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

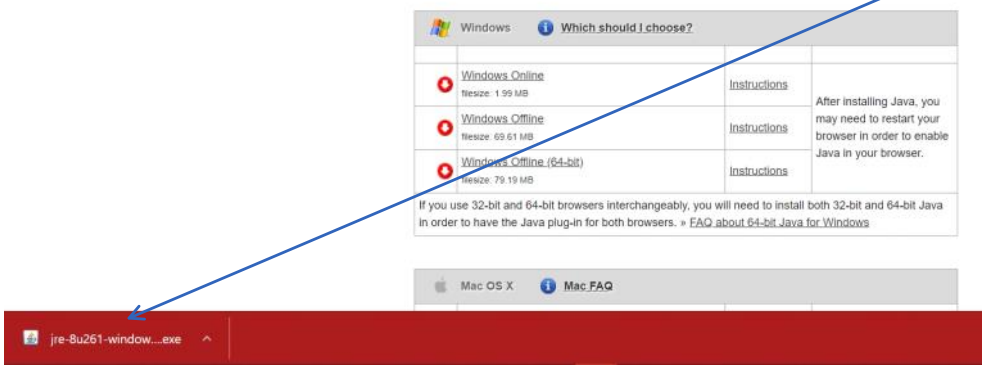
Is this claim valid?

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 lanugage?**
  - **Do everyone needs to program in Java?**
- **When you try to download "java", what are your actually downloading?**



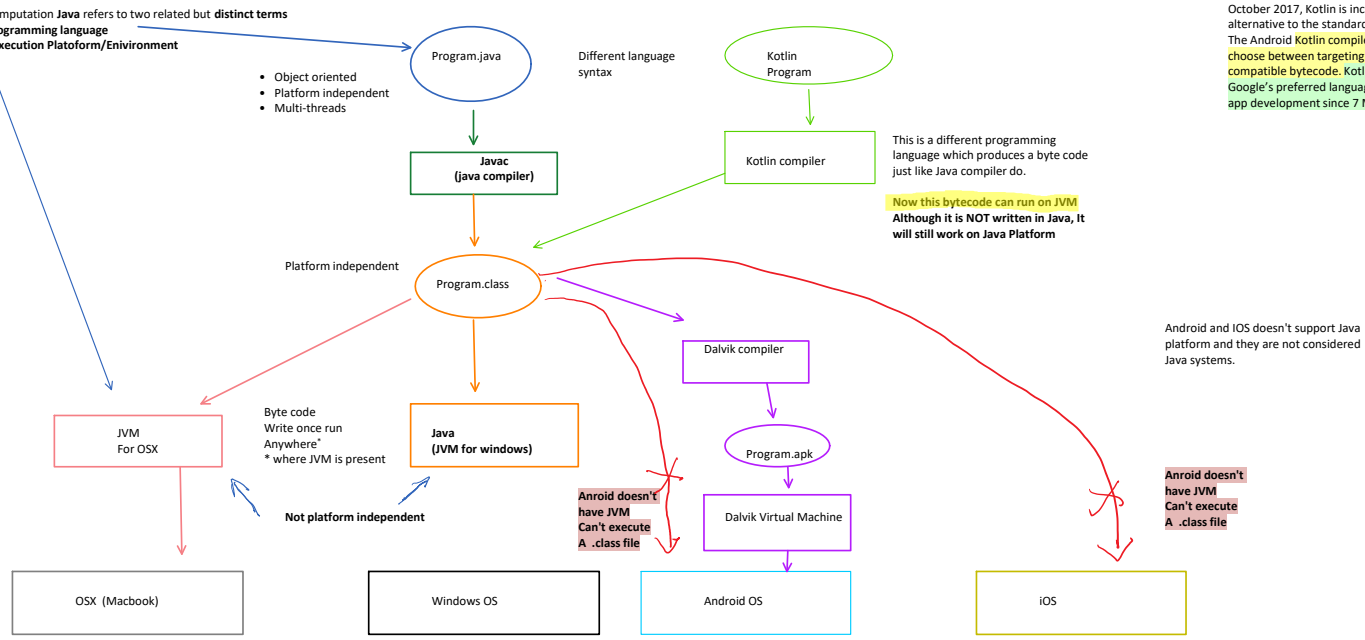
- Why request for a Java download ended up downloading JRE?
- Is JRE and object oriented programming language?
- Is JRE platform independent
- Can you write an object oriented programming language using JRE?

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 Platoform/Enivironment**



- Object oriented
- Platform independent
- Multi-threads

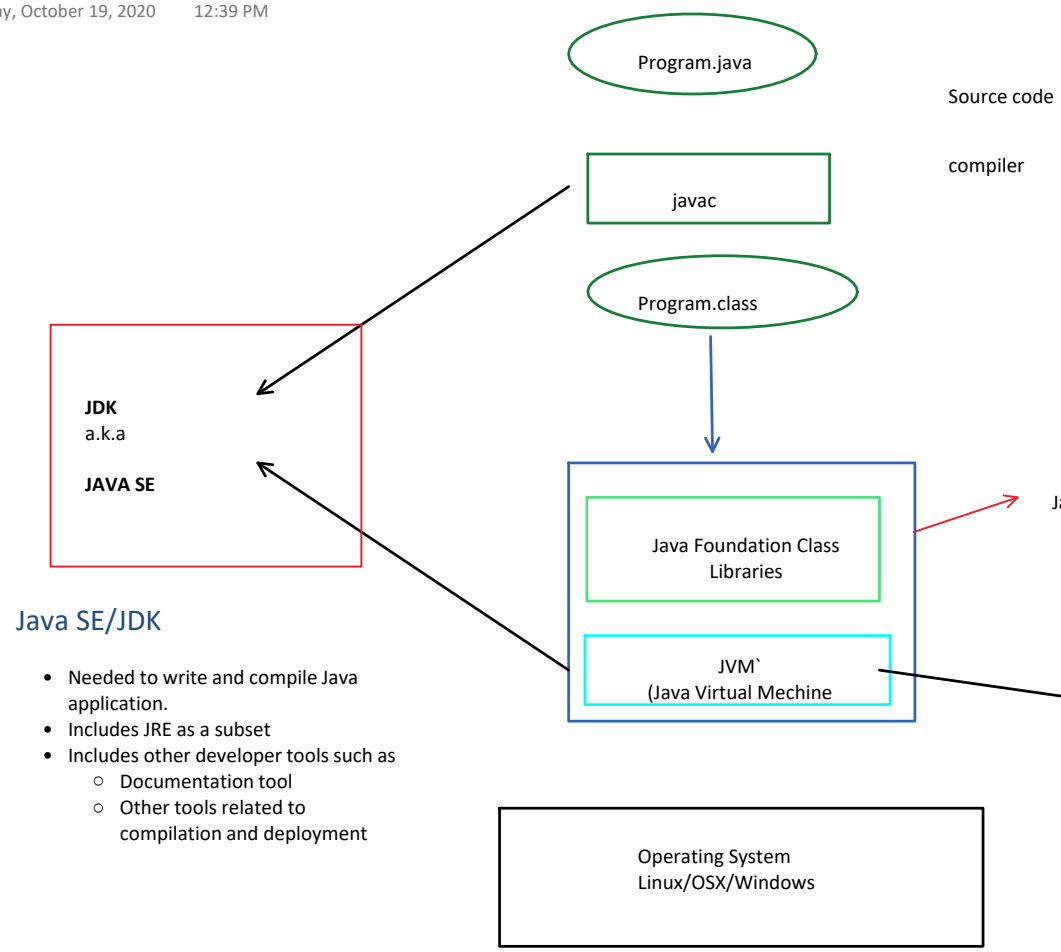
Since the release of Android Studio 3.0 in October 2017, Kotlin is included as an alternative to the standard Java compiler. The Android Kotlin compiler lets the user choose between targeting Java 6 or Java 8 compatible bytecode. Kotlin has been Google's preferred language for Android app development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

Android and IOS doesn't support Java platform and they are not considered Java systems.

Dalvik Virtual machine is an customized non-standard Runtime which is different from JVM and the two are **NOT** interoperable

Java Framework

Monday, October 19, 2020 12:39 PM



Java

- Required by users who plan to run Java based softwares in their system.
- These people are not developing Java applications.

Java SE/JDK

- Needed to write and compile Java application.
- Includes JRE as a subset
- Includes other developer tools such as
  - Documentation tool
  - Other tools related to compilation and deployment

- JVM is the interpreter here
- JRE is **not platform independent but designed for specific OS**
- **JRE makes user code (bytecode) platform independent**

Java Environments based on Use cases

Monday, October 19, 2020 12:49 PM

Java desktop application user

- Needs to install JRE
- The product will include additional library

Server needs to install
JRE+ Web Runtime (like web server)

Webapp user

Just needs standard browser.
Doesn't require to install anything related to java

- NO JRE
- NO other software

Java Desktop API

Java Web API

JDK

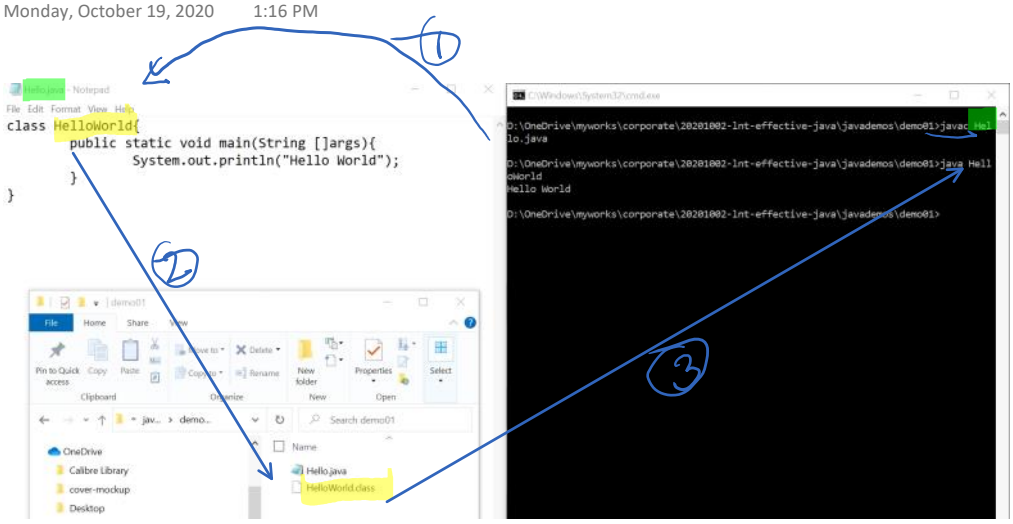
contains

JAVA Runtime Environment

Contains JVM
Foundation class libraries

Compiling and Running Java code

Monday, October 19, 2020 1:16 PM



1. You compile a source file

```
c:> javac Hello.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

*FactorialApp.java - Notepad

File Edit Format View Help

```
class FactorialApp{
        public static void main(String []args){
                int n=5;
                int fn=factorial(n); //user defined function

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

        int factorial(int x){

                int fx=1;
                while(x>1)
                        fx*=x--;

                return fx;
        }
}
```

C:\Windows\System32\cmd.exe

```
D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo02>javac Fac
torialApp.java
FactorialApp.java:5: error: non-static method factorial(int) cannot be referenced fr
om a static context
                int fn=factorial(n); //user defined function
                       ^
1 error

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo02>
```

A static method can't access non-static method of the class directly

Important

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

```
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{

        int calculate(int x){

                int fx=1;
                while(x>1)
                        fx*=x--;

                return fx;
        }
}
```

Static context

```
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\de
mo04

10/19/2020  02:38 PM    <DIR>          .
10/19/2020  02:38 PM    <DIR>          ..
10/19/2020  02:38 PM               291 FactorialApp.java
               1 File(s)            291 bytes
               2 Dir(s)  603,080,458,240 bytes free

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04>javac Fac
torialApp.java
FactorialApp.java:5: error: non-static method calculate(int) cannot be referenced fr
om a static context
                int fn=Factorial.calculate(n);
                                 ^
1 error

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

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.

