ANZ Java

Wednesday, February 1, 2023 10:59 PM

URL: 202302-anz-java https://1drv.ms/u/s!AknT1SrRpCz-wLEUhesLREpkzkkp4w?e=9QtUn0

http://tiny.cc/anz-java

GIT: https://github.com/vivekduttamishra/anz-java-202302

C++

Thursday, February 2, 2023 8:46 AM

C = C+1

X++

- C with class
- new C

C ++ --

Java

Thursday, February 2, 2023 8:54 AM

Java

- **Platform independent**
- **Architectural neutral**
- General purpose
- Object Oriented
- Multi-threaded
- Network
- **secured**
- robust
- high peromance
- **interpreted**

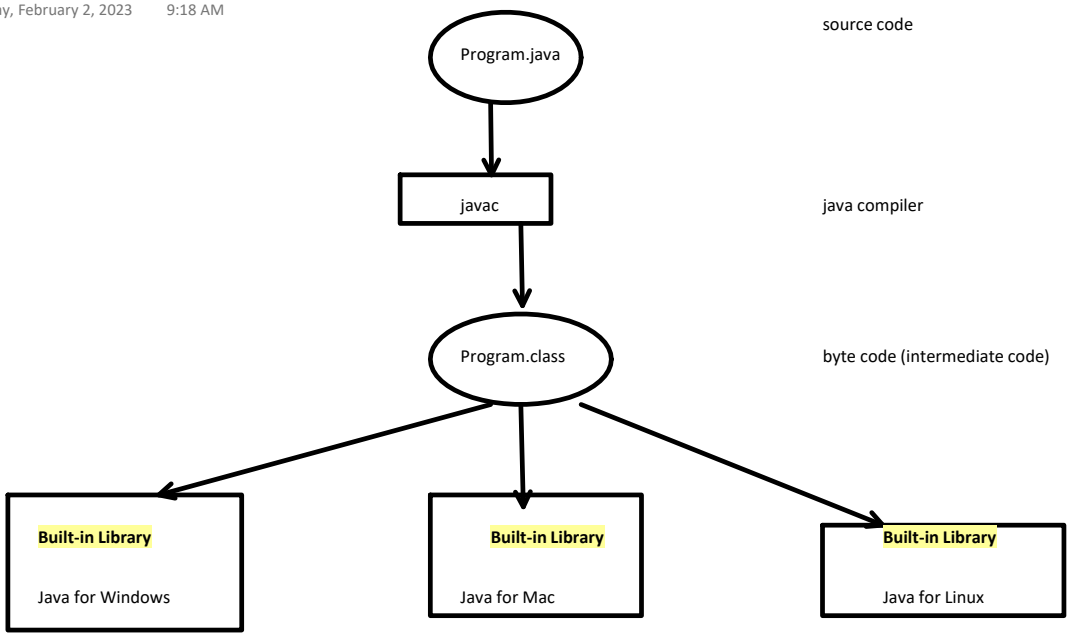
programming language.

Java Promise

- Write Once, Run Anywhere
- No separate code for different OS/Hardware combination.

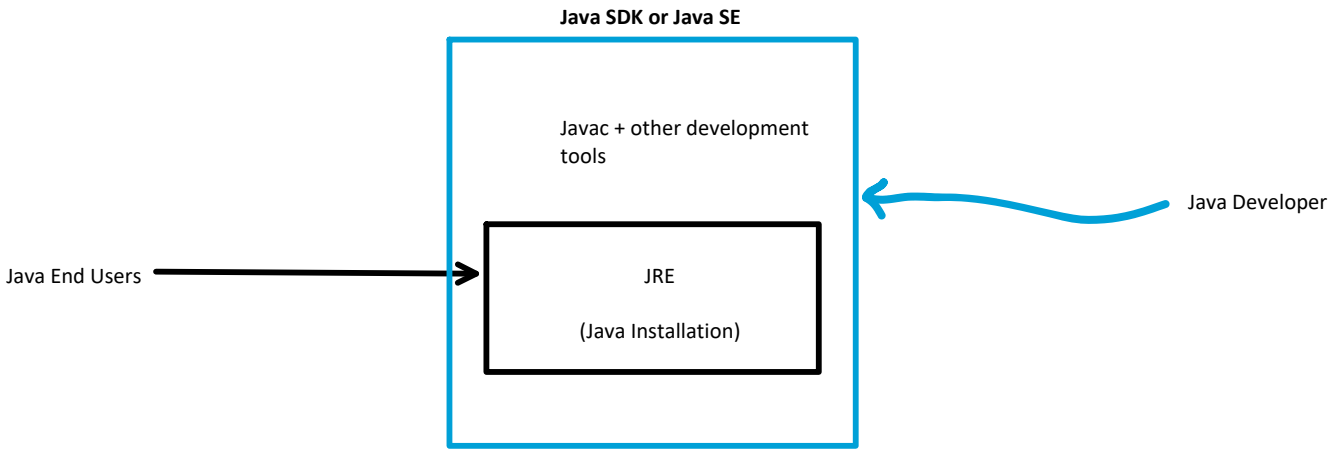
Java Program Flow

Thursday, February 2, 2023 9:18 AM



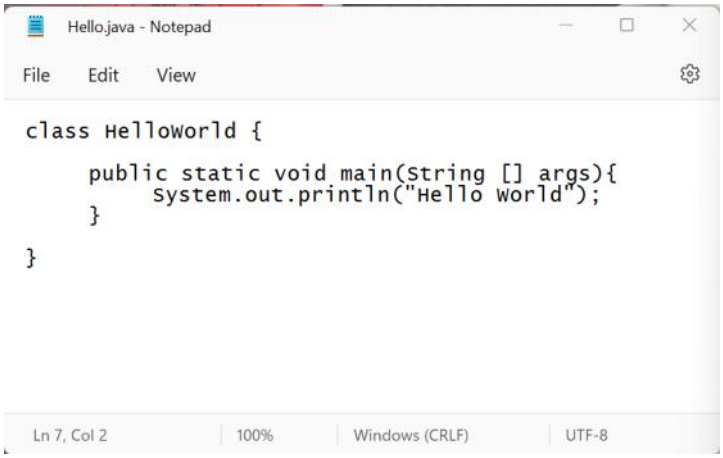
Installation and Bundles

Thursday, February 2, 2023 9:24 AM



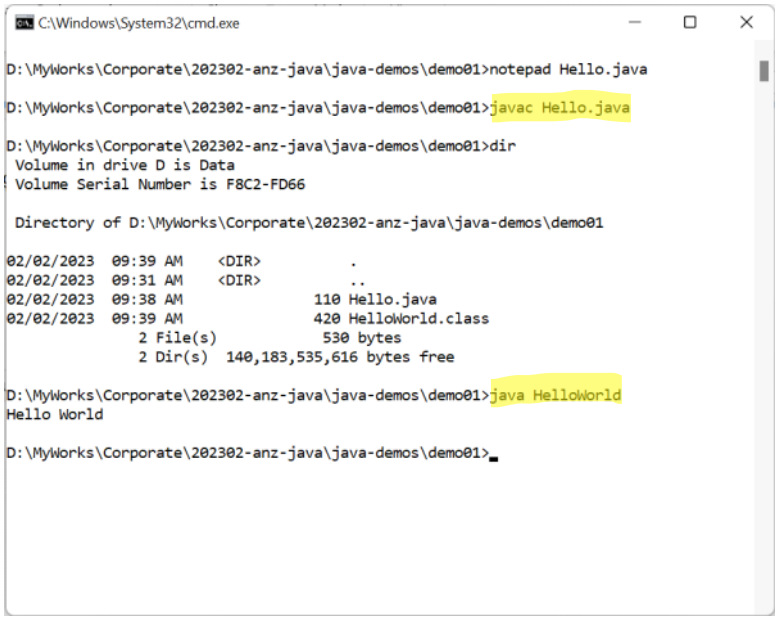
Hello World

Thursday, February 2, 2023 9:59 AM



- Write a Hello.java
- We need

1. A class
    - It can have any name we like
2. main function
    - match exact signature
3. print statement
    a. match exact singature



Step #1 compile

- we compile the source file.
- Here we use the full file name in the exact same case with extension

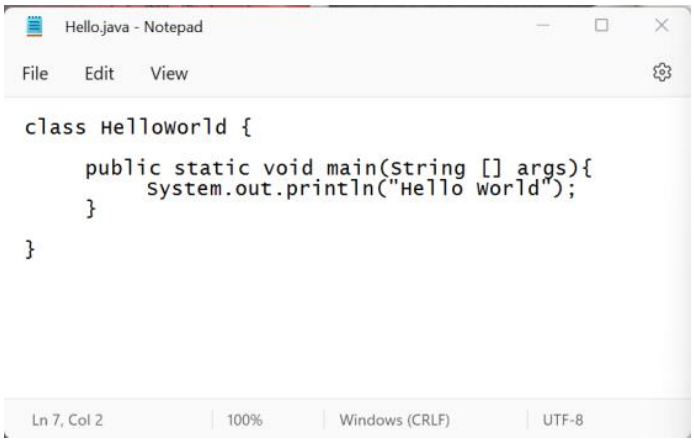
- On success we get
    - A class file with same name as that of class
    - It may not be same as the file name

Step #2 run the program

- we run the class file that contains main
- name is case sensetive without suffixing .class

Basic Java

Thursday, February 2, 2023 10:34 AM



Naming Convention in Java

- Class Name
  - Pascal convention
  - Name should begin with upper case
  - If the name is a composite name it each word should begin with upper case
  - No underscores
  - Example
    - class Hello
    - class InterestCalculator

- Method Name/ Field Name / Variable Name
  - Camel case
  - Name should begin with lower case
  - In case of composite word each subsequent word should begin with upper case
  - avoid underscore
  - example
    - calculate()
    - calculateInterest()
    - period
    - interestRate
- package name
  - all lower case

Anatomy of Java Program

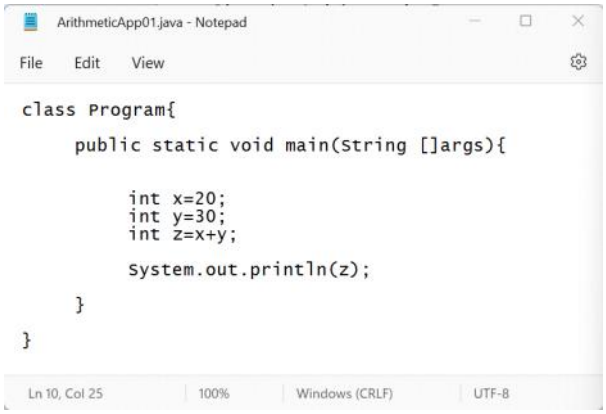
- A Java Program will have one or more classes
  - we need at least one class
- A class may have one or more methods (or functions)
  - Every program should have a "main" function
  - Every class doesn't need main.

