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**

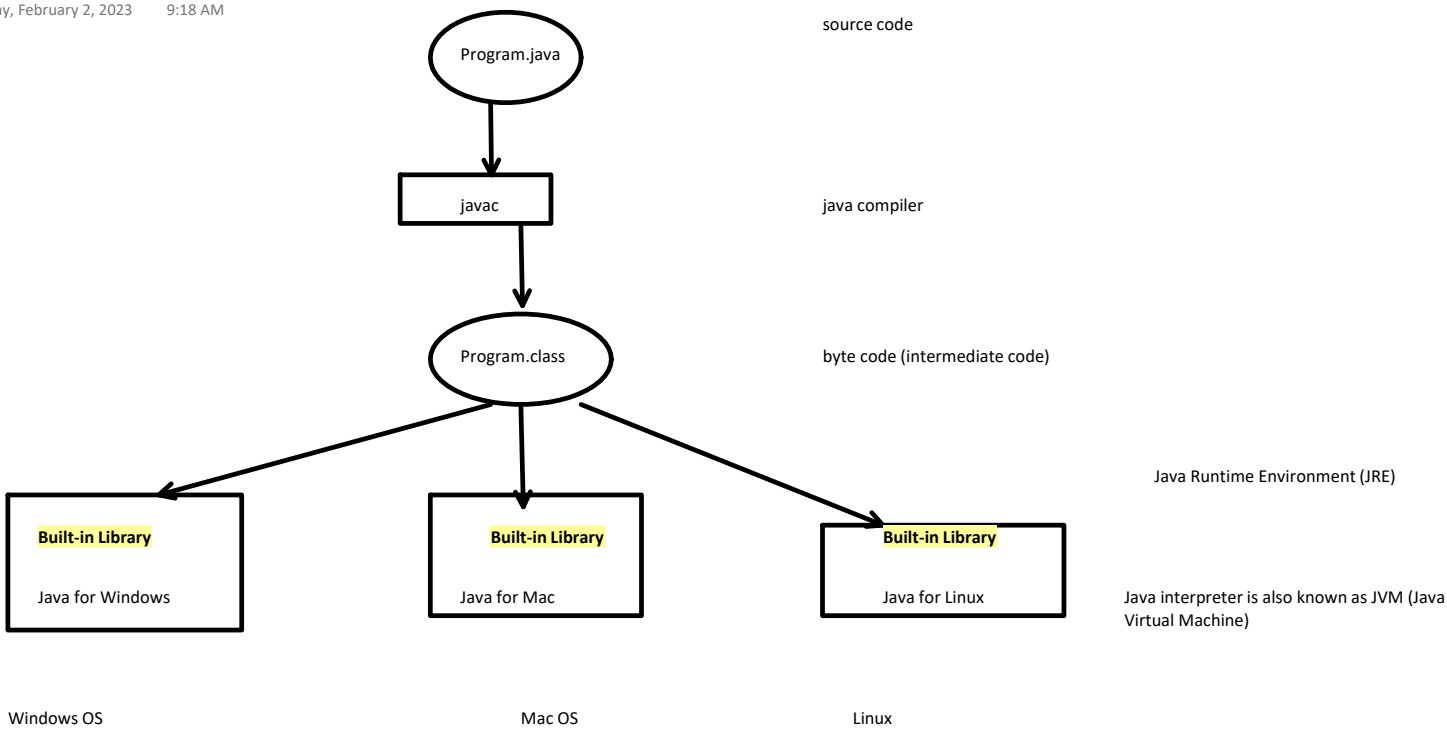
programming language.

Java Promise

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

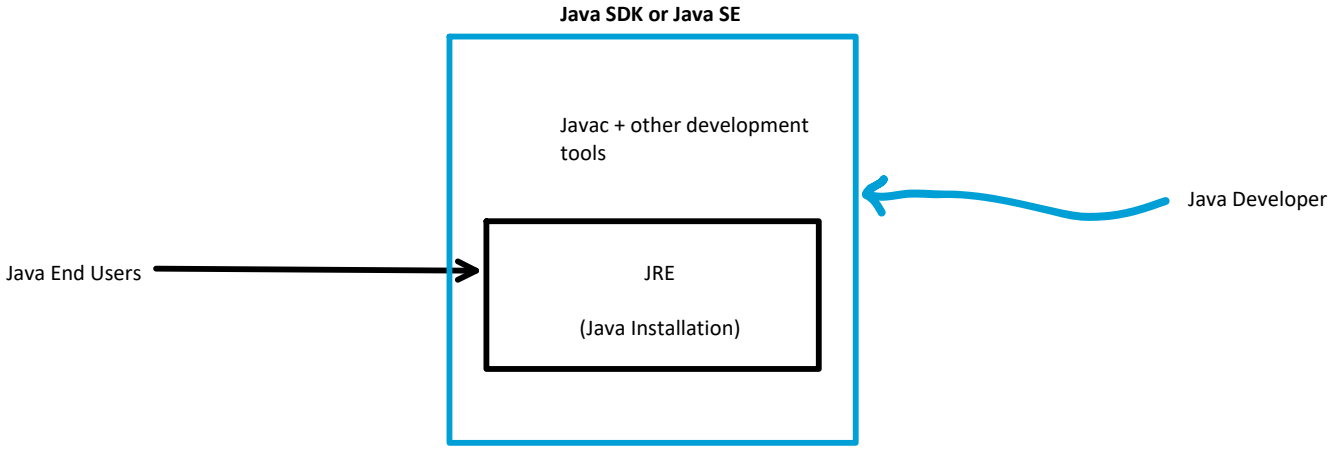
Java Program Flow

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

Hello.java - Notepad

File Edit View

```
class HelloWorld {

    public static void main(String [] args){
        System.out.println("Hello World");
    }

}
```