FactorialApp.java - Notepad

File Edit Format View Help

```
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;
        }
}
```

Non static

instance of object level Method

C:\Windows\System32\cmd.exe

```
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\de
mo04

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 Fac
torialApp.java

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo04>java Fact
orialApp
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\de
mo04

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

FactorialApp.java - Notepad

File Edit Format View Help

```
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;
        }
}
```

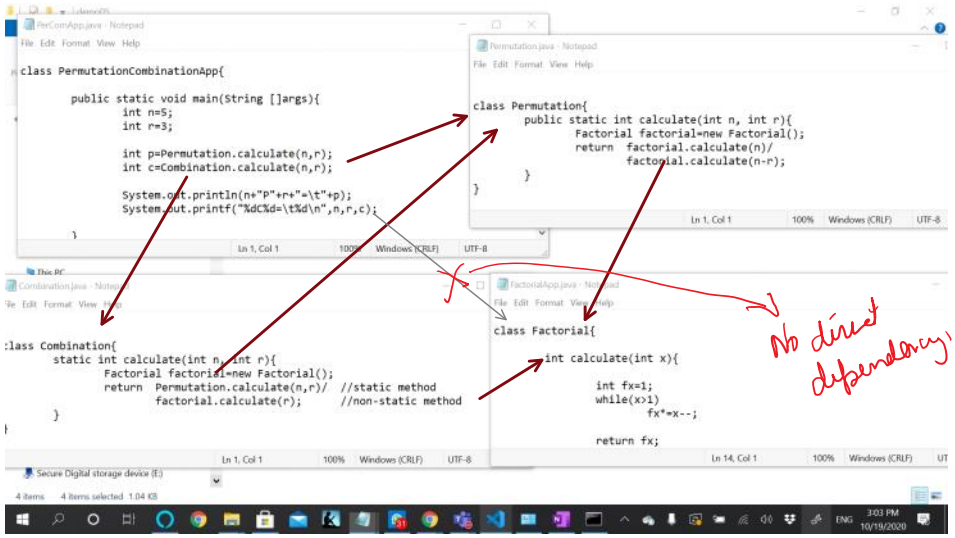
C:\Windows\System32\cmd.exe

```
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\de
mo03

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 Fac
torialApp.java

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo03>java Fact
orialApp
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\de
mo03

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>
```

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

Class Dependencies

Monday, October 19, 2020 3:03 PM



- Unlike c/c++ (or event 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 file 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
Combination.class	✓	10/19/2020 3:27 PM	CLASS File	1 KB
PermutationCombinationApp.class	✓	10/19/2020 3:27 PM	CLASS File	2 KB
Combination.java	✓	10/19/2020 3:26 PM	JAVA File	1 KB
Factorial.class	✓	10/19/2020 3:20 PM	CLASS File	1 KB
Permutation.class	✓	10/19/2020 3:20 PM	CLASS File	1 KB
Permutation.java	✓	10/19/2020 3:19 PM	JAVA File	1 KB
FactorialApp.java	✓	10/19/2020 3:18 PM	JAVA File	1 KB
Factorial.java	✓	10/19/2020 3:17 PM	JAVA File	1 KB
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\20201002-1nt-effective-java\javademos\demo05>javac PerComApp.java

D:\OneDrive\myworks\corporate\20201002-1nt-effective-java\javademos\demo05>
```

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

```
C:\Windows\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
```

Doesn't know where to search!

Java compiler by default doesn't know where to search for the java/class files if it is not present int 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 separted 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-lnt-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-lnt-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-lnt-effective-java\javademos\demo06>set classpath=.\lib\console;.\lib\maths;.

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

D:\OneDrive\myworks\corporate\20201002-lnt-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

Tuesday, October 20, 2020 9:48 AM

What is a Table?

This?

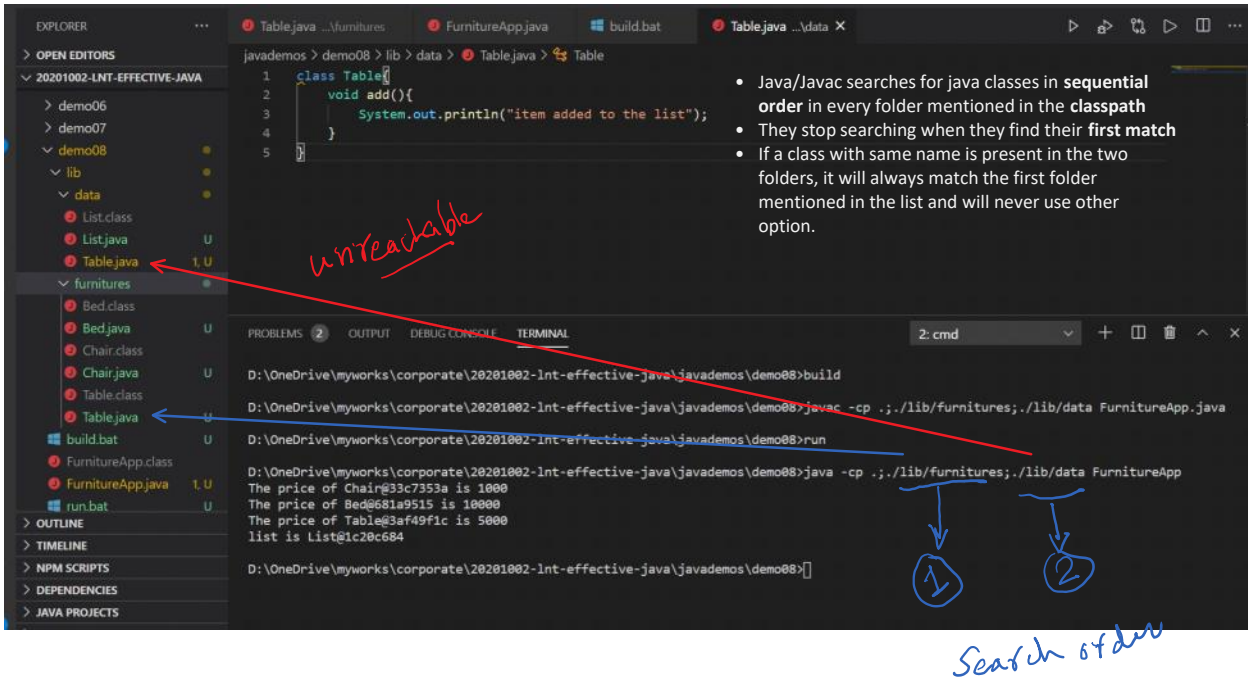
That?

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

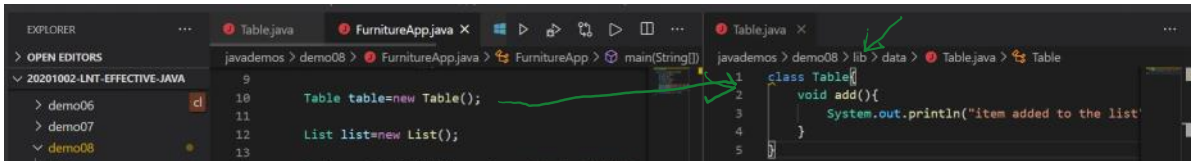
- In real world a word can have different meanings. Same word (like Table) can represent multiple different and unrelated elements.
- Often in a single project we may need to use one or more such objects
  - Example:
    - A Furntiure Shop sells **Table (Furnitures)**
    - The maintain their stock details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

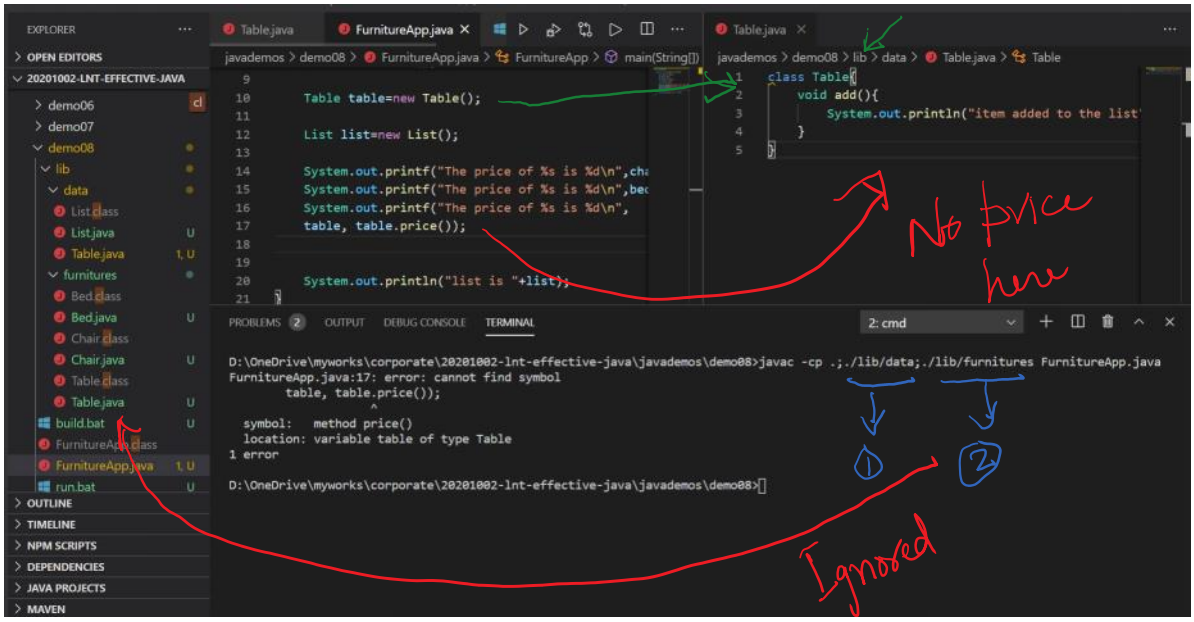
How do I represent Multiple Objects with same name in same application

Why Multiple Folder Design Doesn't work?



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**

```
Volume serial number is F8C2-FD66
D:\ONEDRIVE\MYWORKS\CORPORATE\20201002-LNT-EFFECTIVE-JAVA\JAVADEMOS\DEMO09
    build.bat
    FurnitureApp.java
    run.bat
    lib
      data
          List.java
          Table.java
      furnitures
        bedroom
            Bed.java
        livingroom
            Chair.java
            Table.java
        office
            Chair.java
            Table.java
```

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

- 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

```
Volume serial number is F8C2-FD66
D:\ONEDRIVE\MYWORKS\CORPORATE\20201002-LNT-EFFECTIVE-JAVA\JAVADEMOS\DEMO09
    build.bat
    FurnitureApp.java
    run.bat
    lib
      data
          List.java
          Table.java
      furnitures
        bedroom
            Bed.java
        livingroom
            Chair.java
            Table.java
        office
            Chair.java
            Table.java
```

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	

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**

FurnitureApp.java List.java

demo10 > lib > data > List.java > {} data

```
package data;  //we decided that my package starts

class List{
    void add(){
        System.out.println("item added to the list"
    }
}
```