- A Java Program is case sensetive.
  - You must be careful about the cases (upper case or lower case)
- Java Keywords
  - There are some special keywords that have special meaning in java
    - example
      - class
      - public
      - static
      - void
    - All keywords are in lower case
  - There are user defined words that represent
    - class name
    - method name
    - variable name
    - Example
      - HelloWorld
    - Few class names are pre defined by Java but are not keywords
      - String
      - System
      - out
      - println
    - main is special
      - It is created by user
      - Java expects you to create it
    - All user defined words can be in any case
      - You must use it in subsequent placed based on orginal defintion.
      - We follow certain naming convention to avoid confusion

Simple Arithmetic Program

Thursday, February 2, 2023 10:50 AM

- Write a program to calculate sum of two numbers



Variable

- To store a value of a particular type and refer it back we need to create a user defined name called variable
  - variable indicates that the value can change later.

```
int a= 20;  //a is an integer that has current value 20.

char b= '3'; // can hold international character set

double c=20.7;  //can hold non-integer values


boolean d= true;

boolean e= 7>8;  //false
```

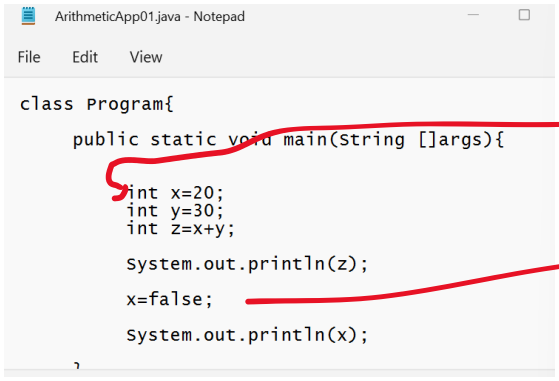
- a variables value can change later

```
a = 30;    //change the value to another value

a = a * 10;  //change the value based on the previous value of same
variabel
```

- You can't store wrong type of value in a variable

```
a="Hello World"; //can't store String in int variable
```




```
D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac ArithmeticApp01.java
ArithmeticApp01.java:12: error: incompatible types: boolean cannot be converted to int
                x=false;
                  ^
1 error
```

Data Types

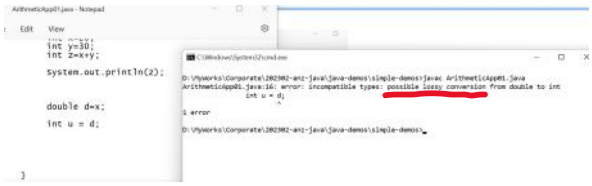
- to store the value in memory we need to create variables
  - variables are memory locations with specific name
  - they are associated with a particular type of value they can hold

- common types
  - int
    - integer
  - float
    - floating point (decimal numbers, single precession)
  - double
    - floating point decimal number, double precession
  - boolean
    - true/false
  - Other less used data types
    - char
      - a unicode char representation
      - represented as a single single quoted letter.
        - 'A'
        - '2'
          - '2' and 2 are different from each other
          - '2' doesn't possess arithematic quality
    - byte
      - represents a single byte
    - short
      - short int
    - long
      - long int

- **String**
  - String is a series of char to represent
    - word
    - sentense
  - It is double qutoed
  - Note String begins with upper case S
    - It is a class and not a keyword
    - It is a predefined class created by Java team

Compatible and Incompatible type

- Few types are compataible if not same
- an int can be assigned to double without any information loss
  - Java allows this conversion automatically
  - implicit type conversion
- a double may be assigned to int with a loss of information (fraction part)
  - They are compatible but lossy
  - Java doesn't allow this conversion automatically



- we can force such conversion by explicit type casting

```
int u = (int)  d;  //force convert value of 'd' in int before assign
```

- Note here 'd' remains double
- The value of d is converted to int and stored in u

Print A report including multiple variables

- wat if we want to say
  - sum of 20 and 30 is 50
- Java allows "+" operator between string and anything
  - String + anything => string

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

        int x=20;
        int y=30;
        int z=x+y;

        System.out.println("sum of "+ x +" and "+ y + " is "+ z);

    }
}
```

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac ArithmeticApp01.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>java Program
sum of 20 and 30 is 50

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>
```

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

          int x=20;
          int y=30;
          int z=x+y;

          String output="sum of "+ x +" and "+ y + " is "+ x + y;

          System.out.println(output);


          System.out.println( "sum of "+ x +" and "+ y + " is "+ (x + y));

     }
}
```

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac ArithmeticApp01.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>java Program
sum of 20 and 30 is 2030
sum of 20 and 30 is 50

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>
```

Statement vs Expression

- An expression can be a
  - simple value
  - arithmetic expression containing variable, constant and operators
- An statement
  - Always ends with a semicolon
  - A statement may be
    - declaring a variable
      - int x=20;
    - calling a method
      - System.out.println(x)
  - A method can take an expression as a parameter
    - It can't take statement as a parameter
    - We can't declare a variable as a method argument

~~System.out.println(int x=20);~~ //Not allowed.

```
System.out.println( x*20); //allowed
```

White space

- Java considers blank space, tab and enter key or their combination as white space
- Whereever we can have a blank space or an operator, we can add any combination of white space
  - A statement may have multiple blank space, tab or even enter key
  - A statement or a expression may span to multiple lines
  - end is marked with semicolon

- valid statements may look like

```
int a=20;  int b= a

+

30

/2 ;
```

- Note
  - statement 2 (declaration of variable b) begins in same line where first statement ends
  - second statement spans in 4 lines
  - It is acceptable

- Exception to this rule
  - A string doesn't follow white space concept
  - A string must end in the same physical line
  - Invalid statement

```
String address = "A2 202, Ozone Evergreens,
            Haralur Road,
            Bangalore
            560102 "
```

- To represent string with multiple line we use special combination characters to represent single character. This is known as escape sequences
  - \n --> new line (also includes \r)
  - \r —> carriage return
  - \t —> tab
  - \b —> back space
  - \' —> '
  - \" —> "
  - \\ —> \

- To represent the above address properly

```
String address = "A2 202, Ozone Evergreens,\nHaralur Road,\nBangalore\npin\t560102";
```

- To represent a large string in source code we can use string concat

```
address=  "A2 202,\n"+
          "Ozone Evergreens,\n"+
          "Haralur Road,\n"+
          "Bangalore,\n"+
          "pin\t560102";
```

Java Operators

Thursday, February 2, 2023 12:01 PM

Operators	Meaning	Associative
()		inner to outer (right to left)
.		left to right
* , / , %, ~, !		left to right
+, -		left to right
<,>,<=,>=, ==, !=	Relational	left to right
&&	Boolean and	left to right
||	Boolean or	left to right
= += -= *=	assignment	right to left
?:		

Increment and Decrement (Prefix and Post fix)

- when increment/decrement is a independent expression they are exactly same
  - x++;
  - ++x
- when increment/decrement comes as part of another expression
  - prefix is resolved before resolving the expression
  - postfix is resolved after resolving the expression

```
int x=20;

x++; //21
++x; //22
```

```
int y=5;

int z = y++ * 10;  //z will be 5* 10 =50,  y will become 6 later


int k=5;

int l = ++k * 10 ; // first k will become 6 then l with become 6*10 = 60
```

Integer Operations

```
int x=50;
int y=4;
```

- an operation between 2 int always returns an int
  - It truncates (not rounds) to fractional part
- An operation involving at least one double makes the result double

- 20 * 30 /40
  - 20* 30 = 600
  - 600 / 40 = 15
- 20 + 40 * 4
  - 40 *4 = 160
  - 20+160 = 180
- (20 + 40) * 4
  - 20+40 = 60
  - 60 *4 = 240

Composite Assignment

- x+=y
  - x= x+y
- x*=y
  - x=x*y
- x=x+1
  - x+=1
  - x++
  - ++x
- x=x-1
  - x-=1
  - x--
  - --x

```
int x=50;
int y=4;

double z= x/ y;

System.out.println(z); //12.5
```

- an operation between 2 int always returns an int
  - It truncates (not rounds) to fractional part
- An operation involving at least one double makes the result double

- z = x / y;
  - x/y => 50/4 = 12~~.5~~ =12
- z= 12
  - z=12.0

How to get 12.5?

- At least one operand should be double (or casted to double)

- Option#1

```
z= (double)x /y;
```

- Option#2

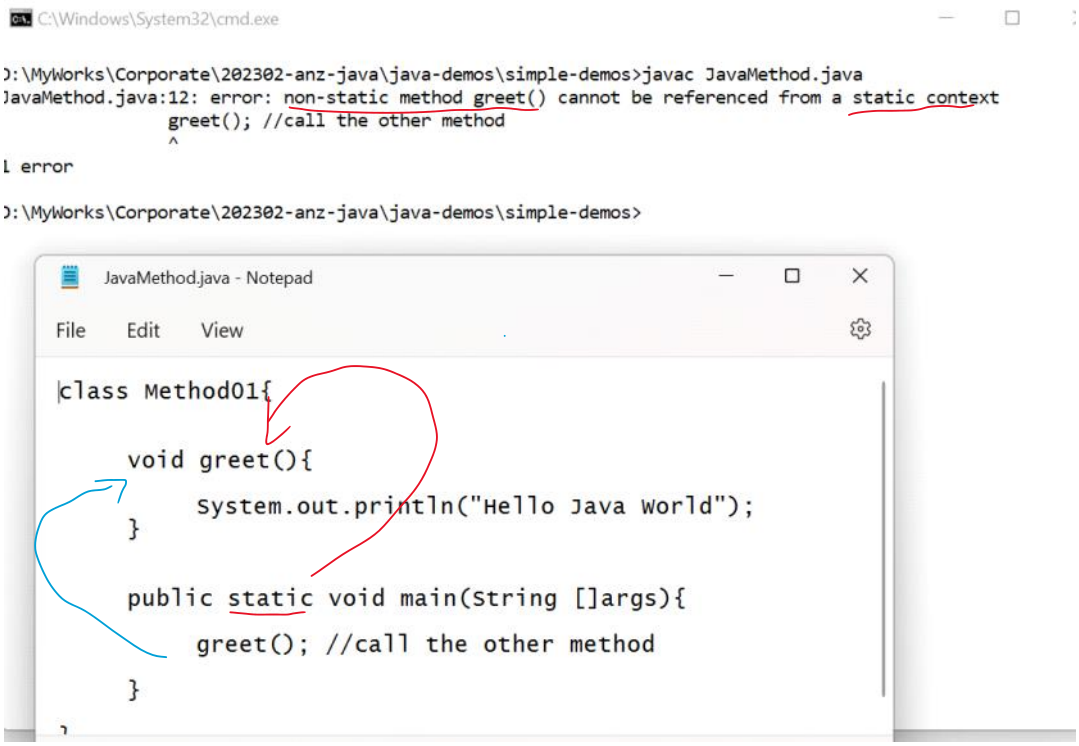
```
z= x*1.0/y;
```

Java Methods (functions)

Thursday, February 2, 2023 12:45 PM

- methods are independent reusable algorithm
- They have
  - name
  - return type (that can also be void if we don't return anything)
  - can take one or more parameters

A Method may represent a piece of exeuctable code



- A static main, can't call non-static greet
- main must be static
- We may also mark greet as static
- Why?
  - We will discuss later!

Working with multiple Methods

```
File  Edit  View

class Method01{

     static void greet(){

          System.out.println("Hello Java World");
     }

     public static void main(String []args){

          greet(); //call the other method
          greet(); // call again
          greet(); //and yet again

     }
}
```

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac JavaMethod.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>java Method01
Hello Java World

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac JavaMethod.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>java Method01
Hello Java World
Hello Java World
Hello Java World

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>
```

Note:

- We have two methods in our program
  - greet
  - main
- Eventhough "greet" is the first method, program always begins with "main"
- "greet" will not work unless it is called explicitly
  - if main never calls it, it doesn't work. just exists
- main may call greet multiple times
- there can be more methods forming a chain
  - main calls method1
  - method1 calls method2
  - ...

What if we need to greet someone specific?

- we may pass the name of the person to be greeted as a prameter
- A parameter is like a varible that is created and assigned the value passed.

```
File  Edit  View

class Method01{

     static void greet(String name){

          System.out.println("Hello "+name+", welcome to Java World");
     }

     public static void main(String []args){

          greet(); //call the other method
          greet(); // call again
          greet(); //and yet again

     }
}
```

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac JavaMethod.java
JavaMethod.java:12: error: method greet in class Method01 cannot be applied to given types;
               greet(); //call the other method
               ^
  required: String
  found:    no arguments
  reason: actual and formal argument lists differ in length
JavaMethod.java:13: error: method greet in class Method01 cannot be applied to given types;
               greet(); // call again
               ^
```

Note

- Here greet expects user to pass a String value
- That string value will be stored in a variable called name.
- greet method may use the name in their code

- But main is not passing the value for name and that is an error here
- Error
  - I couldn't find greet that doesn't take parameter

```
class Method01{

     static void greet(String name){

          System.out.println("Hello "+name+", welcome to Java World");
     }

     public static void main(String []args){

          greet("Vivek"); //call the other method
          greet("Raheem"); // call again
          greet("Venu"); //and yet again

     }
}
```

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac JavaMethod.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>java Method01
Hello Vivek, welcome to Java World
Hello Raheem, welcome to Java World
Hello Venu, welcome to Java World

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>
```

- Here we are calling greet multiple times with different values for name
- The supplied values are called arguments
- variable that is created to store argument is known as parameter
- Example
  - parameter is "name"
  - arguments supplied for "name" are
    - "vivek"
    - "rahim"
    - "venu"

Method chaining

```
File    Edit    View
class Method01{

     static void greet(String name){

          System.out.println("Hello "+name+", welcome to Java World");
     }

     static void goodBye(){
          System.out.println("Good Bye everyone, see you soon");
     }

     public static void main(String []args){

          greetEveryone();

     }

     static void greetEveryone(){

          greet("Vivek"); //call the other method
          greet("Raheem"); // call again
          greet("Venu"); //and yet again
     }
}
```

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac JavaMethod.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>java Method01
Hello Vivek, welcome to Java World
Hello Raheem, welcome to Java World
Hello Venu, welcome to Java World

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>
```

1. Program always begins with main()
2. main calls greetEveryone()
3. greetEveryone() calls greet() thrice with different parameters
4. no one calls goodBye() in the call chain that started with main
    a. It never executes
    b. It will not give any compile time error for being unused.
5. Physical order of method definition has no impact.

