Core Java

* Data types
  + primitive DT
  + derived DT
* Variables (data)
* methods (behaviour)
* constructor
* static Blocks
* Instance Blocks
* Classes
* Interfaces in Java
* Access specifiers
  + private
  + default
  + protected
  + public
* Access modifiers
  + static
  + abstract
  + synchronized
  + final
* Strings in Java
  + functions
  + StringBuffer
  + StringBuilder
* Threads
* Exception Handling
  + try
  + catch
  + throw
  + throws
  + finally
  + User Defined Exception
* Files
  + properties , text file, excel file
  + Read a file
  + Write to a file
* Loops
  + for
  + foreach
  + while
  + do,while
* conditional Statements
  + if
  + if else
  + if elseif else
  + switch
* Abstract Classes
* Wrapper Classes
* Arrays
* Collections
  + List
  + Set
  + Map
* Generics

Automation Testing :

* What is Software Testing
* What are the drawbacks of Manual testing
* What is regression Testing
* What are the different tools available for automation testing
* Why Selenium is popular ???
* Selenium 4.0
* Selenium IDE
  + Record and Playback
* ~~Selenium RC~~
* Selenium WebDriver
  + Object identification
    - id
    - name
    - classname
    - tagname
    - linktext
    - partiallinktext
    - css
    - xpath
      * Basic xpath
      * xpath using Logical Operator
      * xpath using function
      * traversing from parent to child
      * traversing from child to parent
      * xpath using axes functions
        + Traversing to previous siblings
        + Traversing to next siblings
        + Traversing to parent
        + Traversing to child
        + Traversing to ancestor
  + How to play with textbox, radio, checkbox, hyperlinks, images
  + How to play with web table, dynamic elements
  + How to perform validation
  + Select Class
  + Actions Class – KB and Mouse operation
  + End to End Automation
  + Data driven Testing
  + Popups
    - alert
    - confirmation
    - hidden division
    - file download
    - file upload
* AutoIT
* Selenium Grid
* Frameworks
  + ~~Function Driven Automation framework~~
  + ~~Keyword Driven Automation framework~~
  + ~~Hybrid Driven Automation framework~~
  + Page Object Model
  + TestNG
    - annotations

Build Automation Tools :

* Maven
* gradle

Jenkins :

GIT :

Interview Questions : - On the topics covered

2 Sample Applications :

Installations :

1. Java - JDK 1.8
   1. <https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html>
      1. JDK – Java development kit
      2. JRE- Java Runtiime Environment
   2. Open command prompt
   3. java –version ( ON any error set the Environment variables )
2. Editor : Visual Studio Code, **Eclipse**, IntelliJ, ...
   1. https://www.eclipse.org/downloads/packages/release/2021-09/r/eclipse-ide-java-developers
3. GIT
   1. <https://git-scm.com/downloads>
   2. Configuration
      1. signup to gitbub and create your own account
      2. update user name and email in the git commandline tool
         1. git config --global user.name "<<USERNAME>>"
         2. git config --global user.email <<EMAIL>>
   3. Create a Repository in your machine
      1. git init
      2. create some files
      3. add the file to version control -> git add file1, file2......filen OR git add .
      4. git commit –m “Message ” 🡪 this will generate commit id
      5. git push origin master OR
      6. git push --set-upstream origin master

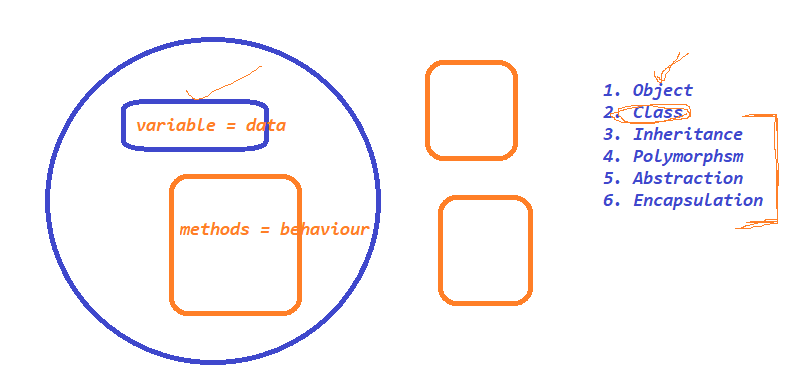
NOTE : link your local repository with the global repository by executing

git remote add origin <<Remote Repository URL>>

JAVA :