Ln 7, Col 2 | 100% | Windows (CRLF) | UTF-8

- 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

C:\Windows\System32\cmd.exe

```
D:\MyWorks\Corporate\202302-anz-java\java-demos\demo01>notepad Hello.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo01>javac Hello.java

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo01>dir
 Volume in drive D is Data
 Volume Serial Number is F8C2-FD66

 Directory of D:\MyWorks\Corporate\202302-anz-java\java-demos\demo01

02/02/2023  09:39 AM    <DIR>          .
02/02/2023  09:31 AM    <DIR>          ..
02/02/2023  09:38 AM               110 Hello.java
02/02/2023  09:39 AM               420 HelloWorld.class
               2 File(s)            530 bytes
               2 Dir(s)  140,183,535,616 bytes free

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo01>java HelloWorld
Hello World

D:\MyWorks\Corporate\202302-anz-java\java-demos\demo01>
```

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

Hello.java - Notepad

File Edit View

```
class HelloWorld {

    public static void main(String [] args){
        System.out.println("Hello World");
    }

}
```

Ln 7, Col 2 | 100% | Windows (CRLF) | UTF-8

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

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

Simple Arithmetic Program

Thursday, February 2, 2023 10:50 AM

- Write a program to calculate sum of two numbers

ArithmeticApp01.java - Notepad

File Edit View

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

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

        System.out.println(z);

    }
}
```

Ln 10, Col 25 | 100% | Windows (CRLF) | UTF-8

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
```

ArithmeticApp01.java - Notepad

File Edit View

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

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

        System.out.println(z);

        x=false;

        System.out.println(x);

    }
```

```
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
```

Compatible and Incompatible type

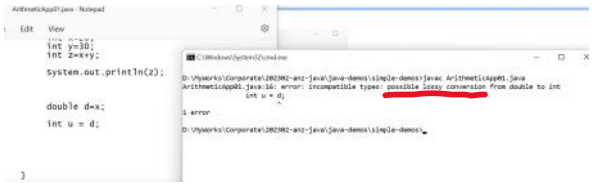
Data Types

- 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

- 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



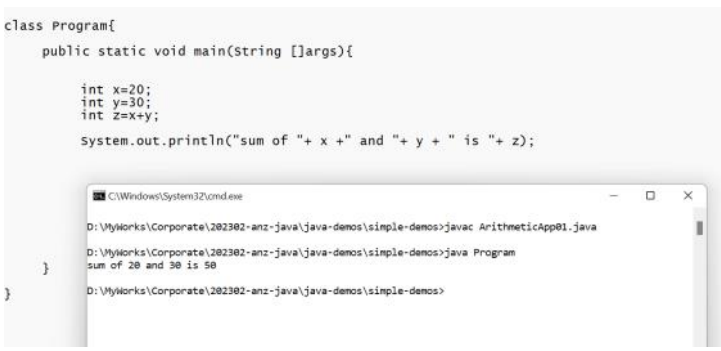
- we can force such conversion by explicit type casting

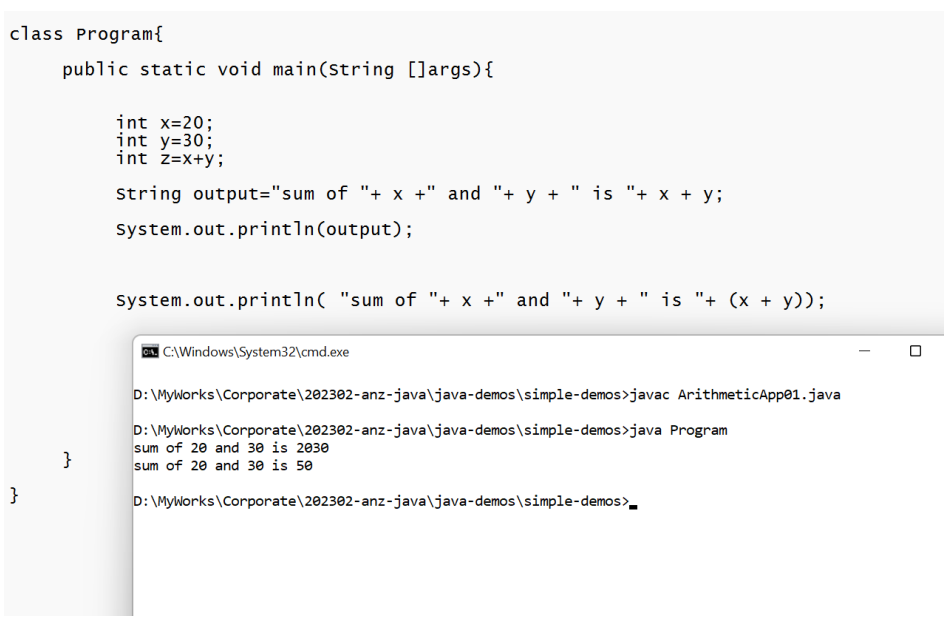
int u = (int) d; //force convert value of 'd' in int before assign

- 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





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
?:		

- 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

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

- 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

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

```
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

C:\Windows\System32\cmd.exe

```
D:\MyWorks\Corporate\202302-anz-java\java-demos\simple-demos>javac JavaMethod.java
JavaMethod.java:12: error: non-static method greet() cannot be referenced from a static context
                greet(); //call the other method
                ^
1 error

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

JavaMethod.java - Notepad

File Edit View

```
class Method01{

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

    public static void main(String []args){
        greet(); //call the other method
    }
}
```

- 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

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

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