=    60
5C3=    10

D:\>
```

Once a classpath is properly set, you
can run your application from
anywhere in your file system.

Assignment01

Monday, October 19, 2020 4:01 PM

Create A Project called Furniture App with following file structure

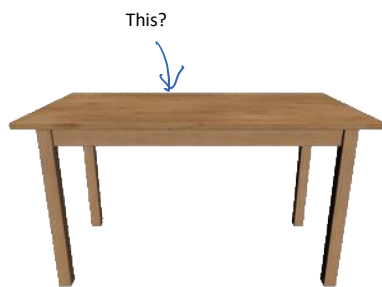
```
<project root>
|
|--> FurnitureApp.java
|
|--> [lib]
|   |
|   |--> [furnitures]
|       |
|       |--> Chair.java
|       |--> Bed.java
|
|--> [data]
|   |
|   |--> List.java
```

FurnitureApp should

1. Craete a list of Furnitures
2. Add Chair and Bed to this list

Write necessary code to compile and run the program

What is a Table?



That?

Product	Rate	Stock
Chair	1000	12
Table	5000	7
Bed	20000	4

- In real world a word can have different meanings. Same word (like Table) can represent multiple different and unrelated elements.
- Often in a single project we may need to use one or more such objects
 - Example:
 - A Furniture Shop sells **Table (Furnitures)**
 - The maintain their stock details in a **Table (Data)**

How do I represent Multiple Objects with same name in same application

Why Multiple Folder Design Doesn't work?

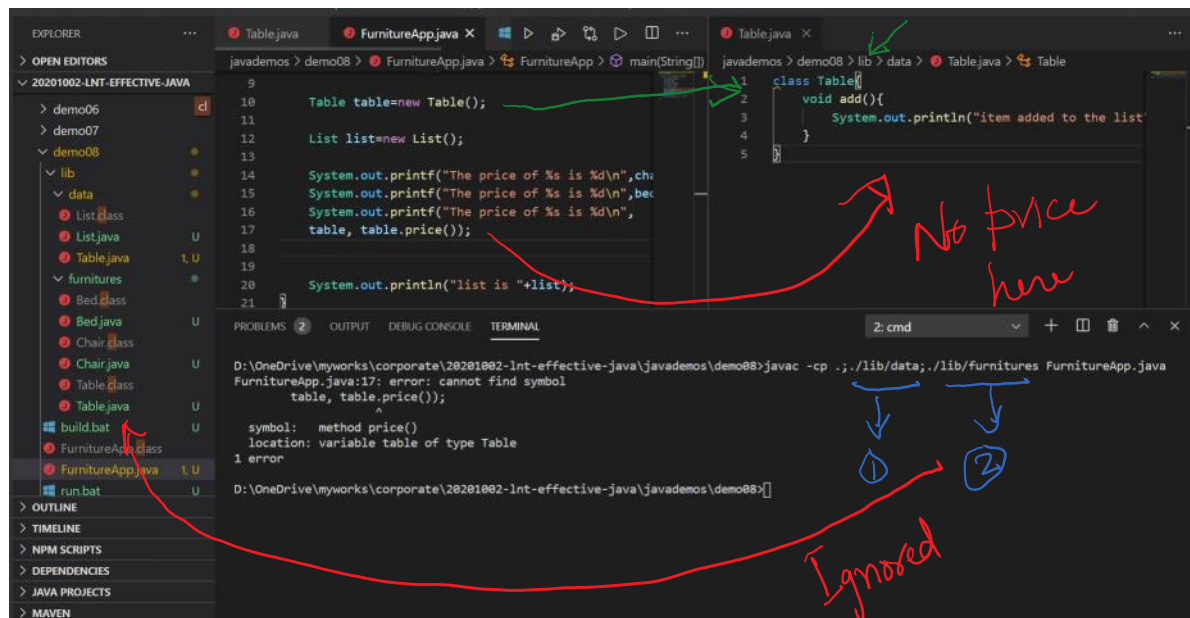
Java/Javac searches for java classes in **sequential order** in every folder mentioned in the **classpath**

- They stop searching when they find their **first match**
- If a class with same name is present in the two folders, it will always match the first folder mentioned in the list and will never use other option.

unreachable

Search order

If we change the order in classpath, it will get a different Table. But we can't get Both Tables to work in single Application



How to make it work?

- Create classes with different Names —
 - FurntiureTable
 - DataTable
- This can avoid name conflicts
- **Why this is not a great idea**
 - We may not always be in a position to find a good prefix
 - Sometimes conflict may be between
 - Your Data Table
 - My Data Table
 - Two different developers may be developing a class for same Purpose
 - Prefixing won't be useful here!

Java Package

Tuesday, October 20, 2020 10:21 AM

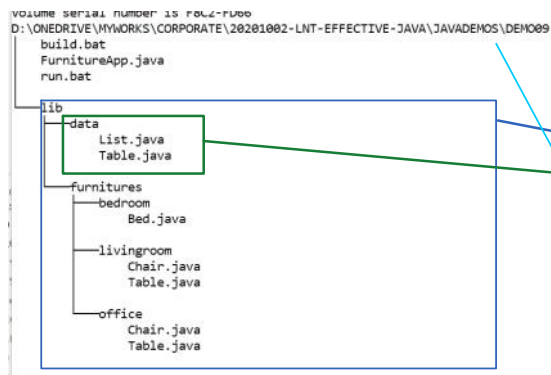
- A java package is a logical grouping for your Java classes and packages