Method returning result

```
class Program{



     public static void main(String []args){

          int a= sumSquare(5,3); //
          System.out.println(a);

          int b= sumSquare(4,6);
          System.out.println(b);

     }

     static int sumSquare(int a, int b){

          int c= a+b;

          return c*c;
     }
}
```

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac JavaMethod02.java
JavaMethod02.java:17: error: invalid method declaration; return type required
       static sumSquare(int a, int b){
              ^
1 error

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac JavaMethod02.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>java Program
64
100

D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>
```

Note

- sumSquare indicates that it is returning a value of type int.
- Before the function ends it must include a return statement with value that we need to return
- The returned may be used in the caller function as expression
    - assigned to a variable
    - included in a forumla
        - int c=sumSquare(5,5)*10;
    - print directly
        - System.out.println(sumSquare(2,3));

- each method has it's own set of variables
- main has
    - a
    - b
- sumSquare ahs
    - a
    - b
    - c
- It is possible that two different method has variables with the same name
    - They belong to different method and are unrelated
    - same name is just a co-incidence

Statements

Thursday, February 2, 2023 2:27 PM

- every statement ends with a semicolon
- a block of statement is wrapped in braces {}
- Java statements like if, while, for etc can take either a single statement or a block of statements

If statement

```
if ( boolean_expression){

     statement1;
     statement2;
}


if(boolean_expression)
     single_statement;
```

Important

- block marker is always required in class and method even if they contain single statement
- for loops and branching braces are optional if there is single statement

If - else

```
if( boolean_expression)
     statement_or_block;
else
     statement_or_block
```

if -else if

```
if( condition1 )
     do_this;

else   if(condition2)
     do_this
else if (condition 3)
     do_this;
else
     do_this;
```

while loop

```
while( condition_is_true)
     block_or_statement;
```

do-while

```
do{
     one_or_more_statement;
}while( condition_is_true);
```

Note

- do-while needs block marker even for a single statement

- do while executes a min of one time before it tests for condition

standard for loop

- similar to c/c++ etc

```
for( initalization;  condtion;  reinitialization)
     block or statement;
```

- for executes in othe order
    1. runs intialization
    2. checks condition
    3. runs block or statement if conditon is true else exists
    4. runs reinitalization
    5. repeats from step 2

```
for(int i=0; i<10; i++)
    greet();
```

- Note
    - Intialization can declare a new variable here.
    - All the three components are optional in for loop
        - You may or may not provide
            - intialization
                - if it is already done before for loop
            - condition
                - defaults to true
                - if not given it is like run for ever
            - re-inialization
                - if you are doing within the block
        - But the two semicolon inside for () is compulsary

- example for a run for ever for loop

```
for(;;)
    run_for_ever;
```

Examples

```
void countDown(int x){

    for(;x>0;x--){
        System.out.println(x);
    }

}
```

Example 2

```
void countDown(int x){

    for(;x>0;){
        System.out.println(x--);
    }

}
```

How to exit a loop without finishing

- sometimes we need to exit a loop (for/while/do-while) before its natural condition
    - we may have more than one condition and it may be complicated to put all in one place
- we can use "break" statement to exit the loop

- for example break the loop after you get three values that are divisible by 5 while counting in a range
- **IMPORTANT!**
    - **NEVER USE BREAK INSIDE A LOOP WITHOUT A CONDITION**

```
void countRange(int min, int max){


}
```

Skipping a loop count

- sometimes we may want to skip ramaining statements of a loop under a given condition.
- We may use continue to denote that.
- continue should be conditional
- Let's skip every multiple of 2

for-each style loop

- will discuss later.

switch statement

semantic

```
switch(expression){

    case value_1:
        statement1;
        statement2;
        break;
    case value_2:
        statement1;
        statement2;
        return;

    default:
        statement;


}
```

- A switch can take a expression that can be
  - number
  - string
- the value is matched to each case and and statement under the case is executed
- if no value matches the passed value it goes to default

- we should end each case with break or return
  - return exists the function
- we may use continue in switch case if it is present in some loop.
  - continue continues loop not switch

Important!

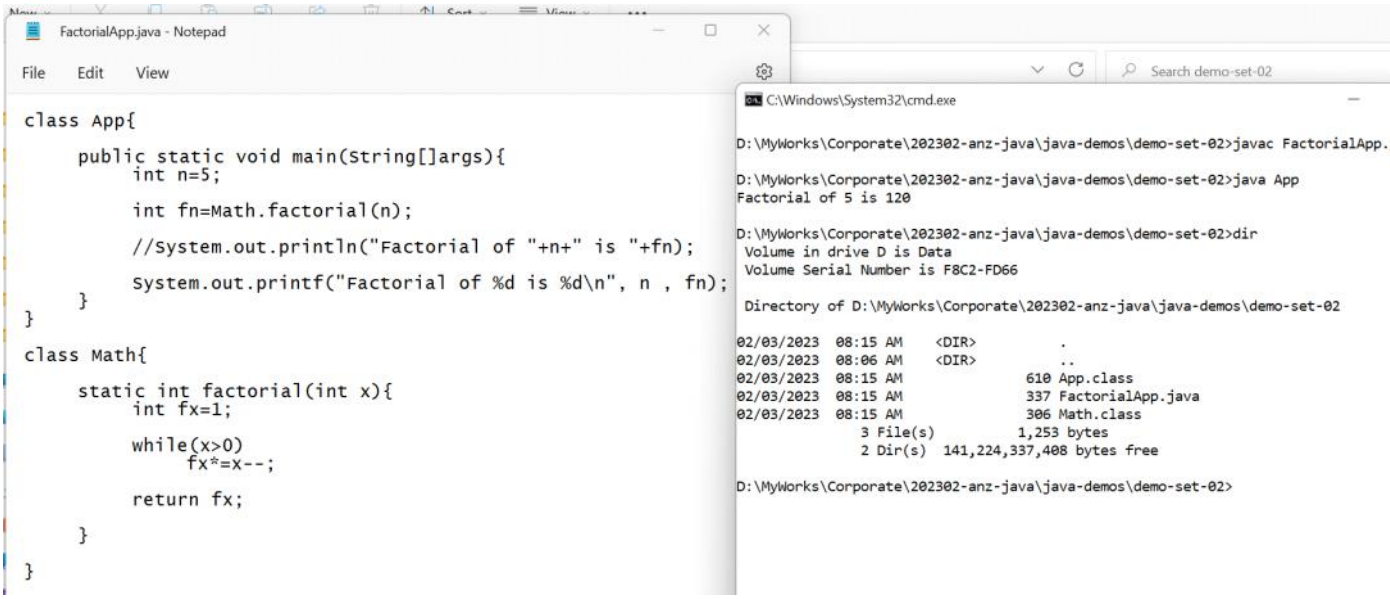
- break and contiue operates on the innermost loop in case of nested loop.
- you may need multiple breaks to come out of all the loops

90360 VIVEK

Multiple classes

Friday, February 3, 2023 8:16 AM

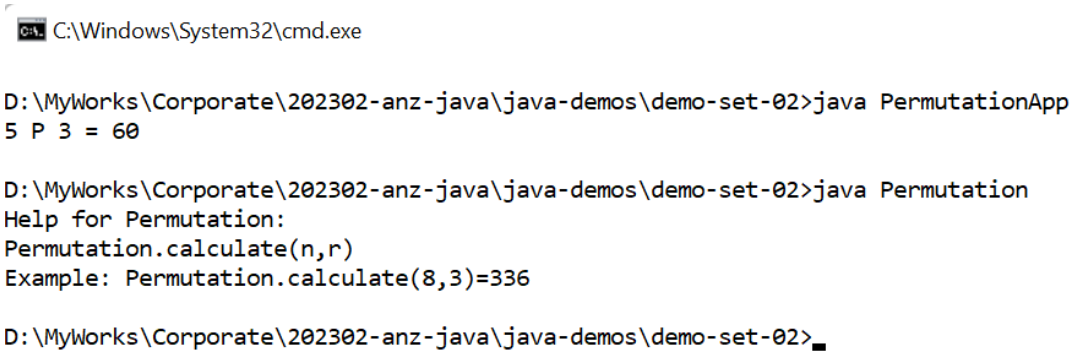
- When we compile a source file it generates one .class file per class (not per file)



How the Application is build with multiple source file?

- When we compile a source file say PermutationApp.java, java compiler finds it's dependency on class Permutation
- Now Java Compiler looks for a file Permutation.class
  - If present it uses the Permutation.class
- If Permutation.class is not present it looks for a source file with the same name
  - If present, it compiles the source file to get .class file
  - If it is not present, compilation aborts with error
    - IMPORTANT
      - While it is not compulsary to have source file and class file with same name, it is good to have to assist the compilation process.
- If both source file and class file is available, compiler checks for the modification date to find which one is latest
  - In case source file is modified after last compilation, it is recompiled
- IMPORTANT:
  - the source file is used only by java compiler and not by java runtime
  - Java runtime can't compile even if files are out of date.

Multiple Main class



- We can have multiple main class in different classes
- The one which is invoked with Java command will be caleld
- Use Case
    - A Dictionary class can be used as
        - stand alone dictionary app
        - embedded in Word to spell check.

Object Oriented Program

Friday, February 3, 2023 8:56 AM

What is a Program?

Set of instructions given to computer to perform some task.

Furniture Shop

Friday, February 3, 2023 9:22 AM

Objects

- Furnitures
  - Chair
    - Material
    - Price
  - Table
  - Bed
- List (Inventory)
- List (Customer)
- Invoice

- Multiple Chairs with similar property and behaviors (purpose)
  - **Each of them will have common set of elements like**
    - **Material**
    - **Price**
  - They may also have different features like
    - recliner
    - drawer
    - (we will not talk about them,yet)

Creating Object, The Java Way (common in most languages)

- We need a class to represent the idea of an object
- A class will describe what an object will be like
  - It can be considered as template or blueprint
    - **Why do we call it a class then? (pending question)**

- To create an object we need a class (to describe it)

```
class Chair{


}
```

- Now we can create multiple chair objects

```
Chair c1 = new Chair();

Chair c2 = new Chair();
```

- Now a class can contain informations related to the object

```
class Chair{

    int price=2000;

}
```

- Now our chairs can have price

```
Chair c1=new Chair();
Chair c2=new Chari();

System.out.println(c1.price); //2000;
```

- We can also change the price

```
c1.price=5000;
System.out.println(c1.price); //5000;
System.out.println(c2.price); //2000
```

- An object can also have it's own behavior or roles

```
class List{
    int items;
```

IMPORTANT!

- We, in most cases, would name our class as Singular
  - Chair, not Chairs
- A class is the description for a single Object
- Once we have the design description we can create multiple objects with same idea (class)

Note

- class doesn't have the price variable
  - It has the definion of price which will belong to the chair object

- Both chair object will have their own price
  - so now we have to price variables
    - c1.price
    - c2.price
  - here 2000 is the default price that will be intially assigned to all chairs
  - each chair can individually change it a

- Here addItem is NOT a static method
- Non static methods are referred as object level methods

```
class List{
     int items;
     void addItem( String item){
          items++;
          System.out.println("Item added");
     }

     int size(){
          return items;
     }

}
```

- Here addItem is NOT a static method
- Non static methods are referred as object level methods or instance methods
    - They belong to individual objects
- Most of your methods should be non-static
    - You are writing an object oriented program

- Now we can use these elements