­

OOPS Concept :



­­­

Features of JAVA :

1. Simple

2. Object oriented Programming language

3. Platform Independent

4. Secure

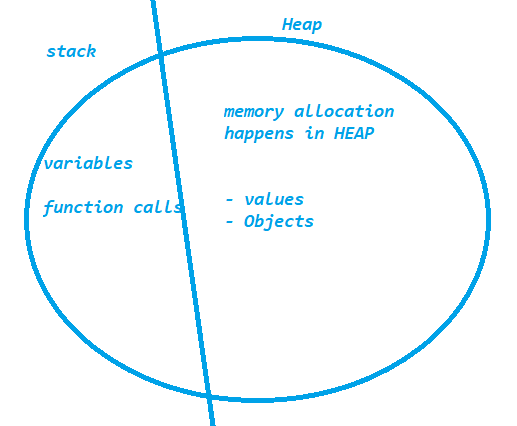
5. Robust

6.Architecture-neutral

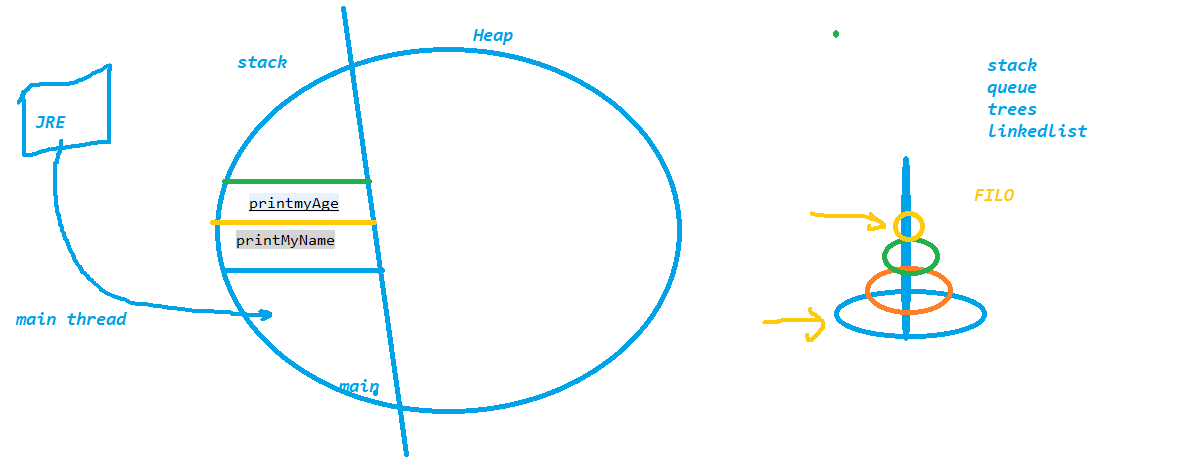