- A Package can contain
 - Java files
 - Class files
 - Other Resource (configurations)
 - Sub Packages
- **A package is physically mapped to a folder on the disk.**
- For every package you will have folder

Note!

- A class which is not part of any specific package is still a part of a global package

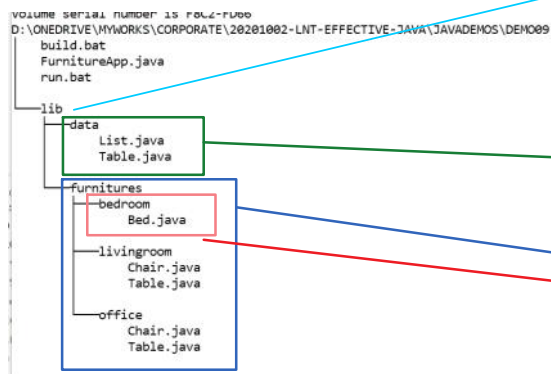
Package is not a folder

- Package resides in a folder.
 - If we have a package **xyz** it would be mapped to a folder **xyz**
- **The key difference between a package and a folder**
 - Folder is an OS concept and Java doesn't know about the folder
 - i. You reach the folders using OS level environment variable like **classpath**
 - ii. Java program internally knows nothing about a folder
- Package is a java concept mapped to folders.
 - **It is a java programming element and using in Java Program**
- **We can designate our selected folder as package**
 - **That would be our choice**



Package	Sub Package	Sub Package	Sub Package	Class	Java Name	
lib					lib	
	data				lib.data	
				List	lib.data.List	
				Table	lib.data.Table	
	furniture				lib.furntiures	
		bedroom			lib.furnitures.bedroom	
				Bed	lib.furnitures.bedroom.Bed	

Alternative View



Package	Sub Package	Sub Package	Class	Java Name	
data				data	
			List	data.List	
			Table	data.Table	
furniture				furntiures	
	bedroom			furnitures.bedroom	
			Bed	furnitures.bedroom.Bed	

- This is just a folder and not a package.
- **This is where the you package lives.**
- **This folder should be present in the classpath**
- Your package folders will not be present in classpath

How do you mark your package?

- We mark our package and sub package by giving **package** statement on the top of class

- **package** statement if present **must be the first statement in a file.**
- **There can be only one package statement per file**
- Package must include entire package sub package hierarchy
- If not package is specified it is assumed to be part of a global un-named package
- .class file must be present in folder mentioned as per package hierarchy
- **Root of the package** should be present in **CLASSPATH**

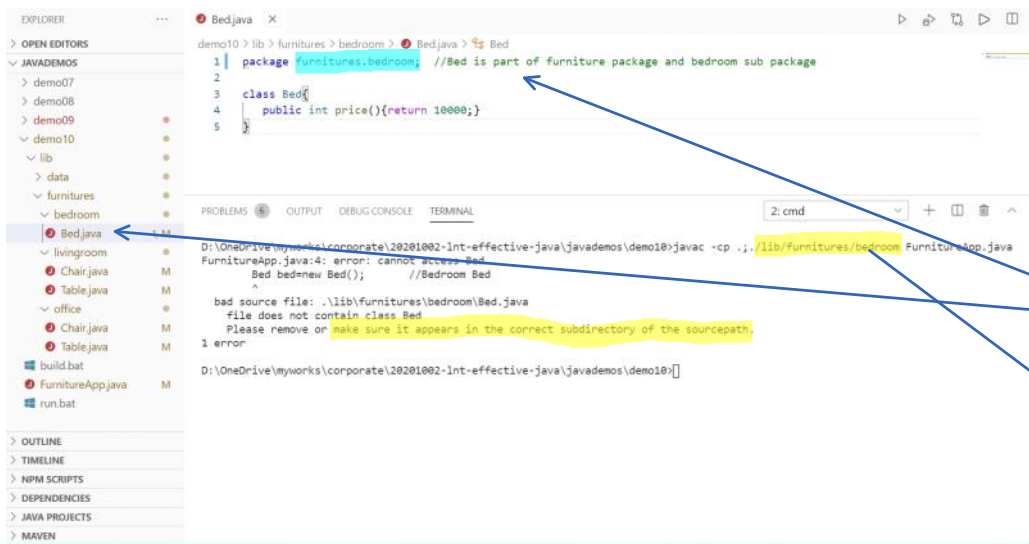
```