```
List inventory = new List();

inventory.addItem("Chair");
inventory.addItem("Table");


List customers=new List();
customers.addItem("Vivek");

System.out.println( inventory.size()); //2
System.out.println(customers.size()); //1
```

Assignment 2.1

Friday, February 3, 2023 9:45 AM

- Create a List class
- Add the methods to
    - AddItem
    - RemoveItem
    - Size
- Test the methods with at least two list objects

Naming Convention

Friday, February 3, 2023 10:30 AM

```
class ClassList{

     int items; //defaults to 0

     void addItem(String item){
          items++;
     }

     void removeItem(String item){
          items--;
     }

     int countItems(){
          return items;
     }

}
```

- This is a working code.
  - But a working code may not be equal to a good code

- Important considerations
  - Class Name doesn't need a Class Prefix
  - We Generally avoid prefix in any code
  - Between Prefix and Suffix prefer suffix
    - avoid both if possible
  - While addItem is a good name
    - Item is redundant suffix
      - in list add means addList
      - It can be avoided in this context
    - we don’t need chairPrice and tablePrice in Chair and table class
      - Both can have price
      - meaning will be clear when we write
        - chair1.price
        - chair2.price
        - table1.price

Closer Look at the Objects

Friday, February 3, 2023 10:48 AM

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

	List customerList=new List();
	customerList.add("Vivek");


	List furnitures=new List();
	furnitures.add("Chair");
	furnitures.add("Table");
	furnitures.add("Bed");


	System.out.printf("Total Customers: %d\n", customerList.count());
	System.out.printf("Furntitures: %d\n", furnitures.count());

	Chair c1=new Chair();
	Chair c2=new Chair();

	c1.price=3000;
	System.out.printf("c1.price=%d\tc2.price=%d\n",c1.price,c2.price);


	Bed b1=new Bed();

	Inventory inventory=new Inventory();

	Invoice invoice1=new Invoice();

	Invoice invoice2=new Invoice();


	Table t1=new Table();


	System.out.println(t1);
	System.out.println(b1);
	System.out.println(inventory);
	System.out.println(invoice1);
	System.out.println(invoice2);
```

List

- A list has three methods
  - add
  - remove
  - count
- It has a property
  - items
- They are interconnected for the same object
  - add increases items count
  - remove declreases the same field
  - count returns the result for the same
- The two lists are different from each other
  - add or customerList and add of furnitures are incrementing different "items" variable
- Similarly we have two Chair with their individual prices

```
Total Customers: 1
Furntitures: 3
c1.price=3000	c2.price=2000
Table@33c7353a
Bed@681a9515
Inventory@3af49f1c
Invoice@19469ea2
Invoice@13221655
```

Note

- We are here printing the entire Object and not some property of Objects
- Java internally prints an object as String with two components separated by "@"
  - Class Name of that object
  - A unique Id or hashcode generated for each individual object
    - By default they will be different for each object
      - Known as hashCode

```
System.out.println(t1);
System.out.println(b1);
System.out.println(inventory);
System.out.println(invoice1);
System.out.println(invoice1.toString());
System.out.println(invoice2);
```

- This information is also available by calling a special method present in all "objects" called toString
  - Internally System.out.println is implicitly calling toString of the current object
- the hashCode can be checked by using another special method hashCode()
- whenever we want to print an object, it internally calls the toString method

What if I want to print a different information for my object?

we can write our own toString method

The Default ToString Behavior

```
customerList List@548c4f57
furntiures List@1218025c
```

After Adding our own toString

```
public String toString(){
     if(items==1)
          return "List of "+items+" item";
     else
          return "List of "+items+" items";
}
```

```
customerList List of 1 item
furntiures List of 3 items
```

Assignment 2.2

Friday, February 3, 2023 11:06 AM

- Define toString in list that should display the list items

```
List customerList=new List();

customerList.add("Vivek");
customerList.add("Sanjay");

List furntiures=new List();


System.out.println(customerList);
System.out.println(furnitures);
```

Expected Output

[\tVivek\tSanjay\t]

(empty)

Non Intialized variables

Friday, February 3, 2023 11:17 AM

- We have two types of variables we declare in Java
  - Fields
    - we declare inside the class
    - They represent the property of an object
    - They are by default initalized to
      - null for objects
      - 0 for numbers
      - false for boolean
      - '' for char
  - Method local variables
    - They remain uninialized till you initalize them
    - You can't use them without first initializing them

```
class Program{

    int number; //by default 0
    String name; //by default null;


    void f1(){
        int x; //un-initalized
        String y; //unitialized


        y="Hi";

        System.out.println(y); //works

        System.out.println(x); //fails. not
        intilized

        System.out.println(number); //works 0

        System.out.println(name); //works null

    }

}
```

Organizing Large Application

Friday, February 3, 2023 11:32 AM

- We have multiple classes in our code
- These classes represent different domain elements
- Furnitures
  - Chair
  - Table
  - Bed
- Generic Data
  - List
- Finance
  - Invoice
  - Inventory

- We don't want to keep all our files at one place
- Generally we may not want to incoude our source file in distribution
- How do we organize
  - Source and class files separately
  - Each domain related classes separately

Folder Based Organization

- we can create separate folder for
  - src
    - should contain .java files
  - class
    - should contain .class files

- Inside both these folders we can have sub folders representing domains
  - furntiures
  - data
  - finance

Our Project src structure

```
D:.
└───src
    │   FurnitureApp.java
    │
    ├───data
    │       List.java
    │
    ├───finance
    │       Inventory.java
    │       Invoice.java
    │
    └───furnitures
            Bed.java
            Chair.java
            Table.java
```

Will the compilation work?

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src>javac FurnitureApp.java
FurnitureApp.java:6: error: cannot find symbol
                List customerList=new List();
                ^
  symbol:   class List
  location: class FurnitureApp
FurnitureApp.java:6: error: cannot find symbol
                List customerList=new List();
                                      ^
  symbol:   class List
  location: class FurnitureApp
FurnitureApp.java:10: error: cannot find symbol
                List furnitures=new List();
```

Problem

- Java doesn't know where to go looking for classes (source or bytecode)

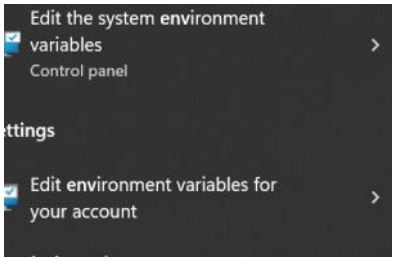
Solution : CLASSPATH

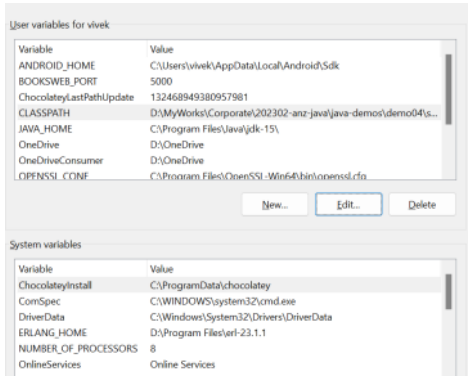
- Just like environment variable PATH that helps us locate executable files (eg. java, javac), java uses a system environment variable CLASSPATH to look for all the paths where classes are likely to be present
- A class path can maintain a list of PATH separated by
  - ; in windows OS
  - : in linux and MAC
  - These are as per the OS convention
- To make our design work we need to include all folders where we expect the path

Setting class path

- There are multiple ways to set the CLASSPATH

Option#1 In the environement setting of your system






```
D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src>echo %classpath%
D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src\furnitures;D:\MyWorks\Corporate\202302-anz-java\java-demos\de
mo04\src\data;D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src\finance

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src>javac FurnitureApp.java
```

```
Volume serial number is F8C2-FD66
D:.
└───src
    │   FurnitureApp.class
    │   FurnitureApp.java
    │
    ├───data
    │       List.class
    │       List.java
    │
    ├───finance
    │       Inventory.class
    │       Inventory.java
    │       Invoice.class
    │       Invoice.java
    │
    └───furnitures
            Bed.class
            Bed.java
            Chair.class
            Chair.java
            Table.class
            Table.java
```

A Small Snag

```
D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src>dir
 Volume in drive D is Data
 Volume Serial Number is F8C2-FD66

 Directory of D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src

02/03/2023  11:47 AM    <DIR>          .
02/03/2023  11:36 AM    <DIR>          ..
02/03/2023  11:47 AM    <DIR>          data
```

Why are we unable to find class file present in the current directory?

- By default Java/javac searches for class (both source/bytecode) in the current working directory

```
 Directory of D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src

02/03/2023  11:47 AM    <DIR>          .
02/03/2023  11:36 AM    <DIR>          ..
02/03/2023  11:47 AM    <DIR>          data
02/03/2023  11:47 AM    <DIR>          finance
02/03/2023  11:47 AM             1,783 FurnitureApp.class
02/03/2023  11:31 AM             1,092 FurnitureApp.java
02/03/2023  11:47 AM    <DIR>          furnitures
               2 File(s)          2,875 bytes
               5 Dir(s)  141,212,073,984 bytes free

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo04\src>java FurnitureApp
Error: Could not find or load main class FurnitureApp
Caused by: java.lang.ClassNotFoundException: FurnitureApp
```

current directory?

- By default Java/javac searches for class (both source/bytecode) in the current working directory
- Once CLASSPATH is set it searches only in the directories mentioned in the classpath
  - they stop searching in current folder

Solution

- We can always ask java to search in current directory by adding "." path in the CLASSPATH

Aside

- appending value to existing environment variable like class path

Windows

