JAVA + Selenium

1996 - Sun Micro systems

now - Oracle

Features of Java :

1. Simple

- syntax is based/similar to c++

- removed many confusing or rarely used concepts/features like explicit pointers , operator overloading

operators
+ - * / %

2 + 5 = 7

+ for power

2 + 5 = 32

- Automatic Garbage Collector

---

2. OO(Object Oriented) Programming

POP Procedure /Functional Programming languages

OOP is a methodology that simplifies s/w development and maintenance using some concepts/rules

- Object
- class
- Polymorphism
- Encapsulation
- Inheritance
- Abstraction

---

3. Robust

- Java uses Strong memory management
- No pointers ==> avoid security problems
- Automatic Garbage Collector (unwanted memory is deallocated)
- Exception Handling and Type Checking Mechanism

```
int i=10;

String j="FLM";

int k=i+j;  ==> error
```

---

4. Secure :

---

5. Platform Independent

OS

---

6. Architectural Netural

Architecture = Micro Processor + RAM

---

7. Portable

IBM

Apple

Solaris

How to Install jdk :

jdk 8 or jdk 11 (is preferred)

download below file

jdk-11.0.20_windows-x64_bin.exe

to check jdk installed or not
go to below loaction

C:\Program Files\Java\jdk-11

Configuration of jdk :

Environment Variables

JAVA_HOME

C:\Program Files\Java\jdk-11

path

C:\Program Files\Java\jdk-11\bin

```
C:\Users\Riyaz>java -version
java version "11.0.20" 2023-07-18 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.20+9-LTS-256)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.20+9-LTS-256, mixed mode)
```

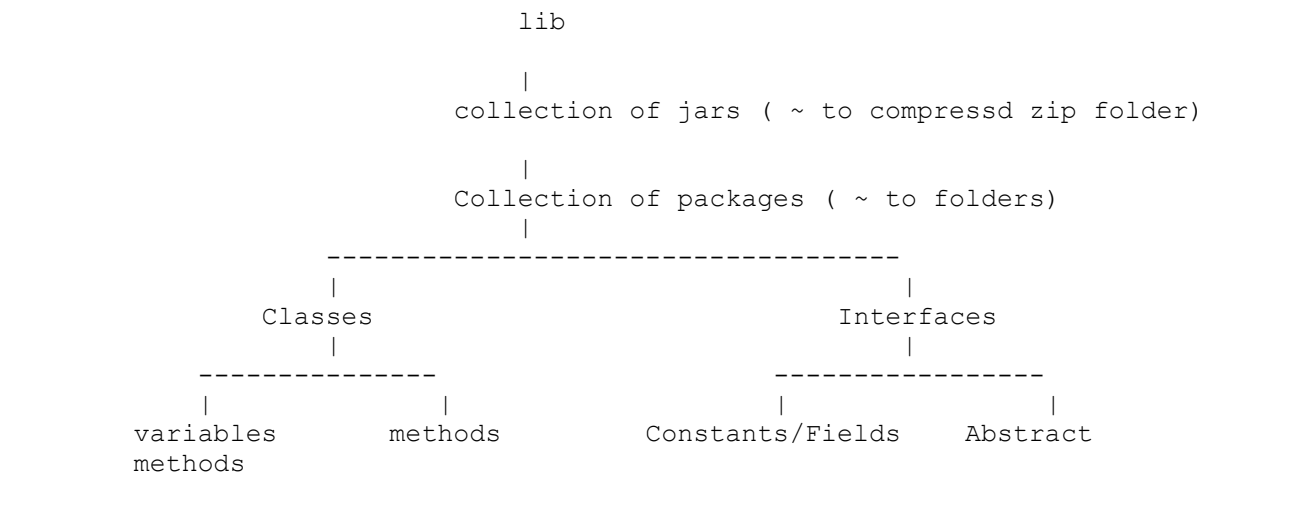
---

jdk

lib bin/tools jre

lib : Library

predefined stuff created by Java Communities



bin / tools :

to compile a java program

javac

syntax :