demo10 > lib > data > List.java > {} data
1 package data; //we decided that my package starts
2
3 class List{
4     void add(){
5         System.out.println("item added to the list"
6     }
7 }

demo10 > lib > furnitures > bedroom > Bed.java > Bed
1 package furnitures.bedroom; //Bed is part of fur
2
3 class Bed{
4     public int price(){return 10000;}
5 }
  
```

Using Package

Tuesday, October 20, 2020 11:35 AM



- Compiler is expecting **Bed** to be present in a folder

- **furnitures/bedroom**

- Is it not already present in the right folder?

- No.
- Because we are searching for this folder by going inside this folder.
 - Remember this path is part of **classpath**
 - Javac goes into the folder and searches those folder inside

IMPORTANT!

- The error is because package name is mentioned in **CLASSPATH**
- You should never mention package itself in classpath
- You should mention the Parent folder for package in class path

Referring a class defined inside the package

- Once you have created a class **Bed** inside a package **furnitures.bedroom**,
 - you can't access the class simply as **Bed**
 - There is not Bed present in global package
 - You have to access the class Bed using its package qualified name that is **furnitures.bedroom.Bed**



- Can't access Bed without package qualification
- Here is the right way to use it

C

Package and Scopes

Tuesday, October 20, 2020 11:56 AM

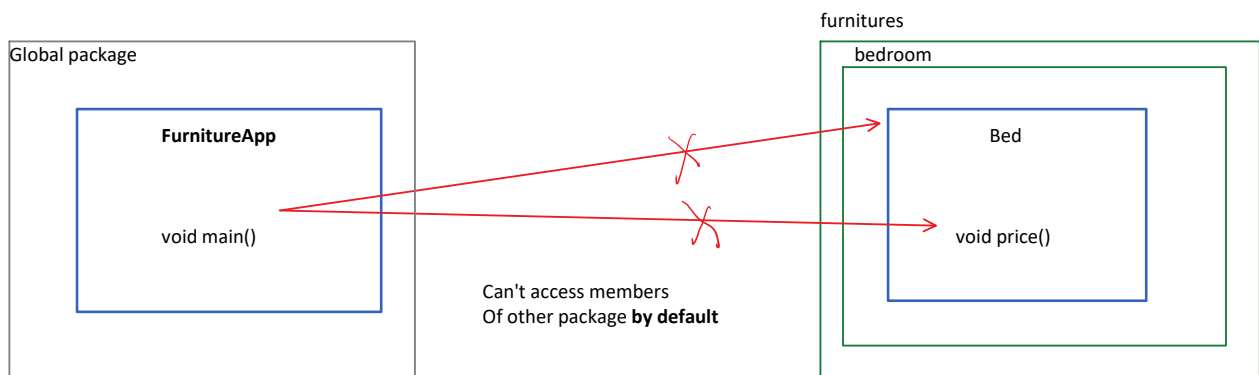
- By default, all elements inside a package (class, class fields, class methods) have a **package scope**.
- They are accessible by members of the same package but not outside the package
- When we write program without package, all classes are part of same global package and can access each other and their method without any problem

```
demo09 > FurnitureApp.java > FurnitureApp > main(String[])
4 public static void main(String []args){
5
6     Chair chair=new Chair(); //Living room chair
7     Bed bed=new Bed(); //Bedroom Bed
8
9
10    Table table=new Table(); //living room table
11
12    List list=new List();
13
14    System.out.printf("The price of %s is %d\n",chair, chair.price());
15    System.out.printf("The price of %s is %d\n",bed, bed.price());
16    System.out.printf("The price of %s is %d\n",table, table.price());
17
18
19
20    System.out.println("list is "+list);
21
22 }
```