```
set CLASSPATH = %CLASSPATH%;.\something
```

Linux/Mac

```
set CLASSPATH = $CLASSPATH:./something
```

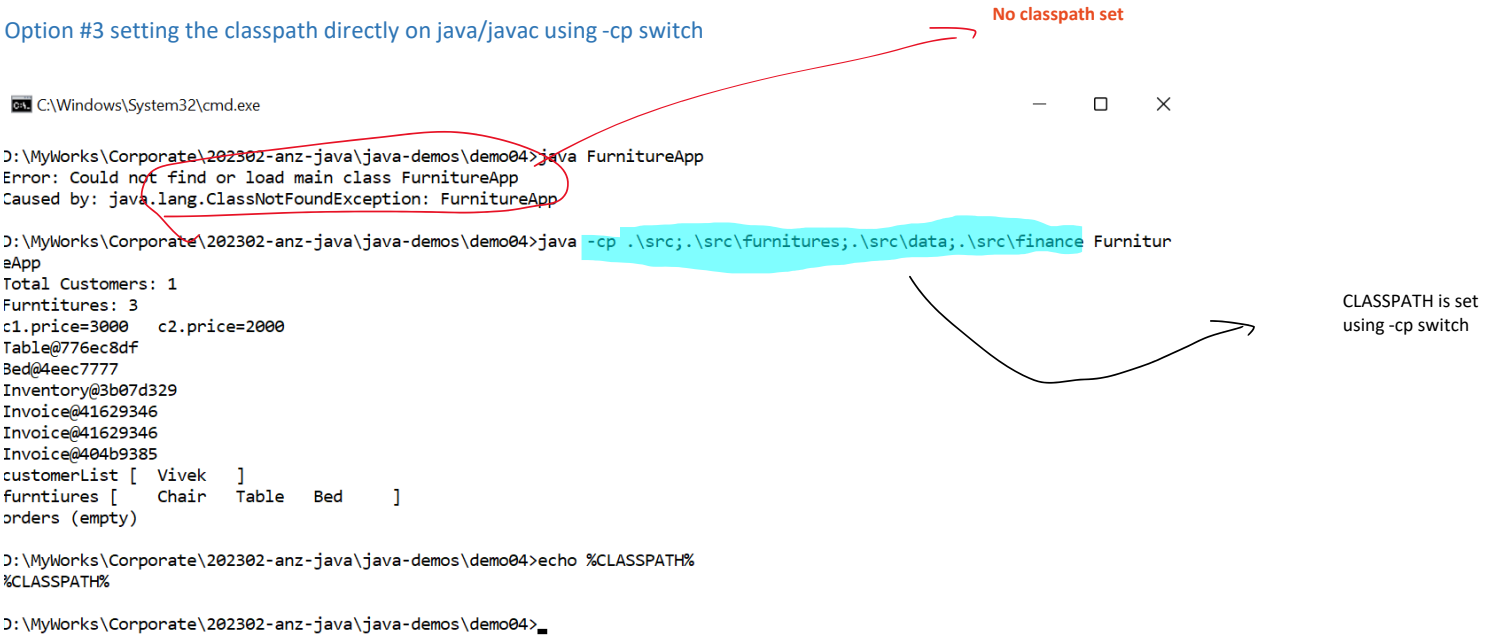
Why we shouldn't set classpath at the system level?

- we may have many applications
- Each will need its own classpath
  - they may conflict with each other

Option#2 creating classpath at application level

- we can declare the classpath directly on the command window
- Problem
  - we need to set the classpath everytime we open a command window
- Solution
  - Use a batch file / shell script

Option #3 setting the classpath directly on java/javac using -cp switch



How to separate source and class files in different folders

Class name conflict

Friday, February 3, 2023 12:27 PM

What is a Table?

Or

Data Table

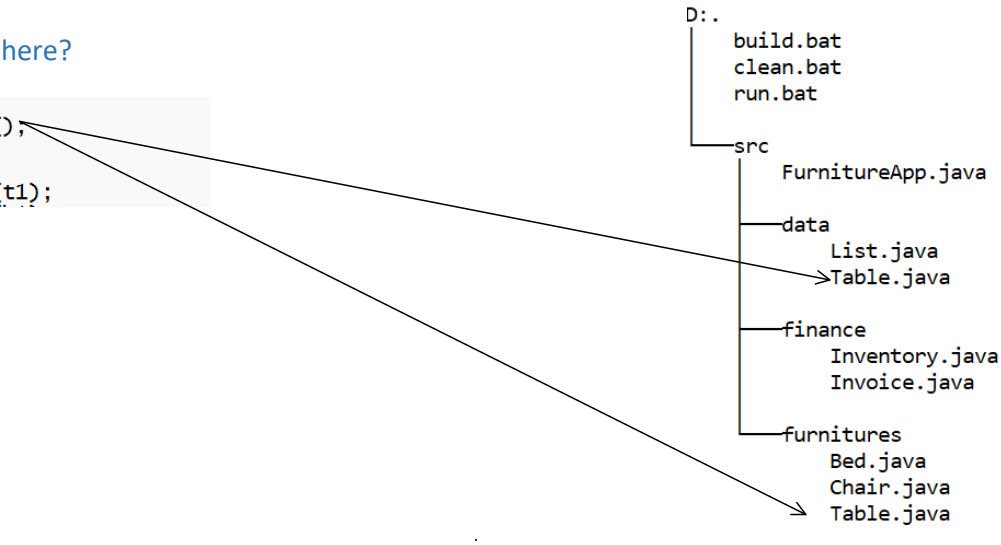
Furniture Table

Do we ever need both the table in the same application?

Which Table is referred here?

```
Table t1=new Table();

System.out.println(t1);
```



How will java decide which table to use?

- Java reads the class path in sequence in which it defined
- In our example we include "data" folder before "furnitures" folder

```
set APP_ROOT=.
set SRC=%APP_ROOT%\src
set CLASSES=%APP_ROOT%\classes
set CP=%CLASSES%\data;%CLASSES%\finance;%CLASSES%\furnitures
javac  -d %CLASSES%\data  %SRC%\data\*.java
javac -d %CLASSES%\finance  %SRC%\finance\*.java
javac  -d %CLASSES%\furnitures %SRC%\furnitures\*.java
javac -d %CLASSES% -cp %CP%  %SRC%\FurnitureApp.java
```

- As such it will be using data table and NOT furniture table
- A classpath search stops the moment a candidate class is located.

How to use both Table in the same class

- we can't keep them in same folder
- classpath will check the first folder only

Java Packages

Friday, February 3, 2023 12:39 PM

- Java Package is a logical grouping of classes and (sub) packages
- We can create packages like

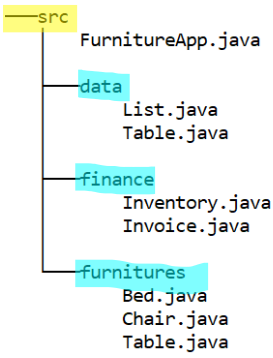
- furnitures
  - Chair
  - Table
  - Bed
- data
  - List
  - Table
- finance
  - Invoice
  - Inventory

What is the difference between a package and folder

- A package is a "java" concept, folder is an "os" feature
- A package name will be used within the java program, folders appear only externally in classpath
- A package will still be using folder structure
  - A package is not a folder
  - A packages lives inside a folder

How do we create a package

- we can designate any folder as a package
  - we may designated a nested folder structure as nested package
- The package appears in the source code

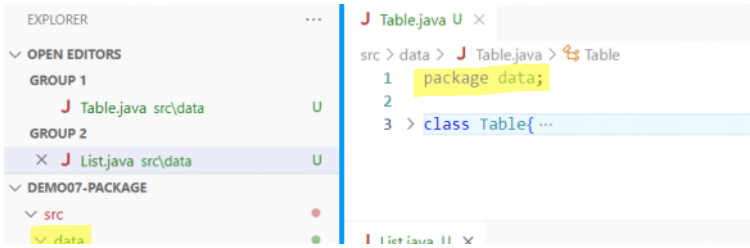


- Here we have designated following folder as packages
  - data
  - finance
  - furnitures
- A package becomes language concpet and doesn't appear in classpath

- we are not considering "src" as package
  - src is a container path for packages
  - This folder will appear in the CLASSPATH

- Why is "src" not a package?
  - Because we don't want

Step 1 Designating Package for Class



- package "data" should match the immeidate folder structure
  - more important for .class file than .java file
- if we chose to name our package as src.data
  - src package will contain subpackage data
- In our example data is package, src is container path for the package.

- If we chose to name our package as src.data
  - src package will contain subpackage data
- In our example data is package, src is container path for the package.

Compiling Classes from a Package

```
D:\MyWorks\Corporate\202302-anz-java\java-demos\demo07-package>javac -d .\classes .\src\data\*.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo07-package>tree/f
Folder PATH listing for volume Data
Volume serial number is F8C2-FD66
D:.
    build.bat
    clean.bat
    run.bat

├───classes
│   └───data
│           List.class
│           Table.class
```

- even when we have marked to store compiled classes to "classes" folder because they belong to a package "data" compiler generates the package folder and stores it

Step 2 Using classes from a Package

- Now we don't have a non-packaged (global) class like
  - Chair
  - Table
  - List
- We have classes like
  - furnitures.Chair
  - furnitures.Table
  - data.Table

Note

- Now we have two distinctly identifiable Tables
  - furnitures.Table
  - data.Table

```
D:\MyWorks\Corporate\202302-anz-java\java-demos\demo07-package>javac -cp .\classes\data -d .\classes .\src\FurnitureApp.
java
.\src\FurnitureApp.java:6: error: cannot access List
                List customerList=new List();
                ^
  bad class file: .\classes\data\List.class
    class file contains wrong class: data.List
    Please remove or make sure it appears in the correct subdirectory of the classpath.
1 error

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo07-package>
```

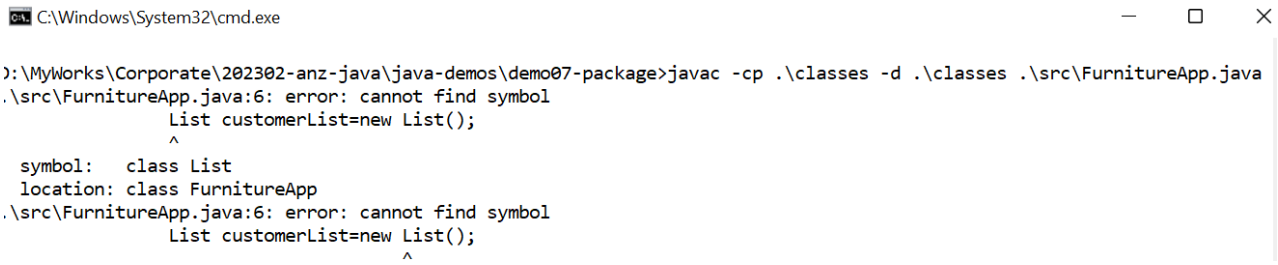
Problem

- classpath includes "data" as sub folder
- compiler enters this folder and tries to search for package "data" which is not present

Solution

- package name shouldn't be part of classpath
- parent of package should be part of classpath
  - classpath is used for searching both
    - class
    - package

Problem 2



- Now It is searching for List class in classes folder
  - It doesn't exist
- In fact we don't have a global List class
  - we have data.List

Solution 2.1 (Step 2.1) using package qualified names

Problem #3

```
D:\MyWorks\Corporate\202302-anz-java\java-demos\demo07-package>javac -cp .\classes -d .\classes .\src\FurnitureApp.jav
.\src\FurnitureApp.java:6: error: List is not public in data; cannot be accessed from outside package
                data.List customerList=new data.List();
                     ^
.\src\FurnitureApp.java:6: error: List is not public in data; cannot be accessed from outside package
                data.List customerList=new data.List();
                                           ^
.\src\FurnitureApp.java:10: error: List is not public in data; cannot be accessed from outside package
                data.List furnitures=new data.List();
                     ^
.\src\FurnitureApp.java:10: error: List is not public in data; cannot be accessed from outside package
                data.List furnitures=new data.List();
```

- So far all our classes belonged to an unamed global package
- They all belonged to same family and can access each other without problem
- Now List belongs to a different package "data" and can't be accessed outside the package unless it is marked public
  - same goes true for list members
    - add
    - remove
    - count

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

    data.List customerList=new data.List();
    customerList.add(item: "Vivek");

    data.List furnitures=new data.List();
    furnitures.Table t1=new furnitures.Table();
    data.Table t2=new data.Table();
```

Advantage

- We can access both Tables
  - data.Table
  - furnitures.Table
- We have smaller class paths
  - we don't need packages folders to be part of classpath
- Easy compile and run
  - we just need one class path
  - It can compile all dependency classes propely
- Auto organization of classes in right package folders

Problem

- We need to include the package qualified name everywhere
- When we have many classes (we always have many classes) package qualified names becomes difficult

Option 2.2 import statement

- we can import a particular pakcage contents (classes) directly so that we can use them without qualfiying the packae name

```
import finance.*; //get all the classes from finance packae

class FurnitureApp{


    public static void main(String []args){


        Invoice i1=new Invoice();
        Invoice i2=new Invoice();
        Inventory inventory=new Inventory();

        System.out.println(i1);
        System.out.println(i2);
        System.out.println(inventory);
```

Note

- import "*" can import all classes from a pacakge not the sub packages
- there is nothing like *.*
- Once imported you can use all the classes from there

Problem with * import

- We generally avoid importing the entire package
- If we import all package with "*" it will be problem similar to not having package

```
    Table t1=new Table();
```

```
C:\Windows\System32\cmd.exe

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo07-package>build
.\src\FurnitureApp.java:20: error: reference to Table is ambiguous
                Table t1=new Table();
                ^
  both class furnitures.Table in furnitures and class data.Table in data match
.\src\FurnitureApp.java:20: error: reference to Table is ambiguous
                Table t1=new Table();
                         ^
  both class furnitures.Table in furnitures and class data.Table in data match
2 errors
```

Option #3 importing a class selectively from a package

- you may specify which class you want to import

```
import finance.*; //get all the classes from finance packae
import data.*;
import furnitures.*;

import furnitures.Table;


class FurnitureApp{


    public static void main(String []args){


        Invoice i1=new Invoice();
        Invoice i2=new Invoice();
        Inventory inventory=new Inventory();

        List customerList=new List();
        List furnitures=new List();

        Table t1=new Table();
```

What if we need both Tables

- In such cases we need to use one of the reference explicitly as fully qualified name

```
import furnitures.Table;

class FurnitureApp{


    public static void main(String []args){


        Invoice i1=new Invoice();
        Invoice i2=new Invoice();
        Inventory inventory=new Inventory();

        List customerList=new List();
        List furnitures=new List();

        Table t1=new Table();   //furnitures.Table

        data.Table t2=new data.Table(); //explicit selection
```

Recommendation

- Java best practice guidelines recommends importing classes rather than package.*

What is the possibility that two different prorammer will create a package with same name and have same class inside it

- High possibility
- Package is a bundle of related classes
- If package name is same chances are we will create classes also in the same way

What is the possibility that we need packages created by two developers in the same project

- folder
- package

- **src**
  - class App
  - **vivek**
    - **data**
      - class List

        - class Tree
    - **furnitures**
        - class Chair
- **sanjay**
    - **data**
        - class Search
        - class Table
        - class List

How do I access both Tree and Search class?