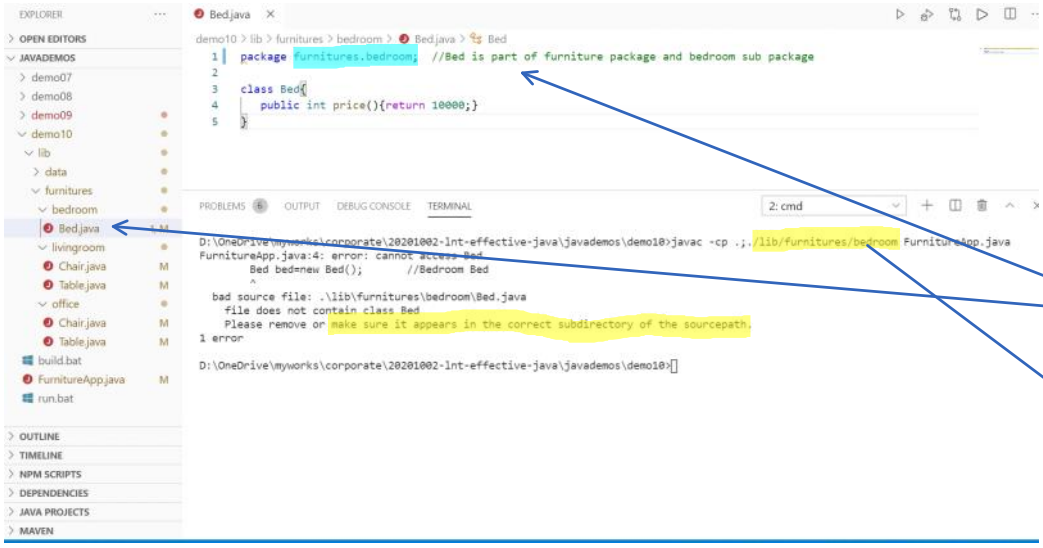
Bed.java

demo10 > lib > furnitures > bedroom > Bed.java > Bed

```
package furnitures.bedroom;  //Bed is part of fur

class Bed{
    public int price(){return 10000;}
}
```

Using Package

Tuesday, October 20, 2020 11:35 AM



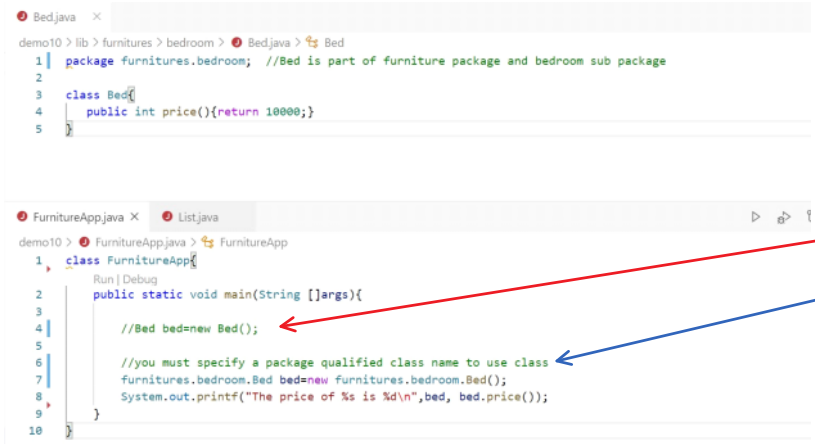
- Compiler is expecting **Bed** to be present in a folder
  - furnitures/bedroom
- **Is it not already present in the right folder?**
  - No.
  - Because we are searching for this folder by going inside this folder.
    - Remember this path is part of **classpath**
    - Javac goes into the folder and 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[])

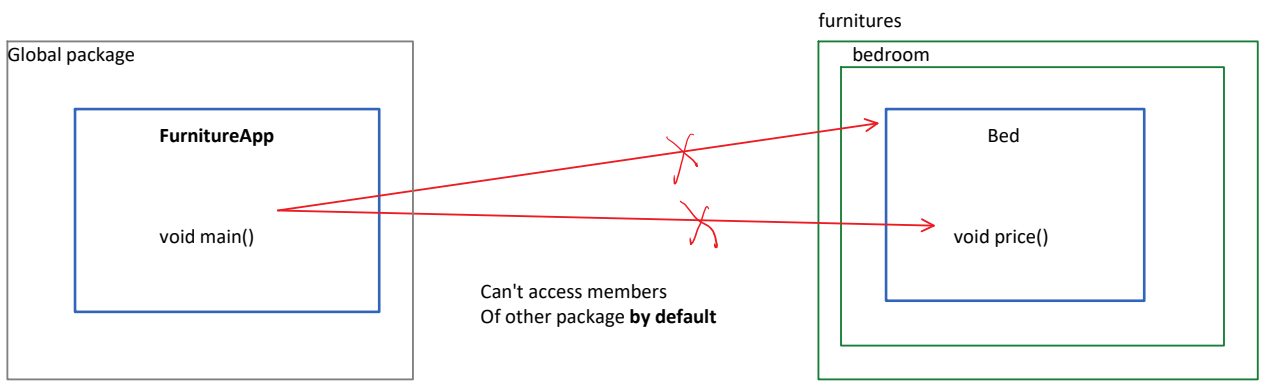
```
4     public static void main(String []args){
5
6         Chair chair=new Chair();  //Living room chair
7
8         Bed bed=new Bed();        //Bedroom Bed
9
10        Table table=new Table(); //living room table
11
12        List list=new List();
13
14        System.out.printf("The price of %s is %d\n",chair, chai
15        System.out.printf("The price of %s is %d\n",bed, bed.pr
16        System.out.printf("The price of %s is %d\n",
17        table, table.price());
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 case global package

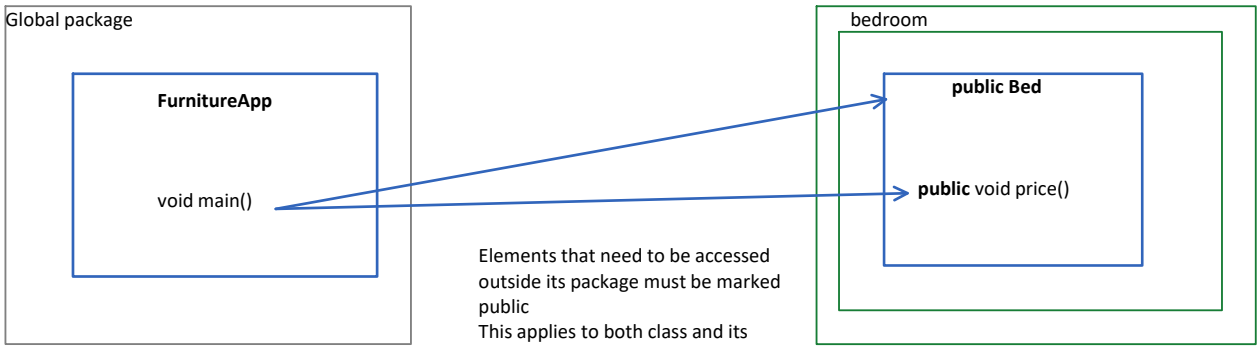
When Using Package



Scope: public

When Using Package

furnitures



Elements that need to be accessed outside its package must be marked public
This applies to both class and its contents that need to be made available outside package

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, Acclereator 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 packagename.*; 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 packagename.*** 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.*;**

```java
import data.*; //check this package for unqualified class names.

class FurnitureApp{
    Run | Debug
    public static void main(String []args){

        List list=new List();  //will be check in global package and data package
        System.out.println(list);

        Table table=new Table();
        System.out.println(table);


    }
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: cmd

```
D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo10>build

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo10>javac -cp .;./lib FurnitureApp.java

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo10>run

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo10>java -cp .;./lib FurnitureApp
data.List@24d46ca6
data.Table@372f7a8d

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo10>
```

Problem with the wildcard (*) import

FurnitureApp.java

demo10 > FurnitureApp.java > FurnitureApp > main(String[])

```java
import data.*; //check this package for unqualified class names.
import furnitures.bedroom.*;
import furnitures.livingroom.* ;

class FurnitureApp{
    Run | Debug
    public static void main(String []args){

        List list=new List();  //will be check in global package and data package
        System.out.println(list);

        Table table=new Table();  //CONFLICT: which Table?
        System.out.println(table);

        Bed bed=new Bed();       //no conflict
        System.out.println(bed);

        Chair chair=new Chair(); //no conflict mode
        System.out.println(chair);
```

which Table

No conflict case

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL 2: cmd

```
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

demo10 > FurnitureApp.java > ...

```java
import data.*; //check this package for unqualified class names.
import furnitures.bedroom.*;
import furnitures.livingroom.* ;
//avoiding wild card conflicts
import data.Table;  //unqualified Table means Data.Table

class FurnitureApp{
    Run | Debug
    public static void main(String []args){

        List list=new List();  //will be check in global package and data package
        System.out.println(list);

        Table table=new Table();  //data.Table
        System.out.println(table);

        Bed bed=new Bed();       //no conflict
        System.out.println(bed);

        Chair chair=new Chair(); //furnitures.livingroom.Chair
        System.out.println(chair);

        //How do I resolve furnitures.livingroom.Table
        //by using fully qualified paths
        furnitures.livingroom.Table table2=new furnitures.livingroom.Table();
        System.out.println(table2);

        //same goes for office furnitures

    }
```

overrides name conflict

Must have fully qualified name for a second conflicting class

A Good Package Name?

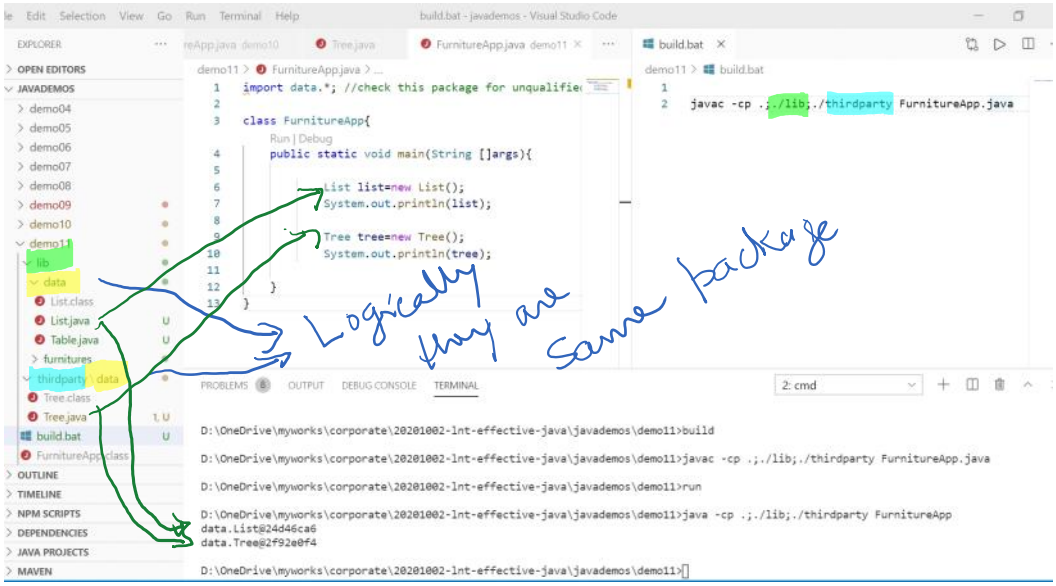
Tuesday, October 20, 2020 1:11 PM

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?**

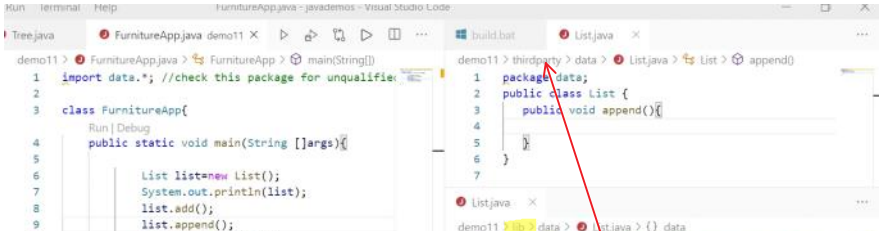
- It is very likely.
- List is a popular element in programming (and real world)
- A List is most likely be present in a package which would be called
  - data
  - collection
  - Datastructure
- Because these names are few, it is highly likely that many developers would use the same packages to house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



- Due to classpath order

```
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
```

List.java

demo11 > lib > data > List.java > {} data

```
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   }
```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL 2: cmd

```
D:\OneDrive\myworks\corporate\20201002-1nt-effective-java\javademos\demo11>build

D:\OneDrive\myworks\corporate\20201002-1nt-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

D:\OneDrive\myworks\corporate\20201002-1nt-effective-java\javademos\demo11>
```

- 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 pacakge (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.projedt19.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 Probality
  - 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

FurnitureApp.java | build.bat

build.bat