```
demo09 > lib > furnitures > bedroom > Bed.java > ...
1
2
3 class Bed{
4     public int price(){return 10000;}
5 }
```

Can Access members of same package & in this can global package

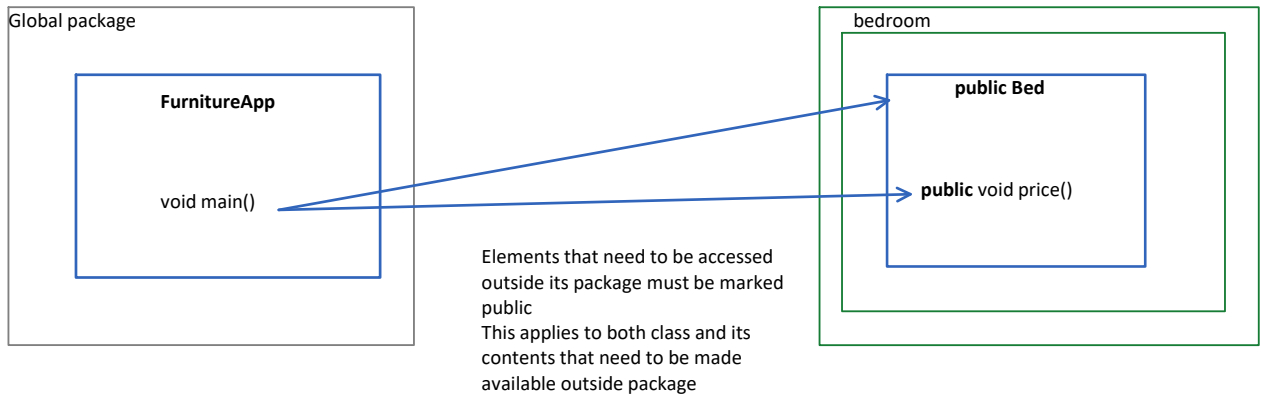
When Using Package



Scope: public

When Using Package

furnitures



Important!

- We don't need all classes and class contents to be directly accessible from outside
- Example
 - You car needs an engine
 - Engine need to be directly accessed by the driver
 - You access car using few public elements like
 - Steering
 - Gear
 - Clutch, Break, Acclerator Paddles
 - This public components internally use other components which are not directly used by drivers
- We can make
 - public class Steering{}
 - class Engine {} ← no scope is package scope

Assignment02

Tuesday, October 20, 2020 12:11 PM

Complete the furniture shop app by

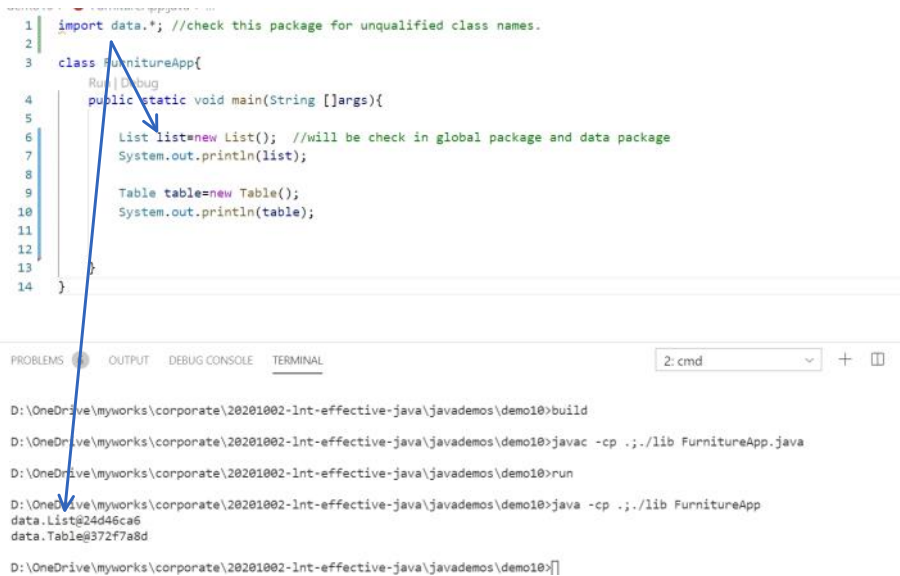
- Adding the right packages
- Use all the classes from all the package in the main function
- Update build and run script
- Build your project
- Run your project
- Take a screen shot of running code
- Update everything **Assignment02 folder**

Package Import

Tuesday, October 20, 2020 12:54 PM

import package.*; or import an entire package

- It imports all the packages from the given package
- When you use an unqualified name it searches for this name in
 - Global package
 - Imported package
- **NOTE**
 - **import package.*** imports only current package and its classes
 - It **doesn't** import subpackages
 - **import furnitures.*** will not import subpackages or their classes like furnitures.bedroom.bed
 - You must import
 - **import furnitures.bedroom.*;**