- we can have both src\sanjay and src\vivek in CLASSPATH
- Now we have a **single package (logical entity) called data** which holds
    - List
    - Tree
    - Search
    - Table
    - List

- When we see multiple package definitions they merge as a single logical unit

compilation

```
$ javac -d .\classes -cp .\src\vivek;.\src\sanjay;.\src App.java
```

Run

```
$ java -cp .\classes  App
```

What if I want to access List?

- Here it will access the List from that package folder which appears first in CLASSPATH

How do I access both the List?

- There is NO Java WAY.

Takeaway

- Contents inside two classes never conflict
    - class acts as boundry
    - Two classes can have field and methods with same name
- Class names may conflict
- This conflict can be resolved using packages
- Package names don't conflict. They merge
- When we have two package folders (same package) with same class, then we have a problem that can't be resolved
    - whichever folder is first in the list will be used.
    - other is unreachable

Solution

- to avoid conflict within the package, we use the concept of nested package
- generally we use outer package as identity space (identification of the creator)
    - Example
        - package vivek.data
        - package sanjay.data
- What is the possiblity of name conflict between vivek.data and sanjay.data
    - vivek and sanjay are very common names and most likely will conflict

Package naming convention

- A package should always be nested
- The outer most package (generally 2) defines identity
    - conventionally we use reverse domain as unique identity

        - in.conceptarchitect
        - org.apache
        - com.anz
    - starting third level package we may use for logical grouping

- example
    - in.conceptarchitect.commons.data
    - in.conceptarchitect.commons.finance
    - in.conceptarchitect.app.furnitureapp.furnitures
    - in.co.sanjay.data

Distribution

Friday, February 3, 2023 3:01 PM

- A java project will have typically hundreds of classes in dozens of packages and sub packages (folders and sub folders)yste
- Distributing files this way is goint to be difficult
- Java provides a simpler alternative

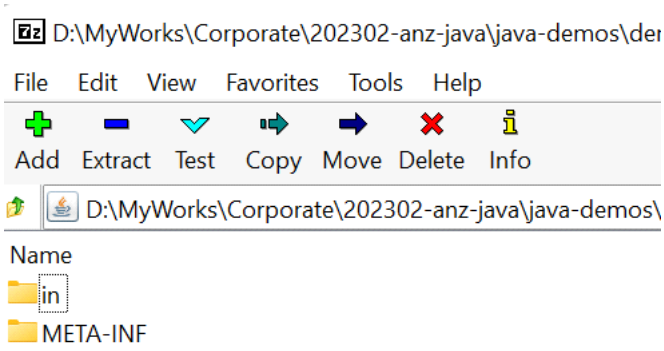
Jar file

- Jar stands fro Java Archive
- Concept is similar to a zip file
- You bundle the class and package in a single file
- Java can run the application without uncompressing this file
- It improves performance as you will have fewer I/O tor ead the archive

To create a Jar file

```
jar -cf  ..\app.jar  .
```

- create a jar file including all files and folder and sub folder from current folder
- the jar file should be saved as ..\app.jar



- A jar contains all my files
- It also include some Meta information needed by Java

Running the program from Jar

- we can just use the jar file as class path

Manifest

- can include any information realated to jar as key value pair
- we need to create our own manifest file and add the information
- information provided by us shall be merged in actual manifest

Recursive Function

Monday, February 6, 2023 9:19 AM

- A Recursive Function is a Function Calling Itself
- Sometimes a large and complex algorithm can be broken up into some other term of itself
- Example

- Factorial of 5

5 x 4 x 3 x 2 x 1

Factorial of 4

- Now we can say

5! = 5 * 4!

- Programmetically we can write

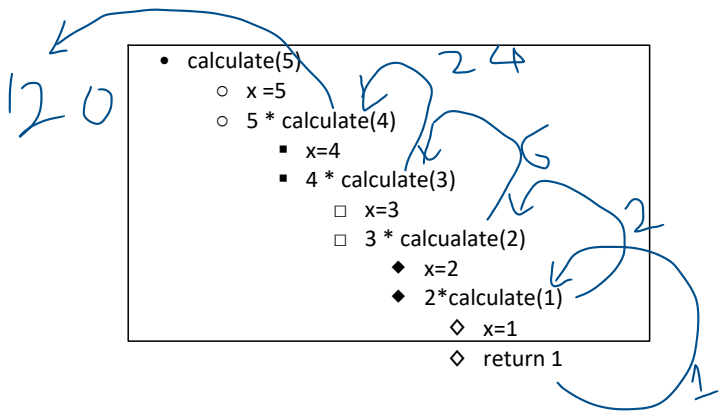
Factorial.java

```
package in.concpetarchitect.maths;

public class Factorial {

    int calculate(int x) {
        if(x<=1)
            return 1;
        else
            return x* calculate(x-1);
    }

}
```



How many x we have?

- There are 5 different "x" variable present in memory at this point in time
  - each x is different from each other
- a new set of variables are created each time a method is called
  - a new set of parameters
  - a new set of all local variables declared within the method
- A class level field (static) is created only once
- An object level (non-static) field is created per object

Important Note

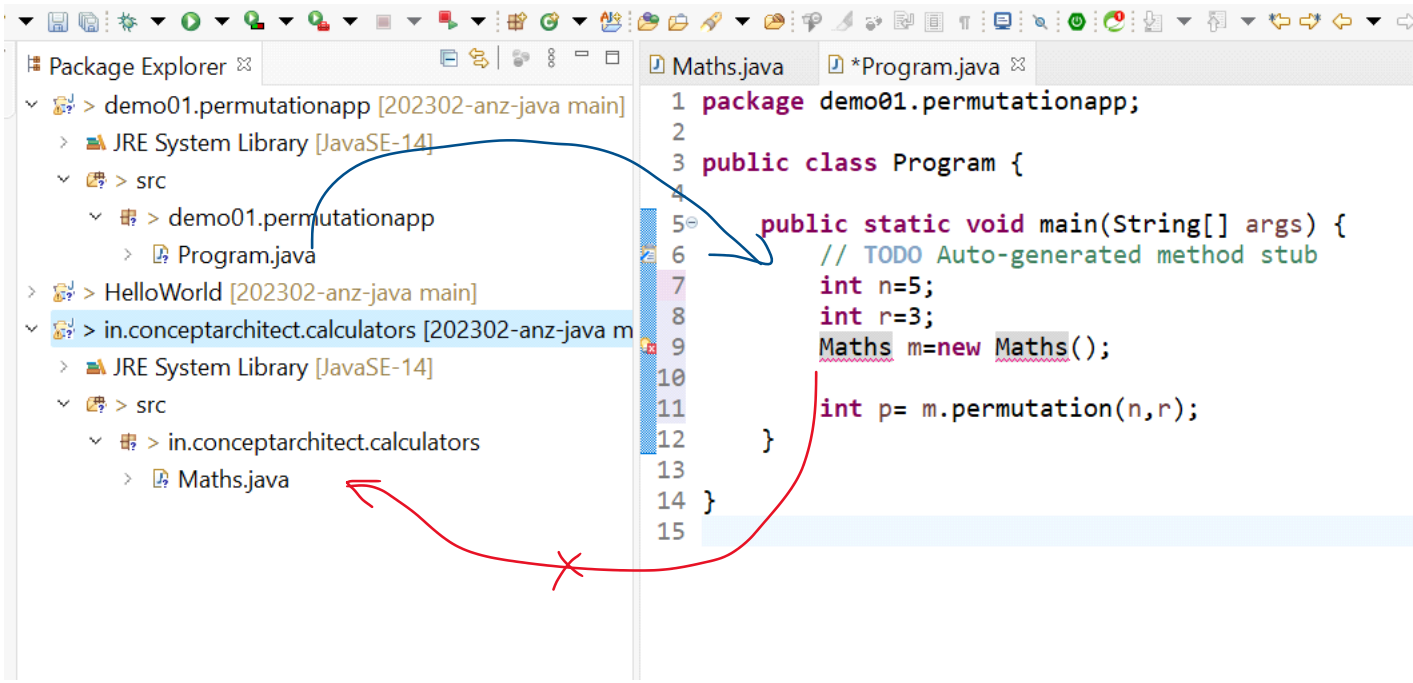
- for every recursive function there should be at least two returns
- One recursive return (that is why it is recursive function)
- One non-recursive direct return
  - If we don't have a direct return the method may enter infinite loop causing "stack overflow error"

Using class from other eclipse Project

Monday, February 6, 2023 10:24 AM

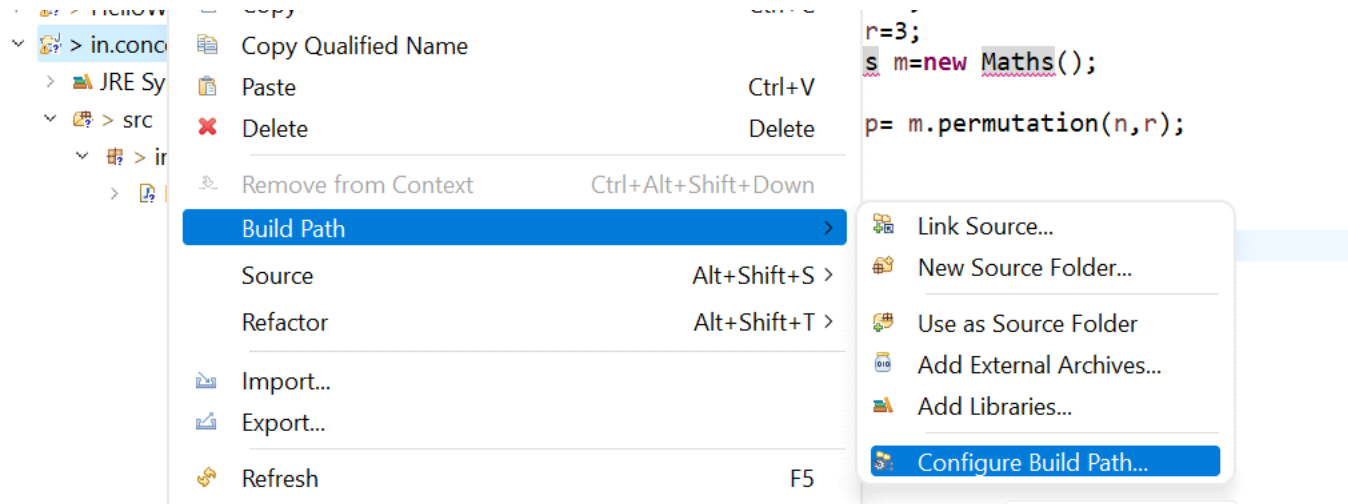
How do I access class from other projects

- By default eclipse will search the classes within the project itself
- ctrl+space or ctrl+shift+o will not get details from other projects within or outside the workspace