7. Portable

8. Multi-Threading

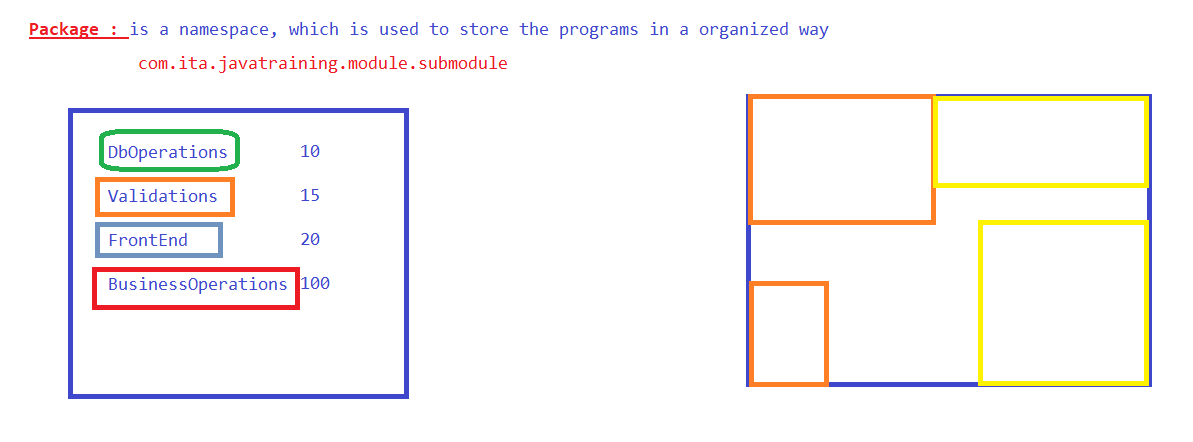
HelloWorld JAVA Program



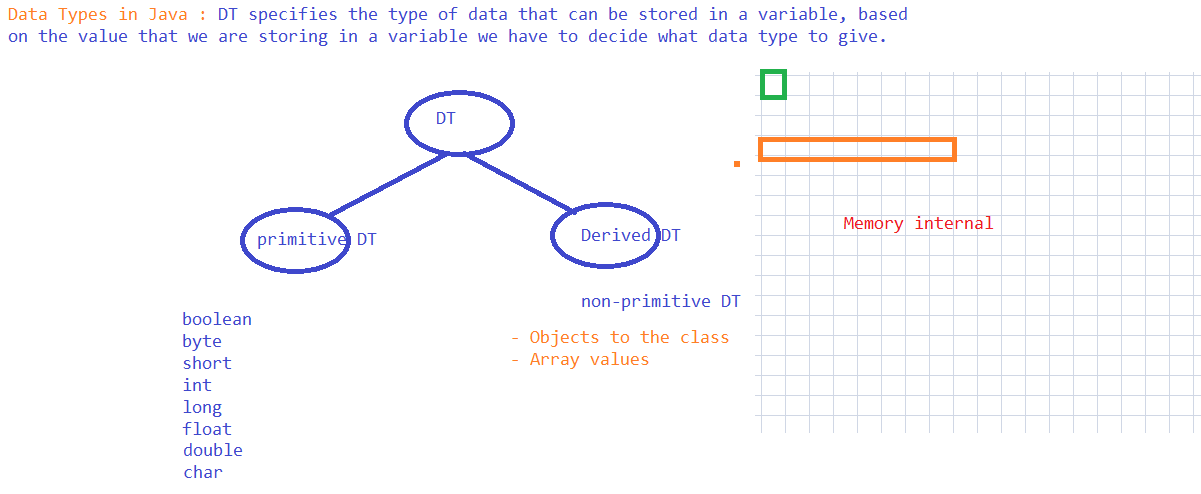
Stack Trace :

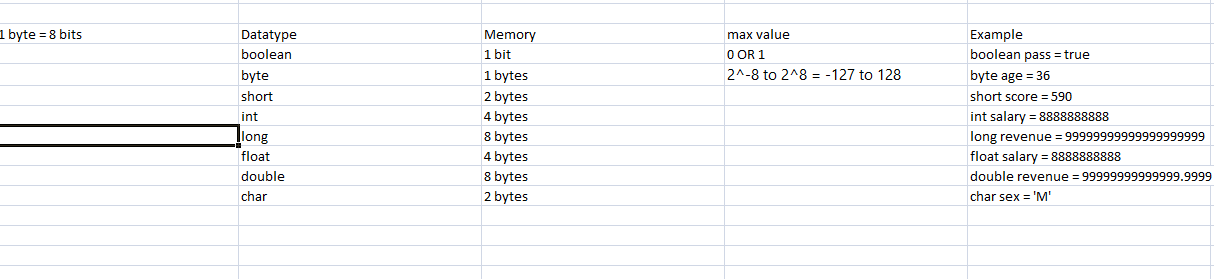


Packages in java :