```
1 import data.*; //check this package for unqualified class names.
2
3 class FurnitureApp{
4     public static void main(String []args){
5
6         List list=new List(); //will be check in global package and data package
7         System.out.println(list);
8
9         Table table=new Table();
10        System.out.println(table);
11
12
13    }
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: cmd + -

```
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo10>build
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo10>javac -cp ./lib FurnitureApp.java
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo10>run
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo10>java -cp ./lib FurnitureApp
data.List@24d46ca6
data.Table@372f7a8d
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo10>]
```

Problem with the wildcard (*) import

```
FurnitureApp.java X
demo10 > FurnitureApp.java > FurnitureApp > main(String[])
1 import data.*; //check this package for unqualified class names.
2 import furnitures.bedroom.*;
3 import furnitures.livingroom.*;
4
5 class FurnitureApp{
6     public static void main(String []args){
7
8         List list=new List(); //will be check in global package and data package
9         System.out.println(list);
10
11         Table table=new Table(); //CONFLICT: which Table?
12         System.out.println(table);
13
14         Bed bed=new Bed(); //no conflict
15         System.out.println(bed);
16
17         Chair chair=new Chair(); //no conflict mode
18         System.out.println(chair);
19     }
20 }

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL
FurnitureApp.java:11: error: reference to Table is ambiguous
Table table=new Table();
^
both class furnitures.livingroom.Table in furnitures.livingroom and class data.Table in data match
FurnitureApp.java:11: error: reference to Table is ambiguous
Table table=new Table();
^
both class furnitures.livingroom.Table in furnitures.livingroom and class data.Table in data match
2 errors
```

Recommendation

Avoid wildcard imports

Single class Import (Selective Import)

- Selecting import imports a single class at a time
- They can override wild card import
- If single class import is specified it will be preferred to resolve conflicts coming from wild card import

```
FurnitureApp.java X
demo10 > FurnitureApp.java > ...
1 import data.*; //check this package for unqualified class names.
2 import furnitures.bedroom.*;
3 import furnitures.livingroom.*;
4 //avoiding wild card conflicts
5 import data.Table; //unqualified Table means Data.Table
6
7 class FurnitureApp{
8     public static void main(String []args){
9
10         List list=new List(); //will be check in global package and data package
11         System.out.println(list);
12
13         Table table=new Table(); //data.Table
14         System.out.println(table);
15
16         Bed bed=new Bed(); //no conflict
17         System.out.println(bed);
18
19         Chair chair=new Chair(); //furnitures.livingroom.Chair
20         System.out.println(chair);
21
22         //How do I resolve furnitures.livingroom.Table
23         //by using fully qualified paths
24         furnitures.livingroom.Table table2=new furnitures.livingroom.Table();
25         System.out.println(table2);
26
27         //same goes for office furnitures
28
29     }
30 }
```

overrides name conflict

Must have fully qualified name for a second conflict class

A Good Package Name?

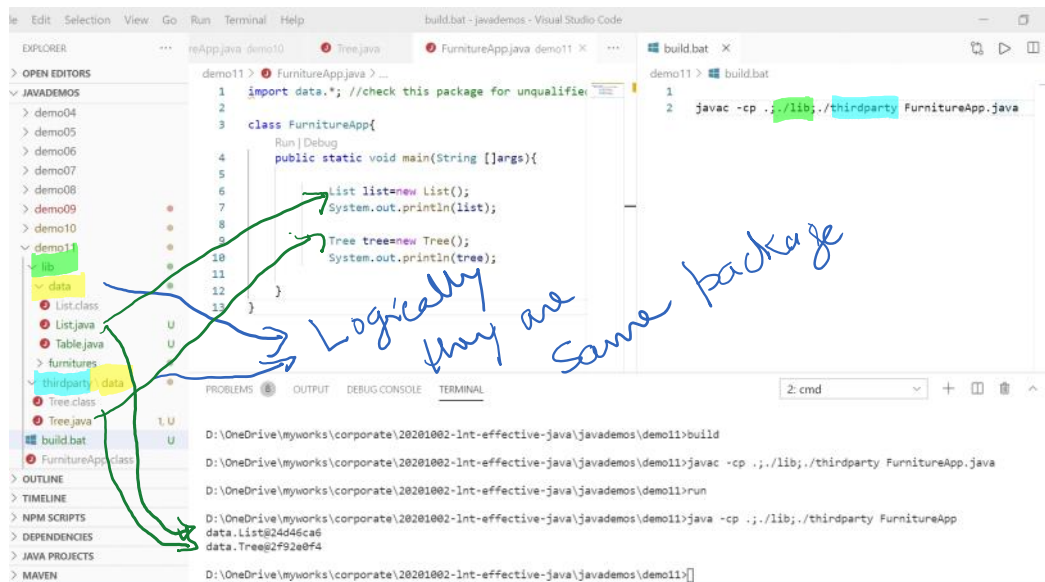
Tuesday, October 20, 2020 1:11 PM

A Package Name Avoid class name conflict. What if Package Name Conflict?