```
javac -cp .\src .\src\in\conceptarchitect\app\FurnitureApp.java
```

When you compile, you compile a file from a path
Here we specify path in OS format using \

run.bat

run.bat

```
java -cp ./src in.conceptarchitect.app.FurnitureApp
```

When your run, you run a java class with package prefix using standard dot (.) notation

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL 1: cmd

```
D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo12>build

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo12>javac -cp .\src .\src\in\conceptarchitect\app\FurnitureApp.java

D:\OneDrive\myworks\corporate\20201002-lnt-effective-java\javademos\demo12>run

D:\OneDrive\myworks\corporate\20201002-lnt-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-lnt-effective-java\javademos\demo12>
```

Organizing your code for Deployment

Tuesday, October 20, 2020 3:21 PM

```
D:\OneDrive\myworks\corporate\20201002-1nt-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 foders

```
├───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 separtely
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 pragram, 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 arugments

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 ever 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 run class mentioned in Main-Class of manifest.

Method 3 -- inject Main Class Manifest only

- Incase you just want to specify MainClass to manifest and not other details such as verion, 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 Manfiest.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

Assignment04

Tuesday, October 20, 2020 5:22 PM

- Create a class to represent a Bank Account
- A Bank Account will have
  - Account number
  - Name
  - Password
  - Balance
  - Interest rate
- A bank account will support following operations
  - Create an account
  - Deposit money
    - Should fail if amount <=0
  - Withdraw money
    - Should fail if
      - Amount<=0
      - Amount>balance
      - Password supplied doesn't match
  - Credit interest (one month interest)
    - Formula: **balance=balance*rate/1200**
  - Provide getters/setters e.g.
    - Account number can't change in future
    -
- Write a main program to test bank account details (menu driven)

Coding Conventions

Wednesday, October 21, 2020 10:00 AM

- Conventions are not syntactical compulsions.
- Violating convention doesn't cause compile time or runtime error.
- They simply make your code harder to
  - Read
  - Maintain
  - Debug
- Conventions are followed throughout developer community
- When we violate conventions other developers find it difficult to follow my codes

1. Package naming

- Your package name should be all lower case letter
- Package Name should start with two top level brand identity in reverse order or your domain
  - Eg.
    - com.javapeople
    - in.conceptarchitectect
    - org.apache
  - In your design you can assume a domain such as
    - in.you-name or org.your-company
- Your business name should be a package at the depth 3.
- Don't use unbranded package names
- Don't use single layer brands
  - Instead of ~~com.app~~ use **com.javadev.finance.app**
- A package name groups a set of classes
  - Package name should represent the purpose of that grouping
    - com.javadev.finance.data.Table
    - com.javadev.finance.ui.Table

- You may use a super package **com.javatraining**

2. Class Naming

- Your class name should follow Pascal Naming Convention
  - Class Name should begin with an upper case letter
    - Triangle
    - Factorial
  - If using More than one word in class name
    - Each word should begin with upper case letter
    - Example
      - RaceCar
      - LinkedList
      - SortedArray
  - Names should be
    - Meaningful

        - Avoid an abstract name for a concrete class
        - Your name should tell what is the purpose of the class
    - Avoid
        - Names with all upper case letters unless you have very good reason like your class represents and Accronym
            - CID
            - GUID
        - DON'T USE snake case naming
            - Name where multiple words are separated by underscore (_)

3. Method Names

- Your method name should follow **Camel Naming Convention**
    - **Should begin with an lower case letter**
        - calculate()
        - show()
    - If using More than one word in class name
        - Each word starting second should begin with upper case letter
        - Example
            - getBalance()
            - toString()
            - createSavingsAccount()
    - Names should be
        - Meaningful
        - Avoid an abstract name for a concrete class
        - Your name should tell what is the purpose of the class
    - Avoid
        - Names with all upper case letters unless you have very good reason like your class represents and Accronym
            - generateGUID()
        - DON'T USE snake case naming
            - Name where multiple words are separated by underscore (_)
        - **Sometimes to highlight an private operation the name may begin with an underscore**
            - **Acceptable. But avoid if you can**

4. Field Names

- Your field name should follow **Camel Naming Convention**
    - **Should begin with an lower case letter**
        - balance
        - amount
    - If using More than one word in class name
        - Each word starting second should begin with upper case letter
        - Example
            - intrestRate
    - Names should be
        - Meaningful
        - Avoid an abstract name for a concrete class
        - Your name should tell what is the purpose of the class
    - Avoid
        - Names with all upper case letters unless you **are defining a constant**
            - PI
            - MAX
        - DON'T USE snake case naming

- Name where multiple words are separated by underscore (_)

- **Sometimes to highlight an private operation the name may begin with an underscore**

What is Object Oriented Programming? (Popular Perception)

Wednesday, October 21, 2020 10:27 AM

Participants Feedbacks

- Creating Objects that contains data and logic
- Programming with class and objects
- Real world objects like car etc.

- Why do I create an object for data and logic?
- Why not create logic as functions and data as simple structure like in C language?

- Why should I really program class and objects?
- Why do we call it **object oriented programming** and not **class oriented programming** or **class and object oriented programming.**

- Why do we need real world entities?
  - Because we program to solve a real world problem

What is a Program?

A set of instructions
Given to computer
For solving a problem.

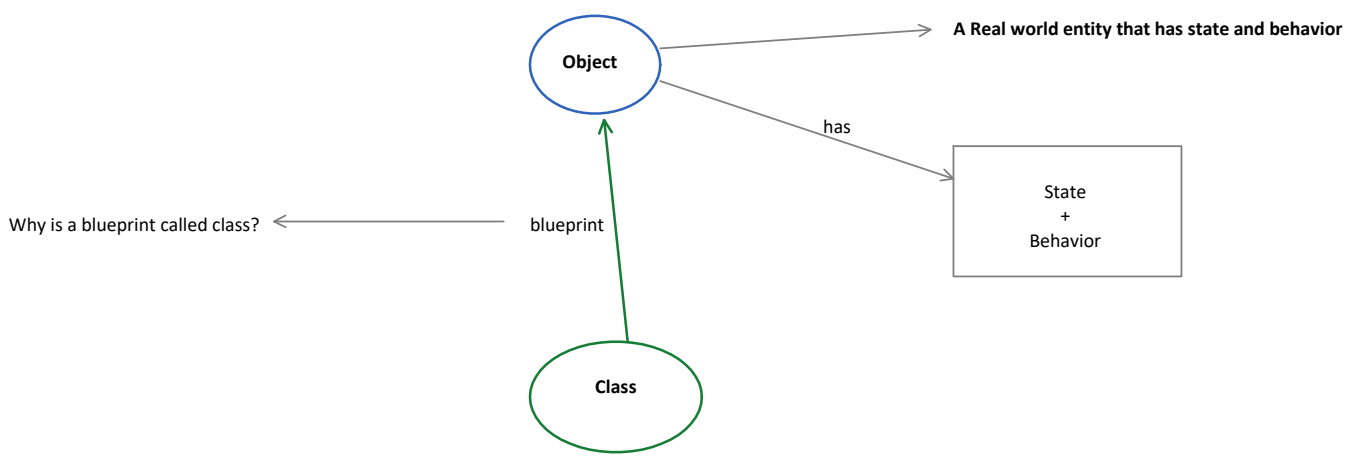
- Where is object in (the definition) of Program?
- Why is there no reference to either object or class in definition of a Program?

What is an Object Oriented Program?

A **set of instructions** constructed by using **classes** and creating ~~blueprint of that class called~~ an **objects** to use the functionalities available in that class.

Three important words in the definition in decreasing order of priority
1. set of instruction
2. class
3. object

What is an Object?



Let's talk about a real world Object

Yamaha Alien —> A New Bike

- **Colors**
- **Mileage**
- **CC**
- **Cost**
- **BHP**
- **AT/MT**

What is the source of your knowledge about Bike?

- **Because you use it**

Dinasourous?

- Animal that existed years ago
- Lived in Jurassic park

What is the source of your knowledge about Dinasourous?

- Have you seen it?
    - Movies
    - Wikpedia

Object Oriented Design

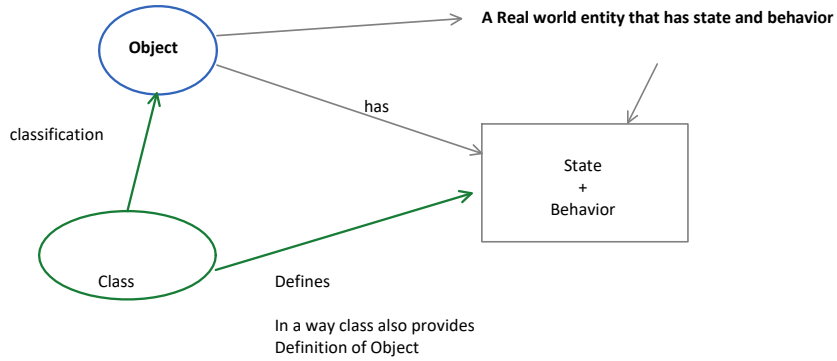
Wednesday, October 21, 2020 10:57 AM

Domain
The problem space

Object

- ~~**Represents a Real World Physical Entity**~~
- **Object represents a Domain Entity**
- **An Object can be**
  - **Real World Physical --> Employee**
  - **Real World Non Physical --> Department**
  - **Non Real Physical --> Magic Wand**
  - **None Real World Non Physical --> Magic Wand**

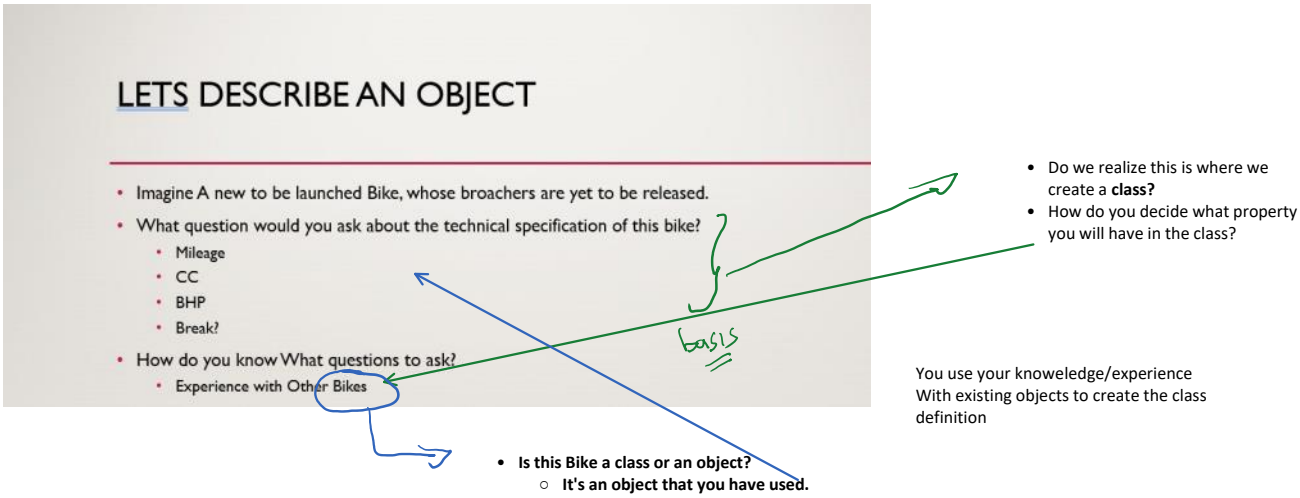
- Has its State and Behavior
- **Object is anything/everything that you can conceptualize**



Class

- Class stands for Classfication
- Class is the basis of object classfication
- It defines parameters for such classfication
  - State
  - behaviors

What comes First — Object or Class?



Why class comes first?

- Class is a language of feature of popular object oriented language like
  - C++
  - Java
  - C#
- It is a semnatical compulsion but **not a real requirement for** Object Oriented Programming
  - Remember we call is Object Oriented and not class Oriented
- There are Object Oriented langauges where Class doesn't even exist
  - Javascript
  - New version has class, but that's optional

Creating An Object

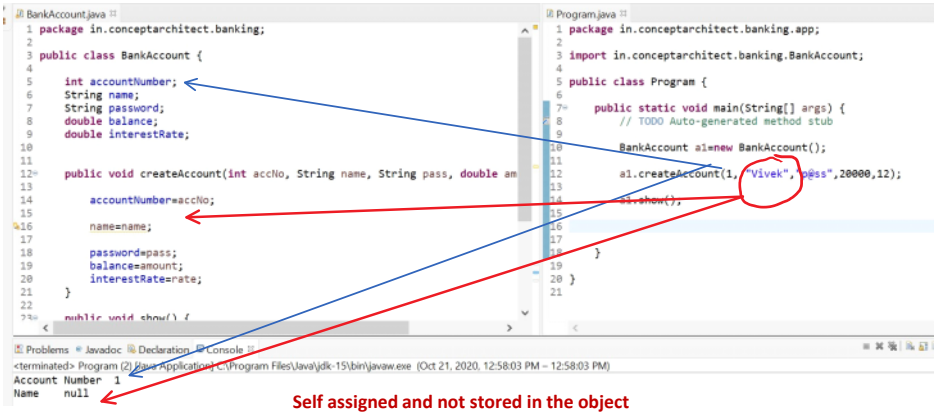
Wednesday, October 21, 2020 12:48 PM