javac FileName.java

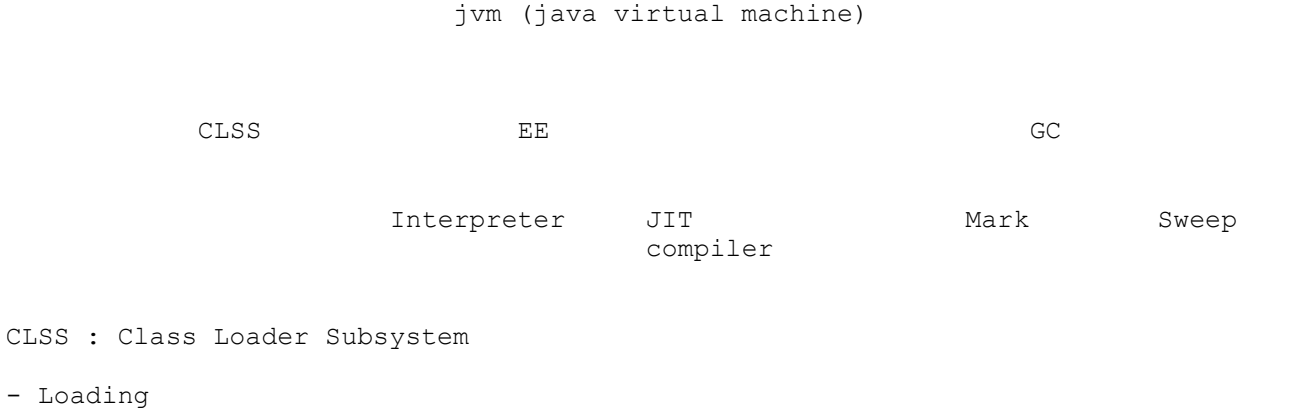
to excute a compiled java program :

java

syntax :

java FileName

JRE : Java Runtime Environment



CLSS : Class Loader Subsystem

- Loading
- Linking
- Initialization

EE (Execution Engine):

Interpreter : BC to MC
byte code to machine code

and also execute the code line by line

JIT (Just In time) compiler :

looping ==> to give performance
for , while , etc ..

Garbage Collector

for Memory deallocation

```
System.gc();
Runtime.gc();
```


JAVA Memory Blocks

1. Class Area / Method Area

- .class File (Programs)
- Static Variables
- Static Methods

2. Heap Area (dynamic memory allocation)

- Objects
- Arrays

3. Java Stack Area

- Local declarations / Local Variables
- expressions

```
int x=10,y=20;
```

x+y ==> 30

- current running logic /code

4. String constant pool Area (SCPA)

only string literals

```
String str="FLM";
```

Structure of Java Programs

4 Sections

1. Documentation Section

2. Package Section

3. Import Section

4. Class/Interface Section

---
1. Documentation Section
---

comments

single line comment :

```
// kjhkjhkjh
```

multiline comment :

```
/*  kjhkjhkjhkj

hkjhjhjkhkjh

kjgkjhkjhkjhkj */
```

---

2. Package Section

Pacakage is similar to folder

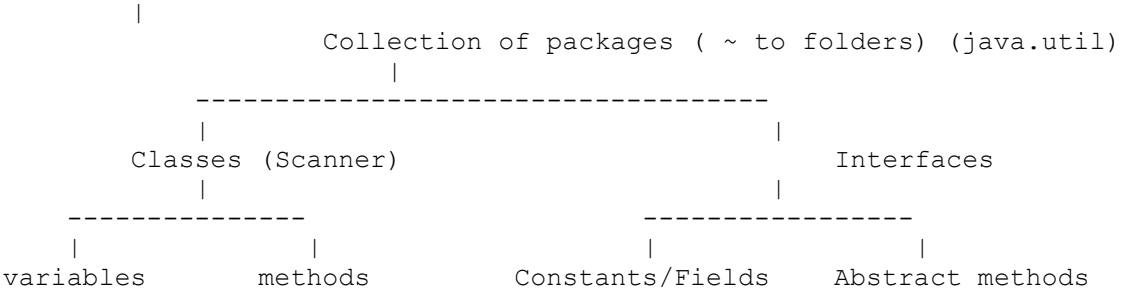
logical grouping

syntax :

```
package package-name;
```

---

3. Import Section




```
import packagename.*;  ==>not preferred
import packagename.ClassName; gives more readability
import static package-name.classname.*;
```