- What if two developer choose to create same package **data**
- Is it likely?
 - YES.

How to resolve package name conflicts?

- Package name doesn't conflict. They merge!



- Package with same in different physical paths are considered to be same package.
- There names don't conflict.
- The content of two packages are treated as part of same package

What is the probability that two different developers would end up creating a package called **data** and have a class inside this package called **List**?

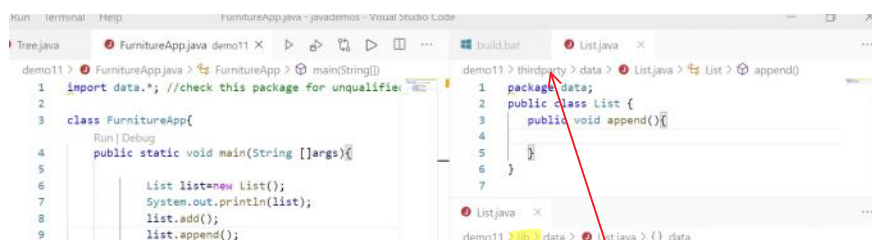
- It is very likely.
- List is a popular element in programming (and real world)
- A List is most likely be present in a package which would be called
 - data
 - collection
 - Datastructure
- Because these names are few, it is highly likely that many developers would use the same packages to house same classes

What happens if two different package with same name exists in the class path

- They merge as one package

What happens if these two different package has class with same name?

- The class name conflicts
- **There is no resolution to this problem**
 - Java has no solution to this problem




```
4 public static void main(String []args){
5
6     List list=new List();
7     System.out.println(list);
8     list.add();
9     list.append();
10    Tree tree=new Tree();
11    System.out.println(tree);
12
13
14 }
```

```
5 }
6 }
7 }
8 }
9 }
10 }
11 }
12 }
13 }
14 }
```

```
1 package data; //we decided that my package starts
2
3 public class List{
4     public void add(){
5         System.out.println("Item added to the list");
6     }
7 }
```

```
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo11>build
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo11>javac -cp .;./lib;./thirdparty FurnitureApp.java
FurnitureApp.java:9: error: cannot find symbol
    list.append();
    ^
  symbol:   method append()
  location: variable list of type List
1 error
```

- Due to classpath order, javac/java never sees the second class with conflicting name
- Since Java does see it and doesn't complain for it, there is no way to resolve it

Conventional Solution

- Make sure your package name is unique
- Generally we never create a one level package (eg. Data)
- We create nested package
- The root package should be an identity/branding package
 - E.g.
 - vivek.data ← data belongs to vivek
 - santosh.data ← data belongs to santosh

Recommendations

- Make sure your identity package (root package) is unique
- One way to ensure is to use copyrighted names as identity
 - Your name is not copyrighted
- Company Name is copyrighted
- We generally use our domain name as package name (in reverse order)
 - Example
 - in.conceptarchitect.data
 - com.ltts.project19.data
 - com.ltts.commons.data
 - in.conceptarchitect.furnitures.office
- A package may include project name or department name as sub package in case or large organization

Problem!

- What is the probability that two vivek will create a package called data and have a class inside called List?
 - High Probability
 - Human names are quite common
 - Not a great choice for avoiding name conflict