```java
public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    double interestRate;


    void createAccount(int accNo, String name, String pass, double amount, double rate) {

        accountNumber=accNo;

        name=name;

        password=pass;
        balance=amount;
        interestRate=rate;
    }

}
```

What will this code do?

name=name;

- Whenever you have two different names in a given context a local (near) one is preferred over far one.
- This this case name refers to parameter that is passed which is closer to function that the class members.
- Here it would end up as a self assignment of function parameter.
  - The code has No effect.



Self assigned and not stored in the object

Solution

Option 1

- Don't use same name for parameter and the class fields (states)

Option 2 (PREFFERED)

- Use **this** keyword
  - "this" is a special keyword in an object method that represents the current object
  - So a "**this.name**" would always represent an object's property and not a method argument
  - You can always use "this" with any class field or method
    - We generally use it only to distinguish between method arugment and properties.

```java
public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    double interestRate;


    public void createAccount(int accountNumber, String name, String password, double amount, double rate) {

        this.accountNumber=accountNumber;
        this.name=name;  //this.name is class field, name is argument
        this.password=password;

        this.balance=amount; //this is not required as balance has no conflict
        interestRate=rate;   //this is not required
    }

    public void show() {
        System.out.println("Account Number\t"+this.accountNumber);//this is not required
```

If there is a name conflict explicit use of **this** is required.

When there is no name conflict

- **this** is optional and implicit.
- Even when you don't write this, you mean this as

```java
        interestRate=rate;   //this is not required
    }

    public void show() {
        System.out.println("Account Number\t"+this.accountNumber);//this is not required
        System.out.println("Name\t"+name); //this is not required
        System.out.println("Password\t"+password); //this is not required
        System.out.println("Balance\t"+balance); //this is not required
        System.out.println("Interest Rate\t"+interestRate); //this is not required
    }

    public void deposit() {
        //TODO: write the deposit logic
    }

}
```

When there is no name conflict

- **this** is optional and implicit.
- Even when you don't write this, you mean this as implicit.
- You can write explicit this
- You can write without this

Value Type Vs Reference

Wednesday, October 21, 2020 1:13 PM

In Java Script a variable can be of two types

Value Types

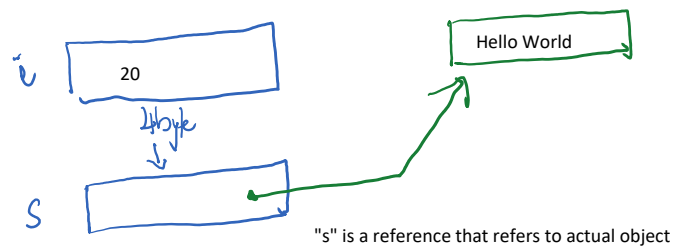
- A primitive data type is called value types.
- They are stored directly in the variable memory
- These include built-in data types such as
  - int
  - float
  - double
  - char
  - byte
  - Boolean

Reference Type

- It represents all complex data type which are created as class
- This also includes all user defined data types
- Variable stores the reference (address) of the actual object that would store value

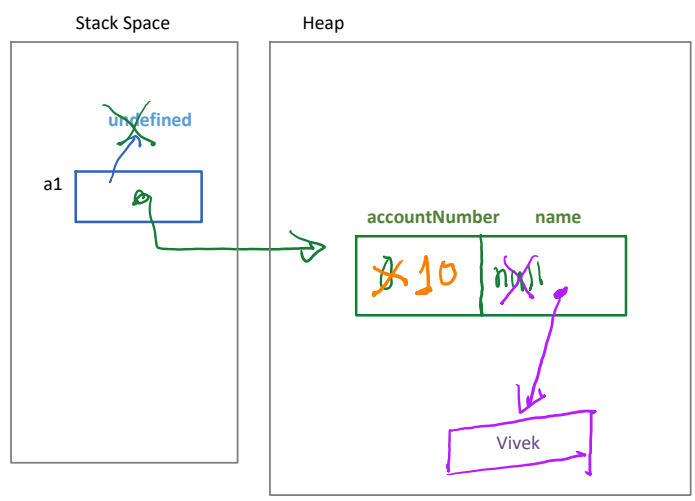
Example 1 -- int is value type, String is reference type

```
public static void main(String [] args){

     int i=20;

     String s="Hello World";

}
```



Example 2 — User defined Object

```
class BankAccount{

    int accountNumber;
    String name;
}

void main(){

    BankAccount a1; //draw memory model here

    a1= new BankAccount(); //draw memory model here

    a1.accountNumber=10;  //draw memory model here

    a1.name="Vivek";  //draw memory model here

}
```



Assignment 5 -- Draw the Memory Snapshot for Following code

Wednesday, October 21, 2020 1:23 PM

```
class BankAccount{

    int accountNumber;
    String name;
}

void main(){

    BankAccount a1; //draw memory snapshot 1

    a1= new BankAccount(); //draw memory model snapshot2

    a1.accountNumber=10;  //draw memory snapshot 3

    a1.name="Vivek";  //draw memory snapshot 4

}
```

- You can create a single diagram and use 4 different color to define the 4 stage of memory allocation
- You may create 5 different diagrams

When is Object Created?

Wednesday, October 21, 2020 1:13 PM

```
BankAccount.java
1  package in.conceptarchitect.banking;
2
3  public class BankAccount {
4
5      int accountNumber;
6      String name;
7      String password;
8      double balance;
9      double interestRate;
10
11
12     public void createAccount(int accNo, String name, String pass, double am
13
14         accountNumber=accNo;
15
16         name=name;
17
18         password=pass;
19         balance=amount;
20         interestRate=rate;
21     }
22
23     public void show() {
```

```
Program.java
1  package in.conceptarchitect.banking.app;
2
3  import in.conceptarchitect.banking.BankAccount;
4
5  public class Program {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9
10         BankAccount a1=new BankAccount();
11
12         a1.createAccount(1, "Vivek","p@ss",20000,12);
13
14         a1.show();
15
16
17
18     }
19
20 }
21
```

```
Problems  Javadoc  Declaration  Console
<terminated> Program (2) [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Oct 21, 2020, 12:58:03 PM – 12:58:03 PM)
Account Number  1
Name    null
```

Where is Bank Account Object Created?

- On Line 10? Or
- On Line 12?
- Here

If object is created on this line — BankAccount a1=new BankAccount();

- What is the account number of this account?
- Whose account this?
- What is someone tries to deposit some amount on line number 11 (before calling createAccount)?

If object is created on this line — a1.createAccount();

- What is the status of account on line 11 — does account exist or no?

Considerations

- **Line 10**
  - This is in memory creation (memory allocation for the object.
  - For **Java** object exists on this line
  - But for **business/domain** the object isn't created here. It is not a valid BankAccount
- **Line 12**
  - For business Object actually gets created here.
  - But object can be misused before this line is called.
  - Programmer may forget to call this line.

Problem

- There are conceptually two objects
  - Java object
  - Business object
- They are created at different point of time.
- The difference in the creation is not desirable.

Solution

- Merge the two creations by creating a **Constructor**

Constructor

- A special class method
- Responsible for Object creation and initialization.
- Same name as that of class
- No Return type — not even void.
- Every class always has a constructor
  - If you don't create your own you get a default 0 argument nothing doing constructor
    - This constructor is removed once you provide your own constructor

Working With Default Constructor

```
BankAccount.java
1  package in.conceptarchitect.banking;
2
3  public class BankAccount {
4
5      int accountNumber;
6      String name;
7      String password;
8      double balance;
9      double interestRate;
10
11
12     public void createAccount(int accNo, String name, String pass, double am
13
14         accountNumber=accNo;
15
16         name=name;
17
18         password=pass;
19         balance=amount;
20         interestRate=rate;
21     }
22
23     public void show() {
```

```
Program.java
1  package in.conceptarchitect.banking.app;
2
3  import in.conceptarchitect.banking.BankAccount;
4
5  public class Program {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9
10         BankAccount a1=new BankAccount();
11
12         a1.createAccount(1, "Vivek","p@ss",20000,12);
13
14         a1.show();
15
16
17
18     }
19
20 }
21
```

```
Problems  Javadoc  Declaration  Console
<terminated> Program (2) [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Oct 21, 2020, 12:58:03 PM – 12:58:03 PM)
Account Number  1
Name    null
```

- Call to default constructor of BankAccount which is provided automatically by Java
- Note we haven't created our constructor.

Default Constructor is removed if we provide our own constructor

```java
*BankAccount.java
public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    double interestRate;

    public  BankAccount(int accountNumber, String name, String password, double amount, double rate) {
        createAccount(accountNumber, name, password,amount, rate);
    }

    public void createAccount(int accountNumber, String name, String password, double amount, double rate) {

        this.accountNumber=accountNumber;
        this.name=name;  //this.name is class field, name is argument
        this.password=password;
```

Once we define our own constructor

```java
Program.java

import in.conceptarchitect.banking.BankAccount;

public class Program {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        BankAccount a1=new BankAccount();

        a1.createAccount(1, "Vivek","p@ss",20000,12);
```

The Default constructor is removed

Right way to call the constructor

```java
ankAccount.java
public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    double interestRate;

    public  BankAccount(int accountNumber, String name, String password, double amount, double rate) {
        createAccount(accountNumber, name, password,amount, rate);
    }

    public void createAccount(int accountNumber, String name, String password, double amount, double rate) {

        this.accountNumber=accountNumber;
        this.name=name;  //this.name is class field, name is argument
        this.password=password;
```

```
Console
<terminated> Program (2) [Java Application] C:\Program Files\Java
Account Number  1
Name    Vivek
Password        p@ss
Balance 20000.0
Interest Rate   12.0
```

```java
rogram.java

import in.conceptarchitect.banking.BankAccount;

public class Program {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        BankAccount a1=new BankAccount(1, "Vivek","p@ss",20000,12);

        //a1.createAccount(1, "Vivek","p@ss",20000,12); //no need to call this line

        a1.show();
```

```java
ankAccount.java
public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    double interestRate;

    public  BankAccount(int accountNumber, String name, String password, double amount, double rate) {
        createAccount(accountNumber, name, password,amount, rate);
    }

    public void createAccount(int accountNumber, String name, String password, double amount, double rate) {

        this.accountNumber=accountNumber;
        this.name=name;  //this.name is class field, name is argument
        this.password=password;
```

```
Console
<terminated> Program (2) [Java Application] C:\Program Files\Java
Account Number  1
Name    Vivek
Password        p@ss
Balance 20000.0
Interest Rate   12.0
```

```java
rogram.java

import in.conceptarchitect.banking.BankAccount;

public class Program {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        BankAccount a1=new BankAccount(1, "Vivek","p@ss",20000,12);