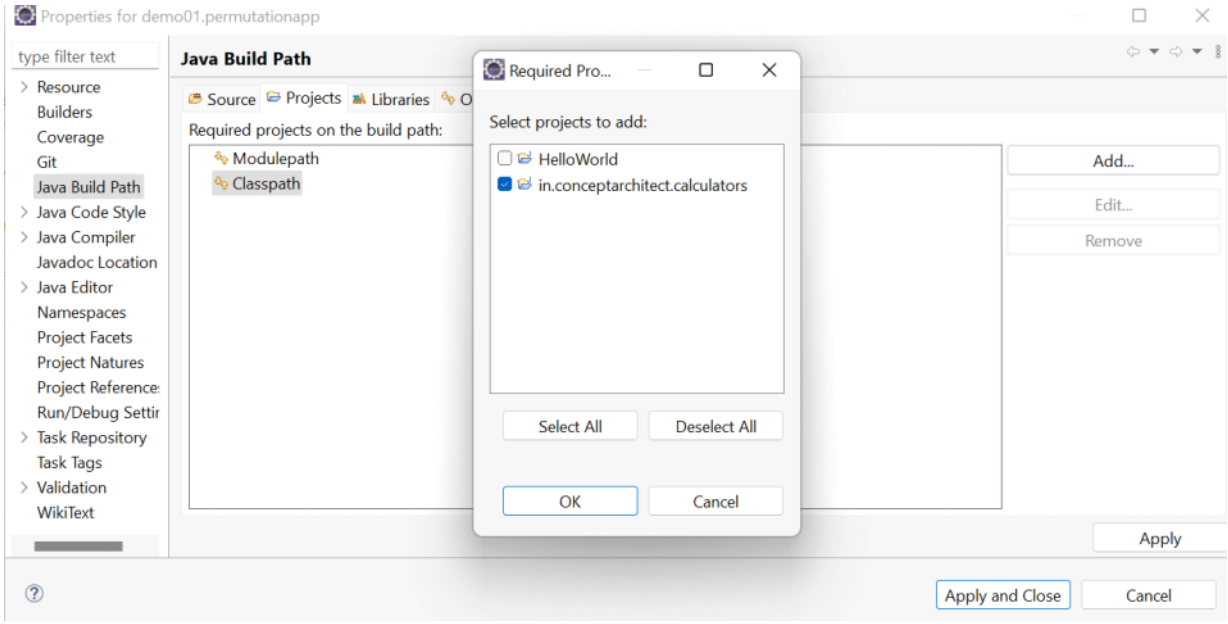


Adding Reference of One Project in another

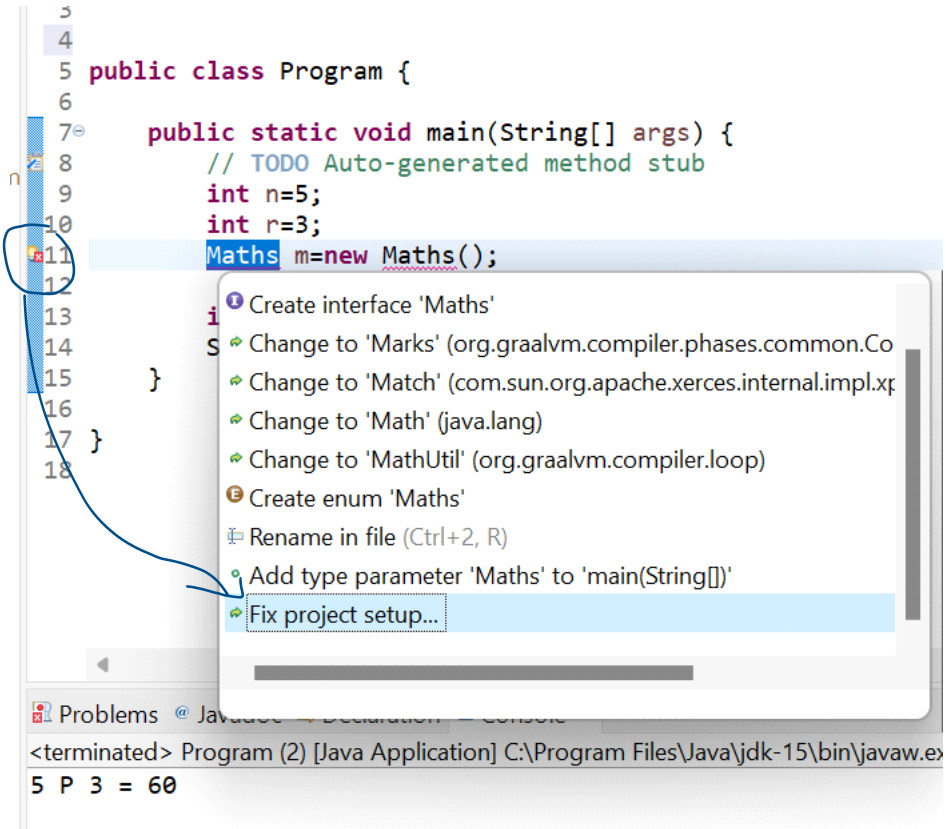
1. Right click project and select the option

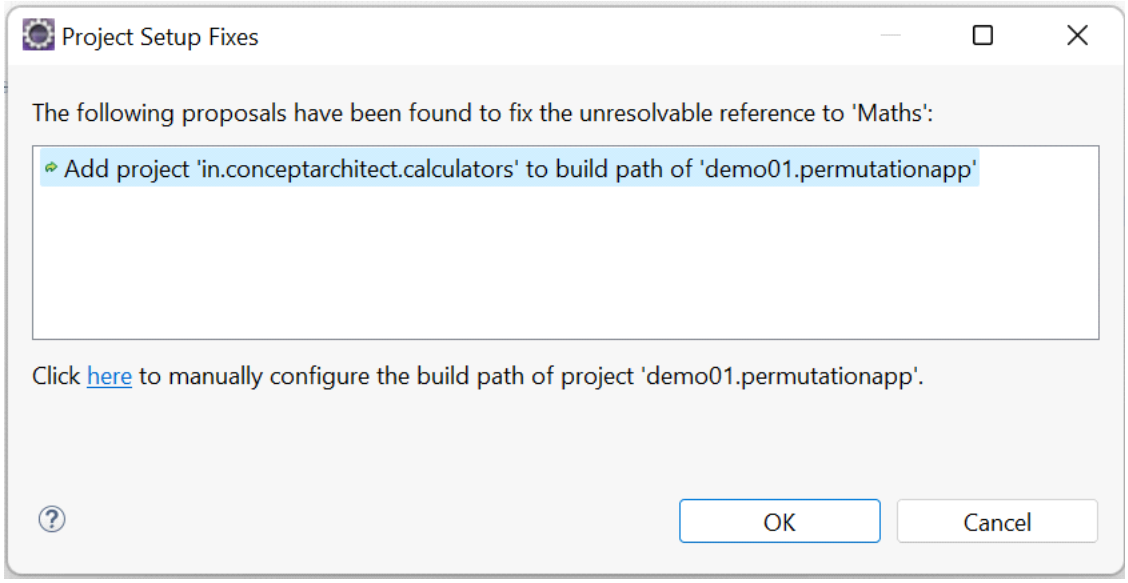


2. Select the Project in the build-path



Alternative Option

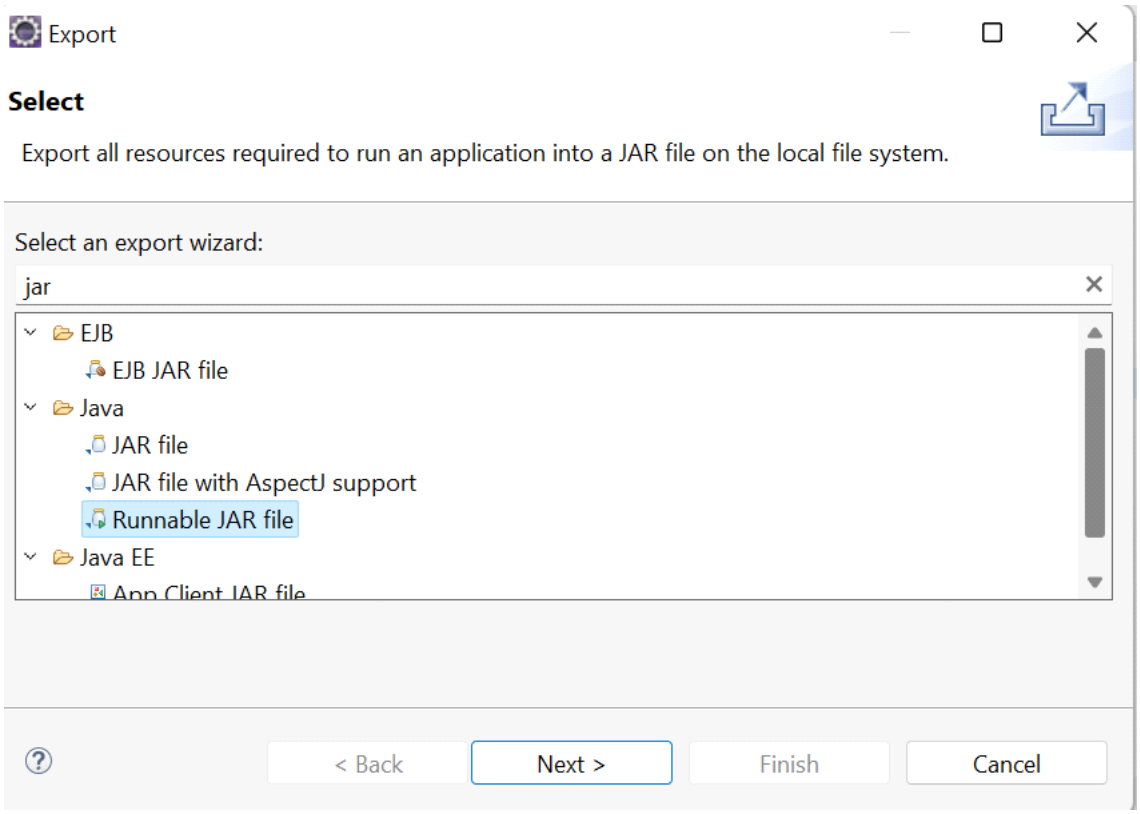



Project Setup Fixes
The following proposals have been found to fix the unresolvable reference to 'Maths':
Add project 'in.conceptarchitect.calculators' to build path of 'demo01.permutationapp'
Click here to manually configure the build path of project 'demo01.permutationapp'.
OK
Cancel

Creating jar from eclipse

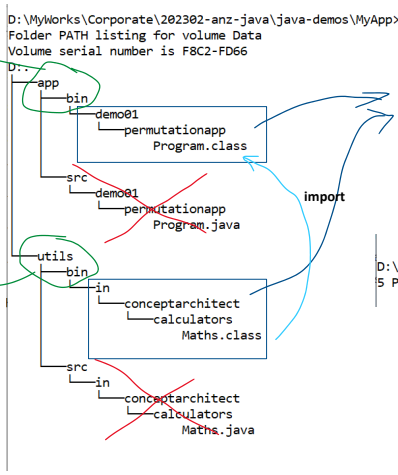
Monday, February 6, 2023 10:36 AM

1. File —> export



Understanding Runtime Classpath

Monday, February 6, 2023 10:55 AM



import

ClassPath

- classpath is the path where Java searches for it's class
  - Since Package is a prefix for class, classpath also searches for packages

Package

- Although we have a physical directory here, they are considered as Java Package
  - And when we see them as package, we don't see them as normal OS level folder

```
D:\MyWorks\Corporate\202302-anz-java\java-demos\MyApp>java -cp .\app\bin;.\utils\bin demo01.permutationapp.Program
5 P 3 = 60
```

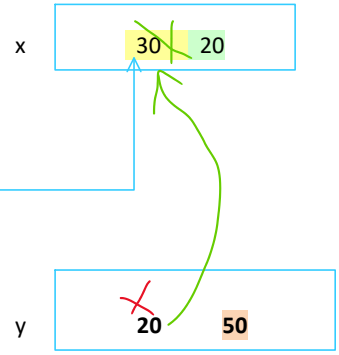
Primitive Type (Value type) Memory allocation

Monday, February 6, 2023 11:38 AM

```
int x;   //memory is allocated for 1 int. It may not be initialized yet
```

x = 30;

int y=20; //memory allocated and initalzed to gether



Assigning another value to a variable

```
x=y;
```

- current value of y is copied to X. But they are not related any further

```
y=50;
```

- replaces the value of y from 20 to 50
- No change in the value of x, which is unrelated

Object (Ref type) Memory Allocation

Monday, February 6, 2023 11:35 AM

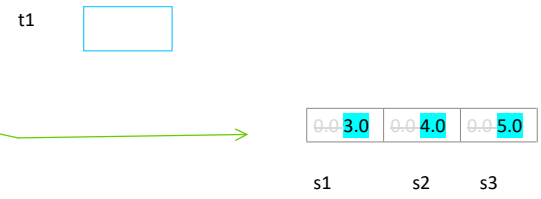
```
Triangle t1;

t1= new Triangle();

t1.setSides(3,4,5);
testTriangle(t1);


Triangle t2=new Triangle();
t2.setSides(1, 1, 1);
testTriangle(t2);
```

- creates a reference (pointer) for a triangle object.
  - Triangle object doesn't exist yet
  - No memory allocated to store s1,s2,s3 at this stage



- **new Triangle() is when we allocated the memory for actual object**
  - **The memory is allocated for all non-static fields declared in class**
  - **They are initialized all to 0 (in binary)**
    - **numbers —> 0**
    - **booleann —> false**
    - **object references —> null**

- **values are modified from original state**

IMPORTANT NOTE

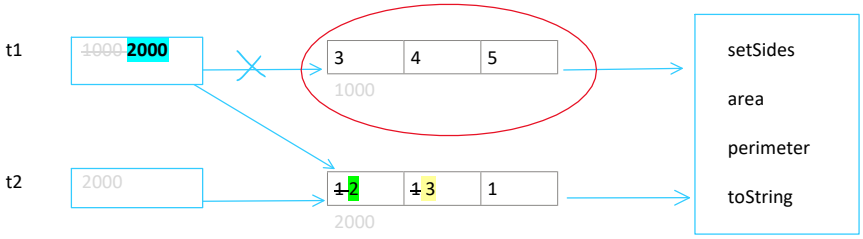
- Memory is only allocated for non-static fields declared inside the calss
- No memory is allocated at this stage for
  - Methods of the class
    - A common copy is used for all objects
  - Any method parameter or method local
    - They are allocated when you call the method
  - Any class field marked static
    - A single copy is stored in memory
    - More on this later.