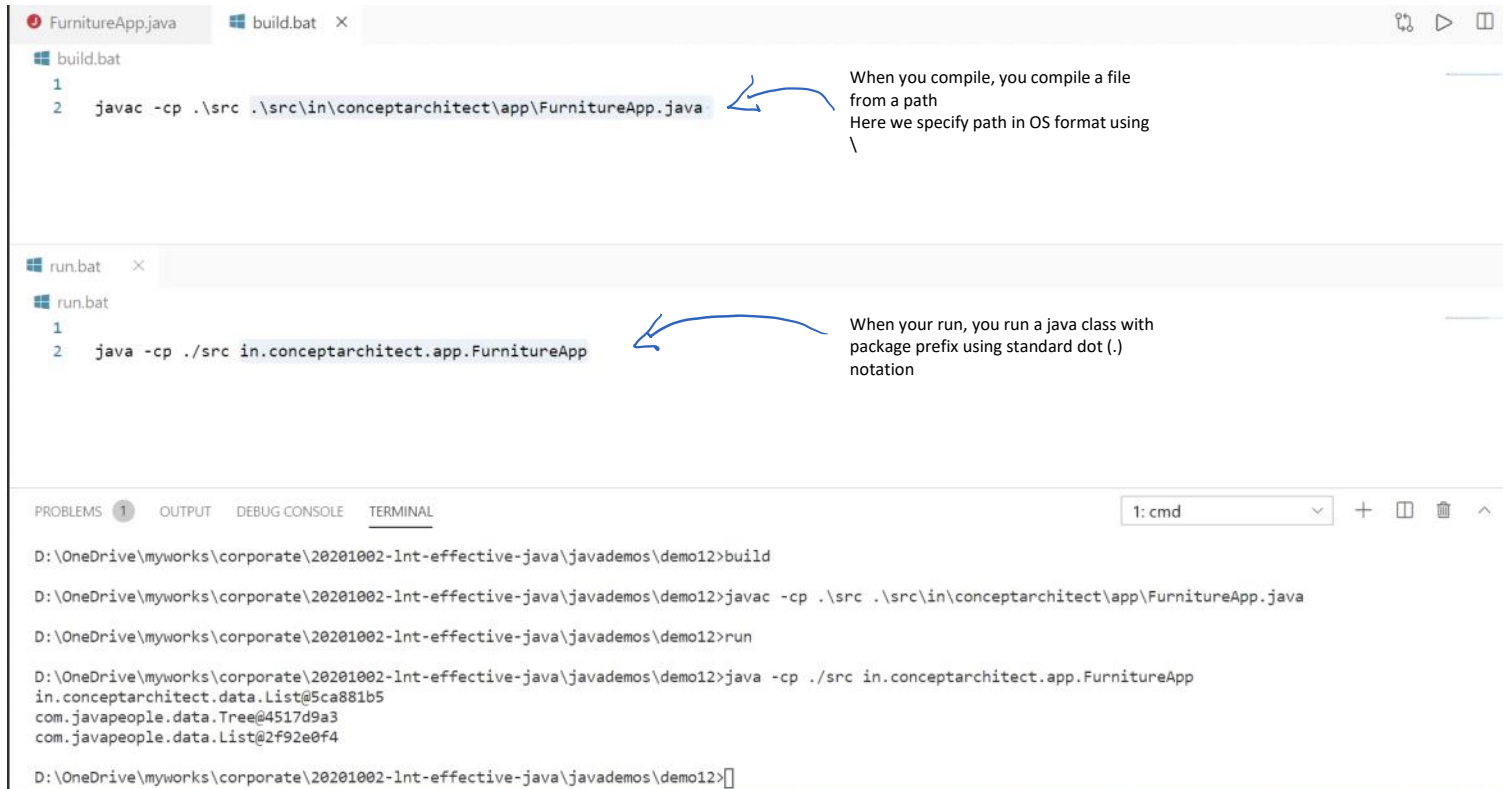
Assignment03

Tuesday, October 20, 2020 1:40 PM

- Start with demo11
- Create your brand:
 - You may use
 - in.surname.name as your package
- Add identity spaces for **lib packages** as **your brand package**
- Add identity package **com.javapeople** as identity to the thirdparty library
- Use all the classes created so far
- Create build path and screenshots.

Building Project with nested classes

Tuesday, October 20, 2020 3:19 PM



The screenshot shows an IDE with two tabs: `FurnitureApp.java` and `build.bat`. The `build.bat` tab is active, showing the following content:

```
build.bat
1
2 javac -cp .\src .\src\in\conceptarchitect\app\FurnitureApp.java
```

A blue arrow points from the text "When you compile, you compile a file from a path Here we specify path in OS format using \\" to the file path in the `javac` command.

Below the `build.bat` tab is the `run.bat` tab, showing the following content:

```
run.bat
1
2 java -cp ./src in.conceptarchitect.app.FurnitureApp
```

A blue arrow points from the text "When your run, you run a java class with package prefix using standard dot (.) notation" to the package name `in.conceptarchitect.app` in the `java` command.

At the bottom of the IDE is the `TERMINAL` panel, showing the execution of the build and run commands:

```
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo12>build
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo12>javac -cp .\src .\src\in\conceptarchitect\app\FurnitureApp.java
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo12>run
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo12>java -cp ./src in.conceptarchitect.app.FurnitureApp
in.conceptarchitect.data.List@5ca881b5
com.javapeople.data.Tree@4517d9a3
com.javapeople.data.List@2f92e0f4
D:\OneDrive\myworks\corporate\20201002-Int-effective-java\javademos\demo12>
```

Organizing your code for Deployment

Tuesday, October 20, 2020 3:21 PM

```
D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo12>tree/f
Folder PATH listing for volume Data
Volume serial number is F8C2-FD66
D:.
  build.bat
  run.bat
  src
    com
      javapeople
        data
          List.class
          List.java
          Tree.class
          Tree.java
    in
      conceptarchitect
        app
          FurnitureApp.class
          FurnitureApp.java
        data
          List.class
          List.java
          Table.java
        furnitures
          bedroom
            Bed.java
          livingroom
            Chair.java
            Table.java
          office
            Chair.java
            Table.java
```

- Source file and .class files are present in same folder
- Source file is generally not required to run the code
- You may not distribute or share your source code with client. You will give only .class file
- As a developer we keep deleting older version of class files.
- Keeping them separate would be great for application design.