java.lang is a default package in java

there is no need of import

---

4. Class/Interface Section

```
access-specifier access-modifier class ClassName
{

//variables

//methods
```

```
}

access-specifier or access label or visibility


public / private /protected / <default>

access-modifier ==> behaviour

static or final or <default>


class Sample
{

access-specifier access-modifier return-type methodName(arguments or parameters)
public static void main(String[] args)
{
//local declarations
//Execuatbale statements


}

}


-------------------------------------
Java Naming Conventions
-------------------------------------


                    Collection of packages
                                |
                    ------------------------------------
                    |                                   |
                Classes (Scanner)                           Interfaces
                    |                                   |
            ---------------                     -----------------
            |              |                    |                |
        variables       methods         Constants/Fields    Abstract methods


packages : all lower case

package sep8th;

package java.util;

package regression;

package org.openqa.selenium;


---------------------------------

Class / Interface : Title Case


RamaKrishna  RK

Hello

VariablesDemo

Scanner

```

ChromeDriver

---

variables / methods / Abstract methods : camel case (LowerCase + Title case)

x y

height

netSalary

costOfItem

noOfLinks

Examples of methods :

methods (

nextInt(

getWindowHandle(

Constants/Fields : All upper case

PI=3.14

MAX_LIMIT=10000;

****

Variables in Java

****

Variables

Local Instance Static

-> refers to a memory block

10 + 20

i=10 j=20

i+j

-> simplifies the expressions

-> stores ones recall anywhere in program

syntax :

access-specifier access-modifier datatype variableName ; Variable Declaration

access-specifier access-modifier datatype variableName = value ; Variable Declaration and assignment

int i=10

Local Variable
---

1. Defintion :

a variable which is declared inside a method is called Local

2. When ?

when memory is getting allocated ?

when method is called

3. where ?

where these variables are getting

Java Stack Area

4. calling style

just variable name

5. scope or life time

till the end of method or within that method

Note :

for Local Variables , it is the responsbility of developer/programmer to initialize some value

```
VariablesDemo1.java:14: error: variable y might not have been initialized
System.out.println(y);
                   ^
```

1. Defintion :

2. When ?

when memory is getting allocated ?

3. where ?

where these variables are getting

4. calling style

5. scope or life time

Instance Variable or non-static variable
---
1. Defintion :

variables which are declared inside a class but outside of method

2. When ?

when memory is getting allocated ?

only when you create an instance / object of that class

how to create an instance or object ?

```
ClassName objectName=new ClassName();
```

3. where ?

where these variables are getting

in Heap Area

4. calling style

```
objectName.variableName;
```

5. scope or life time

till the last usage of object in the program

Static variable

---

1. Defintion :

variables which are declared inside a class but outside of method having keyword static

2. When ?

when memory is getting allocated ?

during class loading

3. where ?

where these variables are getting

Class Area /Method Area

4. calling style

```
ClassName.variableName;
```

5. scope or life time

till the end of program

Data Types in Java :

Java is Strongly typed language

1996

```
                8 4 2 1

3     0 0 0 0   0 0 1 1

      0 0 0 0   0 0 0 0     0 0 0 0     0 0 1 1




                                Data Types in Java


                primitive   (8) `
                non-primitive

boolean
char
byte
short
int
long
float
double



How to download eclipse ?

eclipse is an IDE (Integrated Development Environment)

IDE's like eclipse , Intelli J Idea , Vs code , net beans etc
..

https://www.eclipse.org/downloads/

Click on "Download Packages " link

under "MORE DOWNLOADS"

click on "Eclipse 2021-09 (4.21)"

under "Eclipse IDE for Enterprise Java and Web Developers"

click on "Windows x86_64"




Data Types in Java


Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Type mismatch: cannot convert from int to byte
    Type mismatch: cannot convert from int to short

    at sep13th.DataTypesDemo1.main(DataTypesDemo1.java:9)

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    The literal 2147483648 of type int is out of range

    at sep13th.DataTypesDemo1.main(DataTypesDemo1.java:13)
---------------------------------------------
Methods in Java
---------------------------------------------

Method :

a set of statements or instructions

```

need of methods :

- resusabilty
- reduces code redundancy
- increases code clarity
- code debugging becomes easy

syntax :

```
access-specifier access-modifier return-type methodName(parameters or arguments)
{
//local declarations
//statements

return statement ; if return-type is void no need of return satement

}
```

Methods in Java

Instance or Non-static

Static

Non-Static Methods

1. Defintion :

method defined or declared without static keyword

<default> , abstract , transient , volatile etc ..