        //a1.createAccount(1, "Vivek","p@ss",20000,12); //no need to call this line

        a1.show();
```

Constructor vs createAccount

- What to use?
- If we see our code, my constructor is calling createAccount.
- Why should I create constructor if I already have a createAccount()

Constructor Advantages

1. Constructor merges Programming creation with domain creation
2. Both constructor and create logic are executed at the same time.
3. Constructor is called only once for an object at the beginning.
4. Problem with creator
    a. You may forget to call the method
        i. Object will be invalid till you call it
    b. You may call create method more than once for the same object
        i. This is not desirable.

Object Modelling

Wednesday, October 21, 2020 3:40 PM

BankAccount.java | Input.java

```java
            System.out.println("Invalid Amount. Deposit Failed");
        }

    }

    public void withdraw() {
        // TODO Auto-generated method stub
        Input input=new Input();

        int amount=input.readInt("amount to withdraw? ");
        String password=input.readString("password");

        if(!this.password.equals(password)) {
            System.out.println("invalid credentials");
            return ;
        }else if(amount<=0) {

            System.out.println("invalid denomination. please enter positive

        } else if(amount> balance) {

            System.out.println("Insufficient Funds");

        }else {

            balance-=amount;
            System.out.println("please collect your cash");

        }

    }
```

Program.java

```java
        //a1.createAccount(1, "Vivek","p@ss",20000,12); //no need to call th

        int choice=0;
        Input input=new Input();
        do {

            choice=input.readInt("1. Deposit\n2. Withdraw\n3.Credit Interst\

            switch(choice) {
            case 1:
                a1.deposit();
                break;
            case 2:
                a1.withdraw();
                break;

            case 3:
                a1.creditInterest();
                break;

            case 4:
                a1.show();
                break;
            case 0:
                break;
            default:
                System.out.println("Invalid Input.Retry");

            }
```

- There is Zero Interaction between Client (Program.java) and Service (BankAccount.java)

Problems

- Client has no way to know if withdraw was successful or it failed?
- **Who is responsible for interacting with User?**

Real world Banking Scenario

1. How will a user withdraw the money from his account?
    - ATM
    - Mobile App
    - Web App (paying a bill is also withdrawing from account)

2. Where is BankAccount object stored?
    a. Is **BankAccount** object stored in ATM memory?
        i. No.
    b. It is stored on the central Bank server on a remote location (Cloud)

ATM

BankAccount a1

BankServer

3. Where will Input and Print Statement Execute — on ATM or on Server?
    a. On Server

Problem

- User has no access to the server
- User can reach server to type the input
- User can't see anything typed on the server console

Solution

- Don't include I/O logic (Input or print) in Business layer object
- Don't have any print statement in any not designed to display out
- BankAccount object should have BankAccount Business Logic But not User Interaction Logic
- User Interaction Logic should be present in Presentation layer object such as an ATM machine
- ATM machine should interact with Business objects behind the scene.

Assignment 06

Wednesday, October 21, 2020 3:59 PM

1. Add mechanism to get/set password from the BankAccount Object
2. Remove all the Input/Print from BankAccount object
3. Move all User Interaction to ATM Object main Function

How do I allow BankAccount Password accessed or Modified

```java
package in.conceptarchitect.banking;

import in.conceptarchitect.utils.Input;

public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    static double interestRate;



    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
```

- Anyone can call getPassword() and display it using System.out.println
- Not safe
- What is password is **salted**
    - Generally passwords are stored in an encrypted format in the database so that no one can know the true phrase of your password, including the admins

- Anyone can change your password without knowing your existing password.
- This makes password useless.

How should you model the password access/change mechanism

```java
public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    static double interestRate;



    //Its a dummy and non-secured logic for password hashing just to demonstrated
    //the idea. Search for password hashing algorithm for better logic
    private String salt(String password) {

    public boolean authenticate(String password) {
        return salt(password).equals(this.password);
    }

    public boolean changePassword(String oldPassword, String newPassword) {

        if(authenticate(oldPassword)) {
            password=salt(newPassword);
            return true;
        }else
            return false;

    }

```

Don't give them password. You ask them a password and confirm if it is the correct password.

Before you change the password, authenticate user with current password and then change

Assignment 07

Wednesday, October 21, 2020 4:00 PM

- Introduce ATM object to include User interaction logic
- ATM should take user Input and interact with BankAccount Object

Static Vs NonStatic in Java

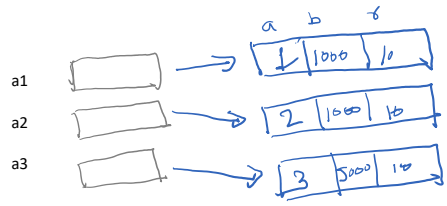
Wednesday, October 21, 2020 4:49 PM

Consider a BankAccount class

```
class BankAccount{
    int accountNumber;
    double balance;
    double rate;
}
```

What happens if I create 3 BankAccount Objects?

```
BankAccount a1=new BankAccount(1,1000,10);
BankAccount a2=new BankAccount(2,1000,10);
BankAccount a3=new BankAccount(3,5000,10);
```



If you have 10000 different BankAccount objects, is it probable that

- Some of them will have same balance?
  - YES
  - May be but a coincidence
  - Each object will have their own balance
- All of them will have same balance?
  - No
- One or more of them has same name?
  - YES
  - Same name may just be a coincidence
- Same account number
  - NEVER
- **Same rate of interest?**
  - **ALWAYS**
  - **For same type of account you would get same interest**

Problem!

- Why should I create 10000 copies of same value?

Static Field

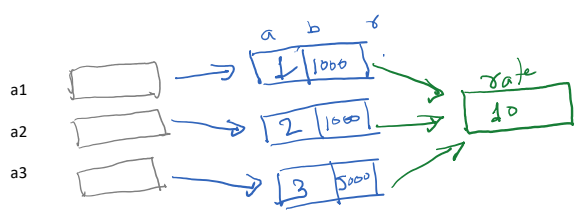
- A field that is marked static is a shared field in the class.
- There will be a single copy of that field irrespective of number of objects have
- We call them class level member
- Non static fields are called instance members or object level members
- **They are not initialized by constructor, rather by static block**

Note!

- Use static to create those states which are common and shared for all object of a class

```
class BankAccount{
    int accountNumber;
    double balance;
    Static double rate;
}

void main(){
    BankAccount a1=new BankAccount(1,1000,10);
    BankAccount a2=new BankAccount(2,1000,10);
    BankAccount a3=new BankAccount(3,5000,10);
}
```

A special copy each non-static or instance member is created per object call



Each instance will access a common copy of the static field or class level field

Non-static interest Rate

```java
package in.conceptarchitect.banking;

import in.conceptarchitect.utils.Input;

public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    double interestRate;


    public int getAccountNumber() {
        return accountNumber;
    }
```

```java
package in.conceptarchitect.banking.app;

import in.conceptarchitect.banking.BankAccount;
import in.conceptarchitect.utils.Input;

public class Program {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        //menuDrivenProgram();

        BankAccount a1=new BankAccount(1,"Vivek","p@ss", 10000,10);

        BankAccount a2= new BankAccount(1,"Vivek","p@ss", 10000,10);

        System.out.println("Interest rate for a1\t"+a1.getInterestRate());
        System.out.println("Interest rate for a2\t"+a2.getInterestRate());

        a1.setInterestRate(12);

        System.out.println("Interest rate for a1\t"+a1.getInterest Rate());
        System.out.println("Interest rate for a2\t"+a2.getInterestRate());



    }

    private static void menuDrivenProgram() {
        BankAccount a1=new BankAccount(1, "Vivek","p@ss",20000,12);
```

```
Console
<terminated> Program (2) [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Oct 21, 2020, 5:07:48 PM -
Interest rate for a1    10.0
Interest rate for a2    10.0
Interest rate for a1    12.0
Interest rate for a2    10.0
```

- Each Object maintains its own copy of the object state.
- Change is one doesn't change value for others.

```java
package in.conceptarchitect.banking;

import in.conceptarchitect.utils.Input;

public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    static double interestRate;


    public int getAccountNumber() {
        return accountNumber;
    }
```

```java
package in.conceptarchitect.banking.app;

import in.conceptarchitect.banking.BankAccount;
import in.conceptarchitect.utils.Input;

public class Program {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        //menuDrivenProgram();

        BankAccount a1=new BankAccount(1,"Vivek","p@ss", 10000,10);

        BankAccount a2= new BankAccount(1,"Vivek","p@ss", 10000,10);

        System.out.println("Interest rate for a1\t"+a1.getInterestRate());
        System.out.println("Interest rate for a2\t"+a2.getInterestRate());

        a1.setInterestRate(12);

        System.out.println("Interest rate for a1\t"+a1.getInterest Rate());
        System.out.println("Interest rate for a2\t"+a2.getInterestRate());



    }

    private static void menuDrivenProgram() {
        BankAccount a1=new BankAccount(1, "Vivek","p@ss",20000,12);
```

```
Console
<terminated> Program (2) [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Oct 21, 2020, 5:10:31 PM -
Interest rate for a1    10.0
Interest rate for a2    10.0
Interest rate for a1    12.0
Interest rate for a2    12.0
```

Changing Value for one changes for everyone

- What is this code doing?
  - Changing the interest rate for a1 or for everyone?

Problem

- It appears to be changing the rate for only a1.
- Actually its changing for everyone

Remember

- If you cant understand what a code is doing, you can manage code

Static Methods

- Just like a fields, a method can also be static.
- A static method belongs to class.
- A static method doesn't contain "**this**" reference
- A static method can't access any **non-static** field or methods
- Both static and non-static method have only one copy present in memory.
  - Static method doesn't help us save memory

S.No	Feature	Static Method	Non Static Method
1	**No of copies in memory**	1	1
2	**this reference**	NO	YES
3	**Accessing Static properties and methods**	YES	YES
4	**Accessing non-static properties and methods**	NO	YES
5	**Calling using object reference**	YES	YES
6	**Calling using class reference**	YES	NO
7	**Object required**	NO	YES

Why do I need static methods?

- You can call the method using Class Reference
- When a method is changing a value for entire class, it is logical that you call the method using class name rather than individual object name.

```java
    }

    //no set balance. to set balance use deposit or withdraw

    //get set interest rate
    public static double getInterestRate() {
        return interestRate;
    }

    public static void setInterestRate(double interestRate) {
        BankAccount.interestRate = interestRate;
    }
```

```
Console
<terminated> Program (2) [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Oct 21, 2020, 5:29:51 PM -
Interest rate for a1    10.0
Interest rate for a2    10.0
Interest rate for a1    12.0
Interest rate for a2    12.0
```

```java
        // TODO Auto-generated method stub

        //menuDrivenProgram();

        BankAccount a1=new BankAccount(1,"Vivek","p@ss", 10000,10);

        BankAccount a2= new BankAccount(1,"Vivek","p@ss", 10000,10);

        System.out.println("Interest rate for a1\t"+a1.getInterestRate());
        System.out.println("Interest rate for a2\t"+a2.getInterestRate());

        //it appears to change rate only for object "a1"
        // it is actually changing rate for everyone
        // Bad code! Unreadable
        a1.setInterestRate(12); //unclear code

        //Logically clear code
        BankAccount.setInterestRate(12); //Oh! its changing for everone

        System.out.println("Interest rate for a1\t"+a1.getInterestRate());
        System.out.println("Interest rate for a2\t"+a2.getInterestRate());



    }

    private static void menuDrivenProgram() {
        BankAccount a1=new BankAccount(1, "Vivek","p@ss",20000,12);
```

cls

This code is allowed in Java
And it defeats the purpose of static as a class level method can still be called using Object reference.

Recommendation!

- Avoid using object reference for static methods

Initializing Static Data

- **Don't initialize static data in constructor**
  - **Constructor is called for every object**
  - **Static data is a single shared copy**

Option 1

- Initialize by assigning a constant value wherever you declared static
- This should be used if the initial value is const

```
class BankAccount{

    static double interestRate=10;

}
```

Option 2

- We may need to get initial value from database or calculate them before use
- In such cases we should use static initalization block