Solution

1. Keep all source files in **src** folder
2. Keep all class files in **classes** folder

Javac -d switch

- You can specify -d switch on javac to specify the folder in which you will store the class files
- In case of packages it will automatically create the package sub folders

```
classes
  com
    javapeople
      data
        List.class
        Tree.class
  in
    conceptarchitect
      app
        FurnitureApp.class
      data
        List.class
      furnitures
        bedroom
          Bed.class
src
  com
    javapeople
      data
        List.java
        Tree.java
  in
    conceptarchitect
      app
        FurnitureApp.java
      data
        List.java
        Table.java
      furnitures
        bedroom
          Bed.java
        livingroom
          Chair.java
          Table.java
```

1. Classes are organized and stored separately
2. Only those classes required by the client is compiled
3. You can distribute the classes folder and not source folder
4. You can delete all classes at once by deleting the classes folder

```
Chair.java
Table.java
office
Chair.java
Table.java
```

Deployment

Tuesday, October 20, 2020 3:56 PM

A java program can with

- Class files in the disk
- Sub folders representing a package
- A batch file to run the command and class path

Problem

- In a typical java program, there would dozens of package (folders) and hundreds of class (files)
- To distribute the code to clients we need to copy all these files and folders
- Sharing so many files may require us to
 - Zip file files and send to client
 - Ask client to unzip in the right folder
 - Tell which java class contains main
 - Tell them the command arguments

Solution

- Create a jar file
- A jar file is like a zip file
- A java program can run from a jar file without needing to unpack it.
- You need to share a single jar file

1. Create a Jar File

Method 1 — Creating a simple jar

- This is the common method for creating a simple jar

```
c:> jar c v f app.jar .
```

```
c--> create a jar
v --> verbose – print whatever your are doing
f -> we will specify the name of output jar file (if not give the data is
simply shown on console and no jar created)
. -> Jar files and subdirectories of current folder
```

- Running code from a simple jar

```
c:> java -cp app.jar in.conceptarchitect.app.FurnitureApp
```

- We will specify the name of our jar as the CLASSPATH
- We will specify the full qualified name of the MainClass which contains the main function

A jar manifest

- A java jar file contains a manifest file ./META-INF/manifest.mf which contains meta informations about jar file
- This file is automatically added to every jar and contains following sample information

```
//META-INF/manifest
Manifest-Version: 1.0
Created-By: 15 (Oracle Corporation)
```

- To add more information to manifest we can
 - Create a text file with additional information
 - Specify the text file while creating the jar file using "m" option
 - The content of your text file will be merged in standard manifest

```
c:> jar cvfm app.jar meta.txt .
```

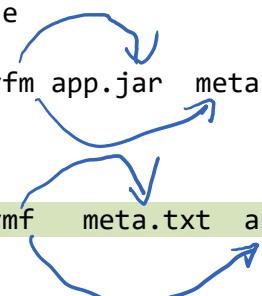
- Note app.jar is final jar file
- meta.txt contains additional data to be added to jar manifest
- The order in which you specify the two files depends on the order in which the options "m" and "f" is applied

- You can use

```
c:> jar cvfm app.jar meta.txt .
```

Or

```
c:> jar cvmf meta.txt app.jar .
```



Important Meta data

Main-Class:

- Specifies your main class within the jar

Class-Path:

- Specify additional class paths that you may need.

Running an application from a jar file that has Main-Class manifest entry

If your jar file has manifest entry you can use it to execute application simply by running

```
c:> java -jar app.jar
```

- This command automatically runs class mentioned in Main-Class of manifest.

Method 3 -- inject Main Class Manifest only

- In case you just want to specify MainClass to manifest and not other details such as version, author etc you have an alternative jar creation syntax

```
c:> jar cvef in.conceptarchitect.app.FurnitureApp app.jar .
```

- "e" stands for entry point or Main-Class
- You must specify the main class
- This is automatically added in Manifest.mf as Main-Class:
 - You don't need to create a meta.txt file just to specify Main-Class
 - If you have to specify other information, you need to specify manifest