2. When ?

when memory is getting allocated ?

creation of object

3. where ?

where these variables are getting

Heap Area

4. calling style

objectName.methodName();

5. scope or life time

till last usage of object

??????????????????????????????

Write a Java Program to calcuate Simple Interest

using methods concept

??????????????????????????????

Static :

1. Defintion :

method defined or declared with static keyword

2. When ?

when memory is getting allocated ?

during .class file loading

3. where ?

where these variables are getting

Class Area /Method Area

4. calling style

ClassName.methodName();

5. scope or life time

till the end of program

==========

OOP Object Oriented Programming

POP Procedure oriented programing (c , COBOL etc)

Object

Class

Encapsulation

Polymorphism

Inheritance

Abstraction

Object :

anything that exits is an object

3 characteristics :

State : data or attributes that represents an object

variables

Behaviour : actions or tasks that object can perform

methods

Identity : unique id used by JVM to identify object

Class

blueprint or template from which you can create an object

Object Creation Technique

---

1. using new operator

```
ClassName objectName=new ClassName();
```

2. using static factory method

```
ClassName objectName=ClassName.methodName();
```

3. using non-static factory method

will be used in case of dependency of a object of some class on another class

ClassName1
    methodName : object of ClassName2

ClassName2

```
ClassName1 object1=new ClassName1();
ClassName2 object2=object1.methodName();
```

XSSFWorkbook

    getSheet(): object of XSSFSheet

XSSFSheet

```
XSSFWorkbook wb=new XSSFWorkbook();
XSSFSheet ws=wb.getSheet();
```

4. using clone method

5. using String Literals

```
String s1="FLM";
```

6. using deserialization
 (no syntax)
 used by JVM to create object

Encapsulation :

Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.

Encapsulation = Data hiding + Abstraction

The field Bank.balance is not visible

For Tightly Encapsulation , we can use access-specifier as private

if you make variables private only that class method can use it

then if you want to access the private variables of a class ,
create a public methods in that class and retrive them using proper validation

```
//getters

methods having prefix get

these are used to retrive data or variables


//setters

methods having prefix set

these are used to modify or alter data or variables
--------------------------------
this keyword
--------------------------------

if we have local varibales same as intance variables then local variables dominates
 to overcome this problem , we have keyword "this"  in java

 this :

 is keyword cum operator which is used for current calling object reference

***********************************************
Java Season - 1
Day 10  - 16th September  - Saturday
***********************************************

------------------
Polymorphism
------------------

Poly means many

morph means forms

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    The method add(int, int) in the type Sum is not applicable for the arguments
    (double, double)

    at sep16th.SumDemo.main(SumDemo.java:12)

The method add(int, int) in the type Sum is not applicable for the arguments (long, long)

---------------------
without Polymorphism

public void add(int x,int y)
    {
        System.out.println("Sum is "+(x+y));
    }

    public void addDouble(double x,double y)
    {
        System.out.println("Sum is "+(x+y));
    }

    public void addLong(long x,long y)
    {
        System.out.println("Sum is "+(x+y));
    }

---------------------------------------

with polymorphism

    public void add(int x,int y)
    {
        System.out.println("Sum is "+(x+y));
```

```
    }

    public void add(double x,double y)
    {
        System.out.println("Sum is "+(x+y));
    }

    public void add(long x,long y)
    {
        System.out.println("Sum is "+(x+y));
    }
```

Polymorphism

Static	Dynamic
or	or
Compile time	Runtime
or	or
Early Binding	Late Binding

binding means mapping or linking

the caller and called method ,

In Static polymorphism or method overloading

binding will happen during compile time ,

Static Polymorphism :

can be achived by means by

Operator Overloading (not encourage or not supprorted in java)

Method Overloading

Dynamic Polymorphism :

can be achived by means by

method overiding (this will be discussed after inhertance topic)

Method Overloading

---

method signature

methodname + Paramaters / arguments

in Method Overloading in same class

we will have same method name and with different signatures

Method Overloading -- different data types

-- change in no of arguments

-- change in order of arguments (not prefered)