```java
package in.conceptarchitect.banking;

import in.conceptarchitect.utils.Input;

public class BankAccount {

    int accountNumber;
    String name;
    String password;
    double balance;
    //static double interestRate= 10; //initializing static member if it is a constant value

    static double interestRate; //what is this value is pulled from a database and not a constant

    static {

        //write one-time initialization logic to initalize your static data
        //it is called before any constructor is called and as soon as class is loaded in the memory

        System.out.println("BankAccount intialized...");
        interestRate=10;
        accountNumber=5;  //can't initalize non-static fields

    }


    //Its a dummy and non-secured logic for password hashing just to demonstrated
    //The idea, search for password hashing algorithm for better logic
    private String salt(String password) {
        String s="";
        for(int i=0;i<password.length();i++) {
            char chp=password.charAt(i);
            int v=Character.getNumericValue(ch);

            s+= Integer.toHexString(v);
        }
```

- Static block is invoked exactly once when the class is loaded
- It is done before any constructor or other methods of the class is called
- It can have logic for static initialization
- **It can't access any non-static member of the class**

Static Field Use Cases

- Share Common Data Between Objects
- Let object connect with each other based on that shared information

```java
*BankAccount.java  Program.java
import in.conceptarchitect.utils.Input;

public class BankAccount {


    int accountNumber;
    String name;
    String password;
    double balance;

    static int accountCount=0;  //initalized with a constant

    static double interestRate; //what is this value is pulled from a database and not a constant

    static {

        //write one-time initialization logic to initalize your static data
        //it is called before any constructor is called and as soon as class is loaded in the memory

        System.out.println("BankAccount intialized...");
        interestRate=10;

        //accountNumber=5;  //can't initalize non-static fields

    }

    public  BankAccount( String name, String password, double amount) {

        this.accountNumber= ++accountCount; //uses shared field to auto increment account number
        this.name=name;
        this.password=salt(password); //we are saving a hashed/salted password and not the original one

        this.balance=amount;
        //interestRate=rate;   //static memebers are not initalized in constructor
    }
```

- Any change in Interest rate is done at one place
- Every object gets to use this value.
- They all share the interestRate like a constant for them
- Remember static is not constant. Its common or shared.

- Can be used coordinating between object
- We can have things like auto increment values which can help assign unique ids
- We can also have a common block where one object can put some information and other object can pull the information from that common block.
- Here object constructor increments the common value (not-constant)
- The next constructor uses the incremented value to further increment it

Static Method Usage

There are three popular usage of Static Methods

1. Define a generic function that is not connected any object
   a. These are independent functions similar to a procedural programming
   b. Examples:
      i. public static void main(String []args)
      ii. All Math formula present in Math class like **Math.pow()**

2. To access or modify static fields
   a. Since you have static fields, they need to be modified or accessed event without object
   b. To access or modify them it is better to create static methods
   c. Example:

Why is main() static?

- main() is the first function that is called in your application.
- It is called by the JVM
- At this stage there is no one to create an object of the class
- JVM simply calls **Program.main()**
  - It doesn't need to create an object of the Program class

Important!

- Static fields can be accessed even using non-static method
- The problem is
  - You need at least one object to call them
  - The operation may look like applied on one object rather than on all object

2. To access or modify static fields
   a. Since you have static fields, they need to be modified or accessed event without object
   b. To access or modify them it is better to create static methods
   c. Example:
      i. BankAccount.setInterestRate()
      ii. BankAccount.getTotalAccounts()

3. To coordinate the interaction between multiple objects
   a. What if I want to write a **transferFunds** method that transfers money from one account to another?
   b. We can do it using BankAccount.transfer(a1,a2,amount,password);

Important!

- Static fields can be accessed even using non-static method
- The problem is
  - You need at least one object to call them
  - The operation may look like applied on one object rather than on all object

```
public static boolean transfer(BankAccount source,  double amount, String password,BankAccount target) {

    if(source.withdraw(amount,password)) {
        target.balance+=amount;
        return true;
    } else {
        return false;
    }
```

Concerns

- Here we are accessing non-static members in static block
  - Non-static method
  - Non-static field
- But static methods can't access non-static members. Right? How is it working?

Remember!

- You can't access non-static member **directly**
- Here we are **not** accessing those fields or methods
- We are calling the object and its methods
  - source.withdraw()
  - target.balance+=amount
- I can't access withdraw() or balance directly
- Here since object is involved, its not a static context

Use Static or Not?

- Do we agree
  - interestRate is not owned by **bank account object**?
    - You own your name, password and balance
    - But you don't own or control interest rate
    - You certainly get the benefit of the interest rate
  - Bank account object can't know the **account count**?

Concerns

- If an object doesn't own a property (that is why it is static), why is it present in the class?
  - A static element is not owned by object, it is owned by the class
- Remember! A class is just the definition an object
  - If it is not present in the object, why is it present in the class?

BankAccount doesn't know or own account count. Who owns it? Who controls interest rate?

- Bank
  - A bank opens the accounts
  - A bank maintains a list opened accounts
  - A bank provides interest on the accounts
  - Interest rate and account count should be the property of a Bank object
  - Different bank can provide different rate of interests they choose

```
class BankAccount{

    static int accountCount;
    static double  intrestRate;

    int acountNumber;
    double balance;
    String name;
    String password;

    static double getInterestRate(){return interestRate;}
    static int getAccountCount(){ return accountCount;}

    public boolean deposit(double amount){
        …
    }

    public BankAccount(…){

    }
}
```

```
class Bank{

static int accountCount;
static double  intrestRate;


static double getInterestRate(){return interestRate;}
static int getAccountCount(){ return accountCount;}



}
```

Remember — Every static member of class X is actually part of some other object y of class Y

- In real world there is no real use case of static
- Static means no object or class level

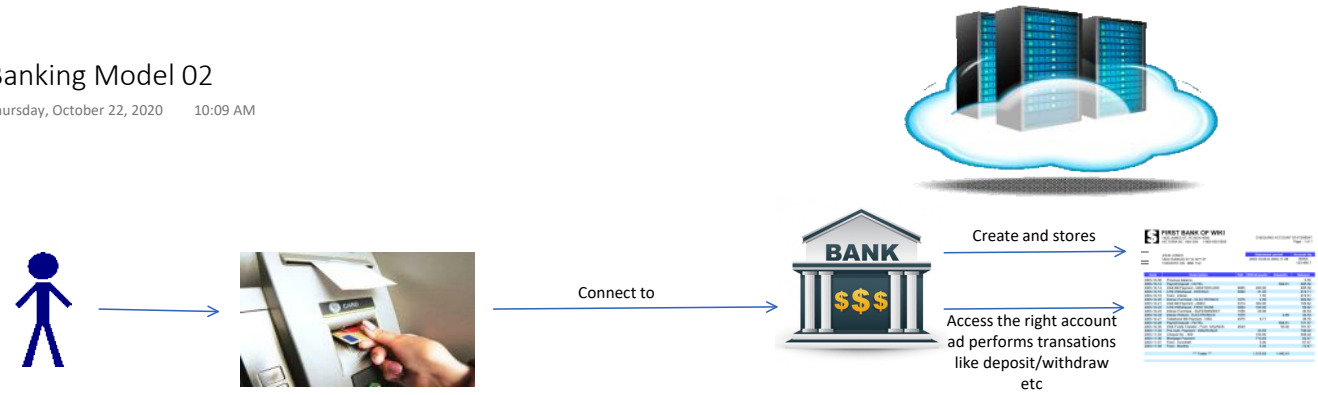
- But classes are not real. They are just notions (abstract ideas)

Should I create 'Bank' class only because I want to keep my interest rate and account count there.

- No.
- You need Bank
  - You can't create an bank account object yourself
    - You need to go to some bank to open your account
  - You can't withdraw money from your account directly
    - You need to go to bank to deposit/withdraw the money
  - Bank mains a list of all the accounts that it has opened.

Banking Model 02

Thursday, October 22, 2020 10:09 AM



User

ATM

1. Allows you to interact with your BankAccount
2. Takes User Input
3. Sends request to Bank
4. Gets Response from Bank
5. Dispense cash if required.
6. Operations
   a. Insert Your Card (Provide Account Number)
   b. Main Menu
      i. Deposit
         1) Enter Deposit Details
      ii. Withdraw
         1) Enter Amount
         2) Enter Pin
      iii. Balance Info

Bank

1. Creates (opens) an account
2. Stores the account for future
3. Allows you to transact with your account
   a. Deposit
   b. withdraw

BankAccount

1. Stores information about individual BankAccounts
2. Can be accessed only by the BankObject
3. Customer or ATM interacts with this object using BankObject

What is the role of main function here?

- It is to setup the ATM and BankAccount and Bank Object

```
class Bank{

int accountCount;
double  intrestRate;

BankAccount [] accounts;
double getInterestRate(){return interestRate;}
int getAccountCount(){ return accountCount;}


int openAccount(String name, String pass, int amount){
     int id=++accountCount;
     BankAccount newAccount=new BankAccount(id,name,pass,amount);

     //store a list of bank account in bank object
}

boolean withdraw( int accountNumber, double amount, String password){

     BankAccount acc= getAccountById(accountNumber);

     return acc.withdraw(amount,password);

}


}
```

```
class BankAccount{


     int acountNumber;
     double balance;
     String name;
     String password;

     public BankAccount(…){

     }


     public boolean withdraw(double amount, String password){
          …
     }

}
```

Assignment 08

Thursday, October 22, 2020 10:47 AM

- In our current BankAccount user is specifying the account Number
- This may conflict between to different account as they may choose same account number
- Create a design to auto assign and auto incremented account number to every bank account that is created
- Constructor shouldn't take account number from the user.

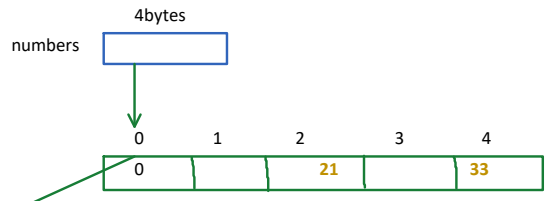
Array

Thursday, October 22, 2020 12:53 PM

- An array is a continuous non-expanding list of values that can be accessed using zero based index

Creating an array

```
int []  numbers;    //creates reference. No memory allocated
```



Allocating the memory

```
numbers= new int[5];  //allocates the memory
```

Accessing the array

numbers[2]=21;
numbers[4]=33;

System.out.println(numbers[2]); //21
System.out.println(numbers[1]); //0

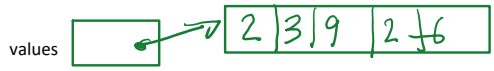
Trying to Access invalid Index throws exception

numbers[5]=29; //throws IndexOutOfBoundsException

numbers[-2]=29; //throws IndexOutOfBoundsException

System.out.println(numbers[21]); //throws IndexOutOfBoundsException

Initialzing Array with Fixed Values

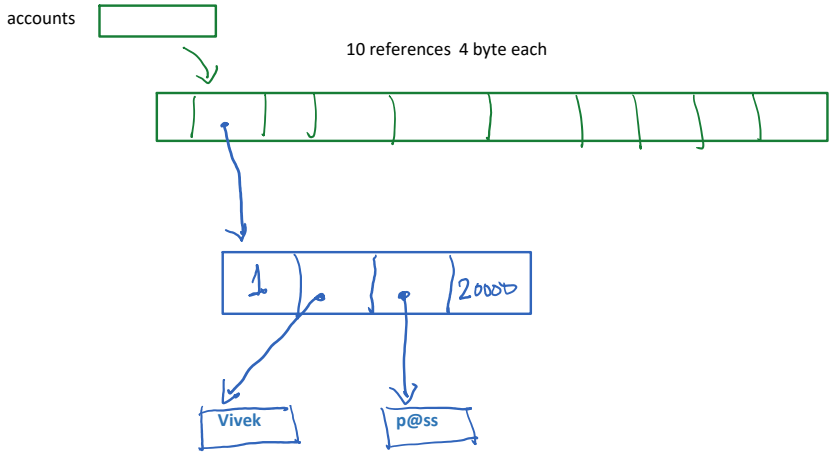