Working with multiple Objects

- all objects share same set of methods

```
Triangle t1= new Triangle();
t1.setSides(3,4,5);

Triangle t2=new Triangle();
t2.setSides(1,1,1);
```



Assigning One Object to another

```
t1=t2;
```

- Here the reference to t2 will be copied to t1
  - It will not copy the object contents
  - Just the reference
- At this stage both t1 and t2 are referring to same object
  - Triangle<1,1,1>

What happens to Object Triangle<3,4,5>?

- In self managed codes (like c/c++) this is considered as a memory leak
  - This object has no reference and it can't be used.
  - It will remain in memory forever (till the app is running)
  - They memory can't be re-used
  - We MUST free the memory by deleting the object before this type of assignment

- It will not copy the object contents
- Just the reference

- At this stage both t1 and t2 are referring to same object
  - Triangle<1,1,1>

If we change any one, it changes both

```
t1.s1=2;
t2.s2=3;
```

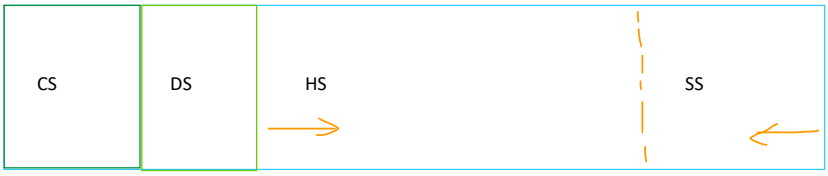
Actually we have just one object here which is referred by two different references

- Any change by any reference changes the same object.
- both t1 and t2 will have the same hashcode as they are the same object

- It will remain in memory forever (till the app is running)
- They memory can't be re-used
- We MUST free the memory by deleting the object before this type of assignment

- In Managed languages like (Java/C++/Python/JavaScript/...)
  - The runtime has a process called "garbage collection"
  - The process frees memory at un undeterministic interval
  - All the un-referenced memory may be freed by garbage collector by it's own strategy or convinience
    - It is a complex and evolving process
    - It involves several generations
    - garbage collection typically works when
      - we start to run out of memory
      - when system resources are comparatively free/idle
    - Even when garbage collector runs there is no surity that it will free all the memory.
      - It may free just one generation of memory
  - Java has a API to force garbage collection
    - This api can initiate garbage collection
      - Even this api doesn't gurantee IMMEDIATE
        - It is meant to be suggestive not authoratative
        - although gernerally it is IMMEDIATE
    - In MOST cases it is not recommened to interfere in the process.

Memory Layout of a typically Application



- Code Segment
  - stores your logic
    - class methods
- Data Segment
  - Stores static and const values
- Heap Segment
  - Stores dynamically allocated memory
  - "new"
- Stack Segment
  - Stores method locals and parameter
- There is no hard division between Stack and Heap
  - They grow towards each other dynamically

Java Heap Management

- Java Heap Management works on assumptions that
  - An object either dies very young or lives to grow old.
    - there will be many object created within a method call and are not used once the call is over.
    - Few objects are required throughout application and may live for entire life

- Java Garbage collection has generation model
  - Gen 0
  - Gen 1
  - Gen 2
  - ...

- All objects are always created on Gen1 Heap
  - few will be dead long before garbage collection
  - other may live on.
- When garbage collectors starts it starts for a particular generation (not for everyone)
  - Gen1 garbage collector may start when gen0 heap is (almost) full
  - It checks for all living objects (not dead ones)
    - It moves all the surviving object to gen 1
      - Note address will change and that is why java never gives the address to program
      - All the references to this object is automatically changed to new address

        - All the dead objects are removed and gen 0 is now completely empty
    - Same thing will happen to gen1 and gen2

Triangle Revisited

Monday, February 6, 2023 1:23 PM

```
Triangle t3=new Triangle();
t3.setSides(2,4,8);
testTriangle(t3);
```

- As per java we have a valid object referred by t3.
- But we can't create a triangle with dimension 2,4,8
- Geometrically (domain rule) for a valid triangle
  - sum of every two sides must be greater than the third.

How do I model a valid Triangle?

- we need to incorporate the triangle rule in the domain object.

Approach #1

- validate value before assigning and display error message otherwise

```
3  public class Triangle {
4
5      double s1,s2,s3;
6
7      void setSides(double x, double y, double z) {
8          if(x>0 && y>0 && z>0 && x+y>z && y+z>x && x+z>y) {
9              s1=x;
10             s2=y;
11             s3=z;
12         } else {
13             //what to do when user enters wrong info?
14             System.out.println("invalid sides");
15         }
16     }
17
18
19     double perimeter() {
20         return s1+s2+s3;
21     }
```

- indicates we have error

Error display may not always be best option

- This display doesn't prevent permieter calculation.
- Any value returned will essentially be wrong as invalid triangle shouldn't have a perimeter.

- perimeter function has no knowledge of any message displayed by setSides.

Approach #2 set a flag (indicator) to mark triangle valid or invalid

```
3  public class Triangle {
4
5      double s1,s2,s3;
6      boolean valid;
7
8      void setSides(double x, double y, double z) {
9          if(x>0 && y>0 && z>0 && x+y>z && y+z>x && x+z>y) {
10             s1=x;
11             s2=y;
12             s3=z;
13             valid=true;
14         } else {
15             //what to do when user enters wrong info?
16             //System.out.println("invalid sides");
17             valid=false;
18         }
19     }
20
21
22     double perimeter() {
23         if(valid)
24             return s1+s2+s3;
25         else
26             return Double.NaN;
27     }
```

- Here we have a Triangle Object
- When we setSides it also internally sets a valid flag to specify if triangle is valid or not
- other behavior of this triangle respects "valid" status and returns expected answers for valid and invalid scenario.

Binding of Data and Behavior (Encapsulation)

- In this object the triangle states (s1,s2,s3,valid) are interconnected.
- setSide sets the values as per the requirement
- area() and perimeter() are also connected to the same triangle rule and represents proper domain model
- internally all the states and behavior together represent Triangle
- **Encapsulation is more about defininig a responsibility.**

But what if client is not reasonable/responsible?

```
//t1 is a valid triangle
//What if I change one of it's side making it invalid
t1.s1=100;  //the valid flag is not changed.

testTriangle(t1);
```

```
Triangle <100.0,4.0,5.0>
Perimeter: 109.0                Area: NaN
```

- we may bypass setSides and change values directly
- Now s1 has changed but valid flag is not reset

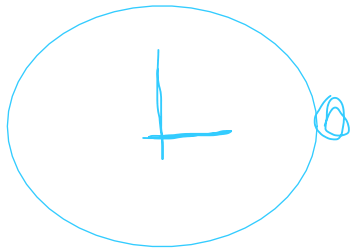
```
//t3 is invalid (sides are 0,0,0)
//but we can change the valid field

t3.valid=true; //a simple lie.
testTriangle(t3);
```

```
Triangle <0.0,0.0,0.0>
Perimeter: 0.0          Area: 0.0
```

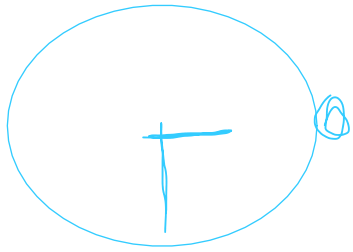
- here Triangle was marked invalid by setSide
- But we can unmark it by resetting valid flag from outside the Triangle object

Encapsulation recommends protection against unwanted changes.



- In our code we should not allow anyone to access my state (data) directly
- It should change using authorized behavior model.

Scope Rules



Why do we need a time changing knob and not manually shift hand?

- Ideally when min hand moves the hour hand too should move

- public
  - —> accessible by everyone
- (package)/no scope
  - —> accessible by everyone within the same package
- private
  - accessible only within the class and not outside
- protected
  - to be disucssed later.

```
//t1 is a valid triangle
//What if I change one of it's side making it inval
t1.s1=100;  //the valid flag is not changed.

testTriangle(t1);


//t3 is invalid (sides are 0,0,0)
//but we can change the valid field

t3.valid=true; //a simple lie.
testTriangle(t3);
```

- with private s1 and valid no body can temper with this information
- This information will be set only using right set of values

Problem —> what if we need to see the value

```
private static void testTriangle(Triangle triangle) {
    System.out.println(triangle);
    if(triangle.valid) {
        System.out.print("Perimeter: "+triangle.perimete
        System.out.println("Area: "+triangle.area());
    }
    System.out.println();
}
```

- Now outsiders can't even see if triangle is valid or not
- It is a valid use case.

Solution —> define a method to return the valid/invalid status

- we should allow only checking for the information and not changing it.

```
private double s1,s2,s3;
private boolean valid;

public boolean isValid() {
    return valid;
}
```

- Here we can check the validity but not change from outside

A little refactor

- let's change the valid flag to invalid flag

Wh

Problem — What if we never call setSides?

```
28        Triangle t4=new Triangle();
29
30        testTriangle(t4);
31
```

- triangle by default is "valid"
- if no side is set it may consider side 0 to be valid

When is the triangle created : Line 12 or Line 14?

```
12        t1= new Triangle();
13
14        t1.setSides(3,4,5);
15
16
17        testTriangle(t1);
```

- If Triangle is created on Line 12
  - what are the sides of triangle on line 13?
  - can a "valid" triangle exists with side 0?
- If Triangle is created on Line 14
  - What was the value of "t1" on line 13?
  - what is we can t1.perimeter() on line 13?

Right answer

- There are two creations here
- Java Object is created on Line 12
  - memory is allocated
  - but the object is not a geometrical triangle
  - it is not usable yet.
- Geometrical triangle is not ready till line 14 is called
  - This is where domain object is created
- Real Problem
  - There are two ideas
    - java object
    - domain object
  - There creations are no in sync
    - there is a gap

Object Oriented Concept —> Constructor

- constructor is a special class method with same name as that of class

```
Triangle t= new Triangle();

|                    |————————> constructor of the class object
|
|
|————————————————————> class name
```

- The two are same name but different concepts

- This method has no return type but returns the newly created object

- This method is called for creating the object with new
- Every class contains a default nothing doing constructor
- We can define our own constructor that replaced the default one.
  - if we define a constructor the default will be removed.
- Our constructor can take multiple arguments
  - We can have overloaded constructor
    - multiple constructor taking different number or type of parameter

Assignment 3.1

Monday, February 6, 2023 2:24 PM

- Create a model for a Bank Account
- We should have following information
  - name
  - account number
  - balance
  - interest rate
  - password

- It should support following operations
  - deposit
    - should reject negative amount
  - withdraw
    - should fail for
      - negative amount
      - access withdrawal
      - invalid password
  - credit interest
    - gives one month interest with formula
      - balance = balance*rate/1200
  - to string
    - to show the account object as string

- Write a test app to work with bank account