****
PrintStream Class ==> java.io

4 print methods

println() : this will print and takes the cursor to new line

print(): this will print and stays cursor in same line

```
*
* *
* * *
* * * *
* * * * *
```

```
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

printf() : from previous lang
printf(String, Object...)

format()

---

git repo for programs

git repo for programs

https://github.com/flm31stjuly/JavaSeasons

??

use method overloading concept for below requirment

2,10,1869 - > 02-10-1869

1,1,500 -> 01-01-0500

??

Constructor :

def : it's method having same name as class name

rules of constructor :

1) Constructor name should be same class name (case senstive)
2) Constructor shouldn't have return-type
if return-type is mentioned .then what ?
then comlpiler will treat as normal method
3) Constructor shouldn't have access-modifier
if access-modifier is specified , then compilation error

Illegal modifier for the constructor in type Sample; only public, protected & private are permitted

Constructor

default constructor parameterized constructor

Constructor Overloading :

3.Inheritance
---

class B extends A

1. Single or Simple
2. MultiLevel
3. Hiererichal
4. Multiple
5. Hybrid

Marks extends Student

child	Parent
derived	base

Java doesn't allow Multiple Inheritance for classes

```
public class C extends A,B{

}
```

Method Overiding
---

method signature

methodname + Paramaters / arguments

methods having same signature in two different classes having relationship

****
Java Season - 1
Day 14   - 23rd September - Saturday
****

---
4. Abstraction

Encapsulation = Data Hiding + Abstraction

Abstraction

Abstract class	Interface
(0 to 100 %)	100 %

abstract methods :

syntax :

access-specifier abstract return-type methodName(arguments); ==> idea and no implementation

Note :

for abstract class we Cannot instantiate/create objects

Interface : (100 % abstraction)

constant / fields and abstract methods

Methods :

1) what is the arguments that method is taking
2) what is the return-type
3) what does this method do ?

operators:

special symbols and it peforms particular operation

based on no of operands ,

operators

unary binary ternary

operand can be a value or variable

```
int x=2;
int y=3;

z = x+y
```

2 + 3

2,3 operands
+ operator

unary :

++ --

post pre

inc dec inc dec

a++ a-- ++a --a

```
int x=20;

x++;

x--;

++x;

--x;
```

Binary operators :

- Aritmetic operator / Mathematical
- Realtional Operators
- Logical Operators
- Bitwise Operators
- assignment operators

- Aritmetic operator / Mathematical :

+ - * / %

- Realtional Operators

comparision between two things

> < >= <= != == ==> true or false

Logical Operators:

b/n two comparisions

&& || !

AND OR NOT

Bitwise Operators:

&	|	~	^	>>	<<	>>>
AND	OR	tilt	XOR	Right shift	left shift	unary right shifter

Assignment :

= += -= *= /= %= >>= <<=

```
x=10;
```

```
if x= 10
x+=10;    => x=x+10;


x=12;   1100

y=4;    0100

---------------
        1100
```

Scanner Class :

Scanner(InputStream)

here InputStream object is nothing but System.in

nextInt():int

nextLong() : long

nextShort() : short

nextByte() : short

nextDouble() : double

nextFloat(): float

nextBoolean() : boolean

next() : String

Ternary operators : (conditional operators or decision making operations)

?:

syntax :

expr1 ? expr2 : expr 3

expr1 condition or comparision

expr2 True part

expr3 False part

Flow Control or Control flow in Java :

if else

```
if(boolean value or an expression which results in boolean or condition)
statement1; // will be excuted when condtion is true
else
```

```
statement2; // will be excuted when condtion is false

if(boolean value or an expression which results in boolean or condition)
{

//block of statements
}
else
{
//block of statements
}


https://www.facebook.com/


    // <= 30000 20%
    // >30000 <=60000 30%
    // >60000 40%


Non-IT   IT


Switch:


syntax


switch(variable/expression)
{

case value1 : statements
              break;
case value2 : statements
              break;


default : statements;


}


byte  short   int  char  String (from 1.7)


Not allowed : long  float double boolean


Loops or Iterative statements



i) for looping

syntax :


for(counter intialization; condition check ; increment or decrement)
{

//statements or code


}
```

ii) while loop

```
while(condition check)
{

//statements or code

}
```

iii) do while

iv)for each or enchanced for loop

syntax :

```
for(dataype someReferenceVaribale: arrayName or Collection refernce)
{

//use the someReferenceVaribale means the value
}
```

Arrays :

```
int x=10;

int y=20;
```

def : indexed collection of homogenous datatype elements having fixed length

syntax :

method 1:

```
datatype[] arrayName={value1,value2,value3..valuen}
```

datatype[] arrayName ==> declaration

arrayName={value1,value2,value3..valuen} ==> initialization

length : int (this is not method)

method 2:

using new