```
int[] values={2,3,9,2,6}
```

Array of Objects

- It uses the same style
- Creates an array of references
- Each reference refers to a different Object
- Objects will be stored in different memory

```
BankAccount accounts[] = new BankAccount[10];
```



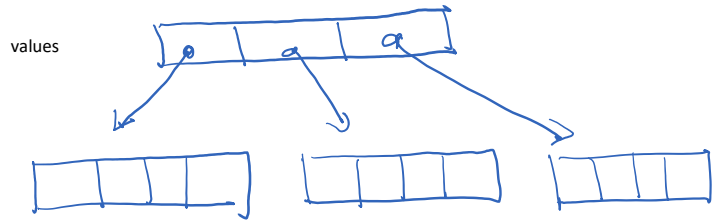
Creating Actual Objects

```
accounts[1]=new BankAccount(1,"Vivek","p@ss",20000);
```

Array of Array

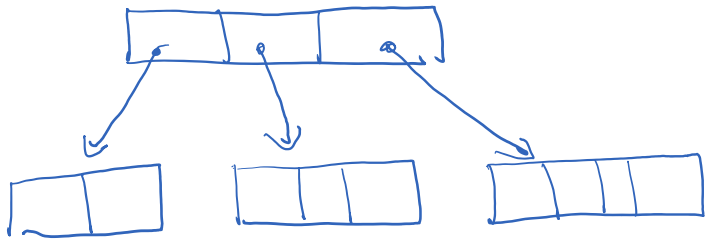
- An Array can have another array as member
- This makes multi dimentional Array

```
int [][] values= new int [3][4];
```

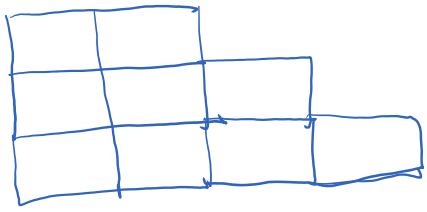


Non Rectangular array

```
int [] [] values= new int [3] [] ; //second is not given

for (int i=0;i<values.length;i++)
        value[i]= new int[i+2];
```



Conceptual View



Multi-dimensional array is rarely used in Object Oriented Programming

- We generally use array of objects rather than array or arrays

String

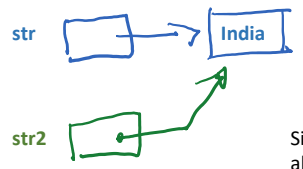
Thursday, October 22, 2020 1:14 PM

- Java Strings are immutable Objects
  - Once an object is created, it can't be modified in place
- Immutable design allows us to reuse the memory for a String object

Creating a string

String str="India";

String str2="India";



Since str2 is also the same immutable string, we will reuse it and not allocate a second memory

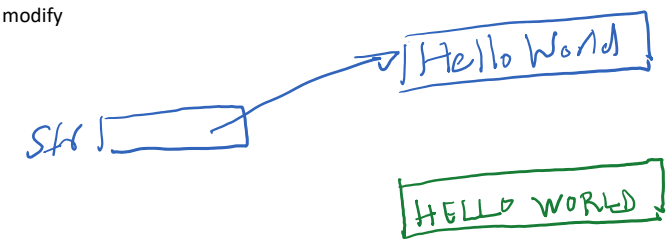
Important String methods

1	length()	Returns string length
	equals()	Compares if strings are equal (same)
	charAt()	Returns character at a 0 based index
	toUpperCase()	Converts String to upper case
	indexOf()	Index of another string into this string
	substr()	Returns a substring
	format()	Creates formatted strings with printf style syntax %s , %d etc

String is immutable

- Any method that tries to change the string actually creates a new string and doesn't modify existing one

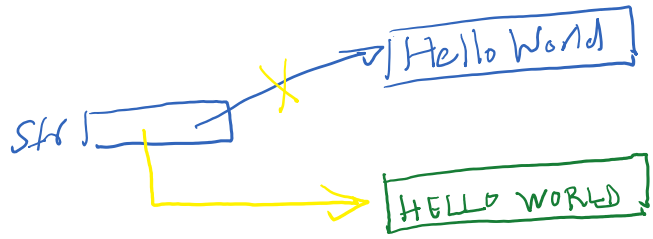
```
String str="Hello World";

str.toUpperCase();
```



Note the changes are in a new memory.
But no one refers this memory. So this change will not reflect any where and will be lost.

How to accept changes

```
String str="Hello World";

str=str.toUpperCase();
```



- Now str points to modified String

- The original string "Hello World" is no more referenced
    - It will be eventually garbage collected.

Assignment 09

Thursday, October 22, 2020 1:25 PM

- **Create the necessary classes to Implment Banking Project as Per the Object Model 2**
- **Complete the code in Bank class and ATM class**

Bank Account Types

Friday, October 23, 2020 9:43 AM

- In real world there are different type of BankAccounts
  - SavingsAccount
  - CurrentAccount
  - OverdraftAccount
  - LoanAccount
  - …
- All these accounts have their own set of rules and facilities
- Here is a simple matrix of how these accounts may differ

Type	Transactions Per Month	Max Withdraw Amoung	Credit Interest
Saving	50	balance-5000 If your balance is 20000 you can withdraw a max of 15000	Normal
Current	unlimited	Balance. If your balance is 20000, you can withdraw 20000	0
OverDraft	20	Balance+odLimit You may withdraw more than you balance*	Normal

Od Limit For Overdraft Account

- An Overdraft Account provides an Overdraft facilitiy
  - It means if you required you can withdraw more than your actual balance
  - You will have to pay a charge for it
  - How much you can withdraw is based on OdLimit
- An OdLimit is 10% of your historic max balance
- OdCharge is 1% of your Overdraft.

Example

- If altime highest balance in your account was 100000 (1 Lac)
  - Your ODLimit —> 10% of 100000 = 10000
- Now if you current balance is 20000
  - You can withdraw upto 20000+10000=30000
- Suppose you want to withdraw 25000
  - Your ovedraft will be 25000-20000=5000
  - The withdrawal will be permitted
  - You will have to pay an od charge
    - 1% of 5000=50
  - Your final balance will be : 20000 - 25000 - 50 = -5050

Assignment10

Friday, October 23, 2020 9:56 AM

- Create a different type of BankAccounts
- Implement the behaviors as per the provided matrix
  - Withdraw
  - Credit interest
- Add functionalities in Bank and ATM to
  - openSavingsAccount
  - openCurrentAccount
  - openOverdraftAccount

Bank Account Types (Implementations)

Friday, October 23, 2020 9:43 AM

- In real world there are different type of BankAccounts
  - SavingsAccount
  - CurrentAccount
  - OverdraftAccount
  - LoanAccount
  - …
- All these accounts have their own set of rules and facilities
- Here is a simple matrix of how these accounts may differ

Type	Transactions Per Month	Max Withdraw Amoung	Credit Interest
Saving	50	balance-5000 If your balance is 20000 you can withdraw a max of 15000	Normal
Current	unlimited	Balance. If your balance is 20000, you can withdraw 20000	0
OverDraft	20	Balance+odLimit You may withdraw more than you balance *	Normal

Od Limit For Overdraft Account

- An Overdraft Account provides an Overdraft facilitiy
  - It means if you required you can withdraw more than your actual balance
  - You will have to pay a charge for it
  - How much you can withdraw is based on OdLimit
- An OdLimit is 10% of your historic max balance
- OdCharge is 1% of your Overdraft.

Example

- If altime highest balance in your account was 100000 (1 Lac)
  - Your ODLimit —> 10% of 100000 = 10000
- Now if you current balance is 20000
  - You can withdraw upto 20000+10000=30000
- Suppose you want to withdraw 25000
  - Your ovedraft will be 25000-20000=5000
  - The withdrawal will be permitted
  - You will have to pay an od charge
    - 1% of 5000=50
  - Your final balance will be : 20000 - 25000 - 50 = -5050

Approach #1 -- Add a type variable in BankAccount

```
enum AccountType{ Savings,Current,OverDraft}

class BankAccount{

    AccountType type; //or may be int type may be supplied to constructor
    int minBalance=5000;  //used for SavingsAccount
    int odLimit;  //used for OverDraftAccount

 //other properties


    public boolean withdraw( double amount, String password){

        if(type==AccountType.Savings){
            if(amount< balance-minBalance){
                …
            }
        } else if(type==(AccountType.Overdraft){
            if( amount< balance+odLimit){
                …
            }
        }
    }

    public boolean creditInterest(double interestRate){
        if(type==AccountType.Savings){
            // provide interest
        } else if(type==(AccountType.Overdraft){
            //provide interest
        } else{
            //don't provide interest to CurrentAccount
        }

    }
}
```

Problems

1. How many AccountTypes should I specify?
   a. Currently we have 3.
   b. We may have more types in future
2. Every Account Type is adding similar code at multiple places
   a. When a new Account type is added we need to add similar block at all those places again
   b. Adding new account type may break my current programming logic.
      i. Does this else represent only **CurrentAccount** or all accounts that may be added in future?
3. Every account type may need some information that is not needed by other account types
   a. Example
      i. minBalance is needed only by SavingsAccount
      ii. odLimit is needed only by OverDraftAccount
   b. But each account, even if the don't need, will still have all those properties
      i. Even CurrentAccount will have minBalance and odLimit and it doesn't use either of them.
4. This is not an object oriented Approach
   - SavingsAccount, CurrentAccount and OverDraftAccount are different

```
        //don't provide interest to CurrentAccount
    }

  }
}
```

4. This is not an object oriented Approach

- SavingsAccount, CurrentAccount and OverDraftAccount are different types of Account just like Chair and Table are different type of Furnitures.
- They are not the same Object
- They shouldn't be part of the same Object
- Since they are different types, they don't belong to exact same class

Approach #2 Create Different Classes to represent different Account Types

```
class SavingsAccount{

    /* logic related to Savings Account */


}

class CurrentAccount{

    /* logic related to Current Account */

}


class OverdraftAccount{

    /* logic related to Overdraft Account */

}
```

Problem

1. There are many things common in All these Object types
   a. Properties
      i. accountNumber
      ii. Balance
      iii. Name
      iv. Password
   b. Behaviors
      i. Authenticate
      ii. Deposit
      iii. Show
2. We will need to write redundant codes
3. Here we can't say that all these are actually types of BankAccount
   a. Bank may need three arrays to store them
      i. SavingsAccount [] savingAccounts;
      ii. CurrentAccount [] currentAccounts
      iii. OverdraftAccount[] overdraftAccounts
   b. We need to search for the account in all the three arrays when a transaction request has come.

Inheritance

Friday, October 23, 2020 11:33 AM

Inheritance

- Object Oriented Programming offers the concept of Inheritance
- Inheritance is used for modeling classification-subclassification
- We may inherit any class X from any other class Y
  - But we must inherit to create a hierarchy by following the rule
- A class X can inherit (extend) class Y if we can say
  - X is a type of Y
  - Example for good inheritance
    - Parrot is a type of Bird
    - SavingsAccount is a type BankAccount
    - Chair is a type of Furniture
- Don't Inherit if there is **no is-a-type-of** relationship
  - **Has a Relationship**
    - Lunch includes Apple
    - Computer Has a HardDrive
    - Bank has BankAccounts
  - **Is similar to / Is Like A**
    - Crow is like a Parrot
    - SavingAccount is like a CurrentAccount
  - Is Associated
    - Employee is assoicated to his/her Manager
    - Computer and Printer works together
  - **Unclear Relationship**
    - They may appear to be is-a-type-of
    - Careful examination would reveal they are not really a type of
    - **Example**
      - A toy car is a type car
        - False
        - Toy car looks like Car
        - Toy Car is a type of Toy not a type of Car
        - It doesn't have real car features.

These are Valid Relationships, but not a candidate for Inheritance.

Don't Inherit,

- **unless you can say X is-a-type-of Y**
- **You see classficiation and subclassficiation.**

How do you Inherit in Java?

-