```
datatype[] arrayName=new datatype[length of array or no of elements in array];
```

****
Java Season - 2
Day   - 5th October - Thursday
****

```
datatype[] arrayName=new datatype[length of array or no of elements in array];
```

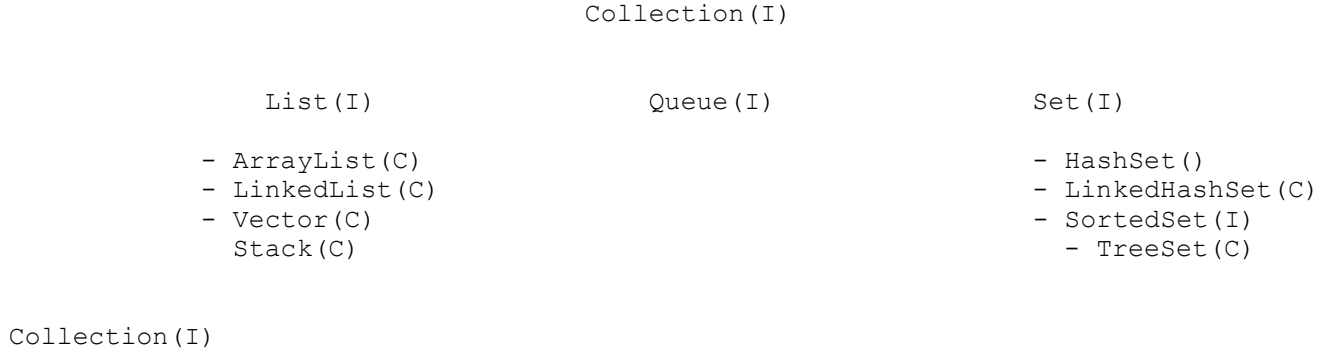
dis advantages :

1. arrays always allow similar datatype
2. arrays doesn't have growable nature

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of
bounds for length 5
    at oct5th.ArraysDemo5.main(ArraysDemo5.java:15)
```

3. data strructured related predefined methods are not available

Collections :



Collection(I)

```
add(E):boolean
addAll(Collection<? extends E>):boolean
contains(Object):boolean
isEmpty():boolean
size():int
```

List(I):

```
add(int, E):void
get(int):E
indexOf(Object):int
toArray():Object[]
```

ArrayList(C)   :

```
ArrayList()
ArrayList(int)
ArrayList(Collection<? extends E>)

subList(int, int):List<E>
```

```
***********************************************
Java Season - 2
Day   - 56h October  - Friday
***********************************************
```

Iterator :

```
iterator():Iterator<E>
```

2 methods :

```
hasNext():boolean

next():E
```

this will do 2 things

- retrive the element
- moves the cursor to next level

Set(I):

No get method here , because insertion order(index) is not preserved

Constructors:

HashSet()
HashSet(int)
HashSet(int, float)
HashSet(int, float, boolean)
HashSet(Collection<? extends E>)

Here float is meant for Load factor

List (ArrayList	vs	Set HashSet
- insertion order is preserved means index		- insertion order is not preserved because of hashing technique
- duplicates are allowed		- duplicates are not allowed
internal structure or implementation - linked list		- HashMap
to retrive the elements - get(index)		we shoud use for-each or iterator

---

Exception Handling :

Runtime Error is called Exception .

try catch finally throw throws

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at oct7th.ExceptionDemo1.main(ExceptionDemo1.java:15)
```

10

```
try
{

// risky code

}
catch(argument)  //here argument is an exceptionclass with reference
{

//handling code or info code
}
finally
{
```

```
// some code to be excuted irrespective of exception
//compulsory

}


try finally

try catch

try catch finally

try catch catch ....

try catch catch ....finally


                                Exception Types


            1)Checked Exceptions                2)UnChecked Exceptions



MEthods in Throwable or Exception Class :


getMessage():String

printStackTrace():void


Note : while using multiple catch blocks you should follow bottom to top  hiererchy

Unreachable catch block for ArithmeticException. It is already handled by the catch
block for RuntimeException

---------------------------------------

throw


used to create and throw user-defined or custom exception

https://github.com/flm31stjuly




break statement:

used in loops , switch

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Unreachable code

    at oct9th.BreakDemo2.main(BreakDemo2.java:16)


continue statement :

use in loop
```