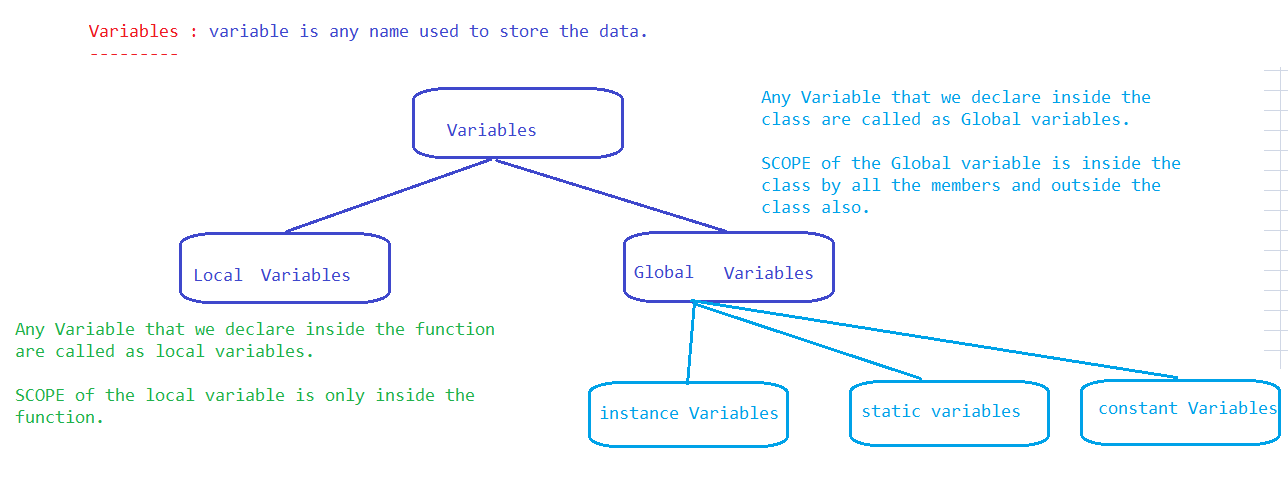


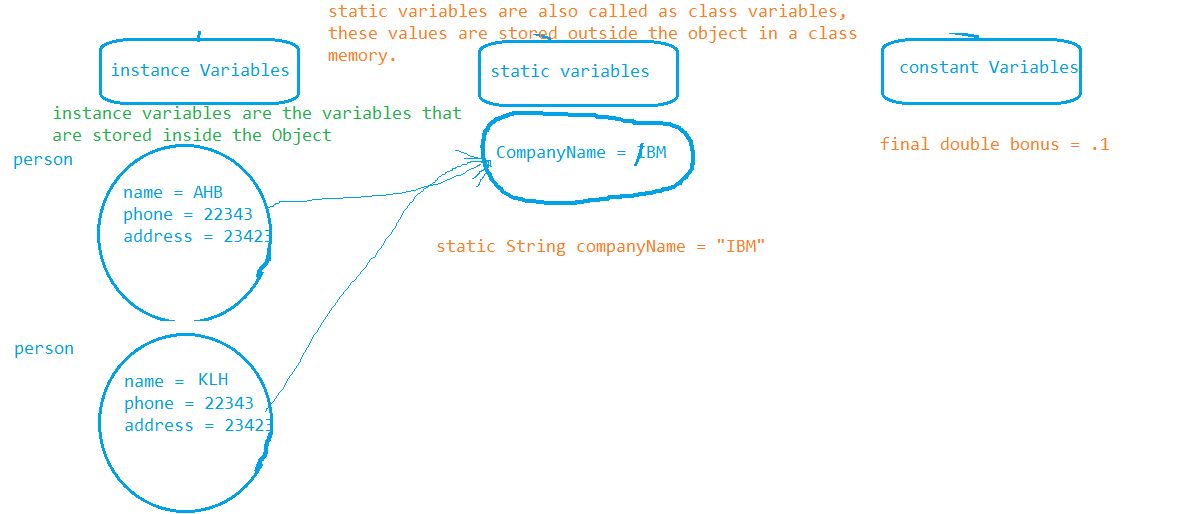
Data Types in JAVA :



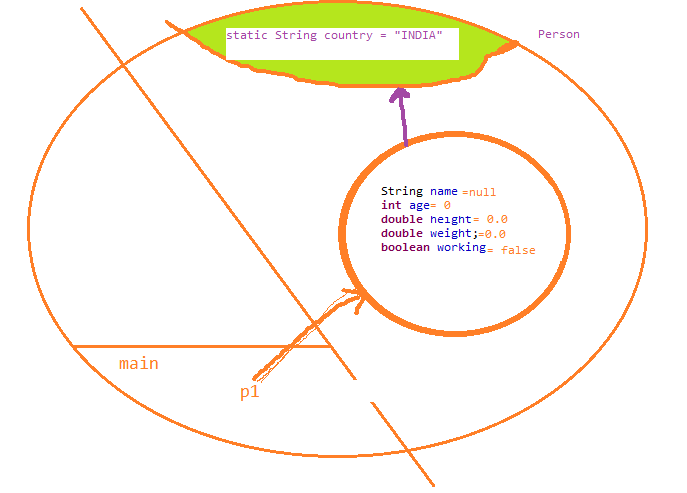


Variables :

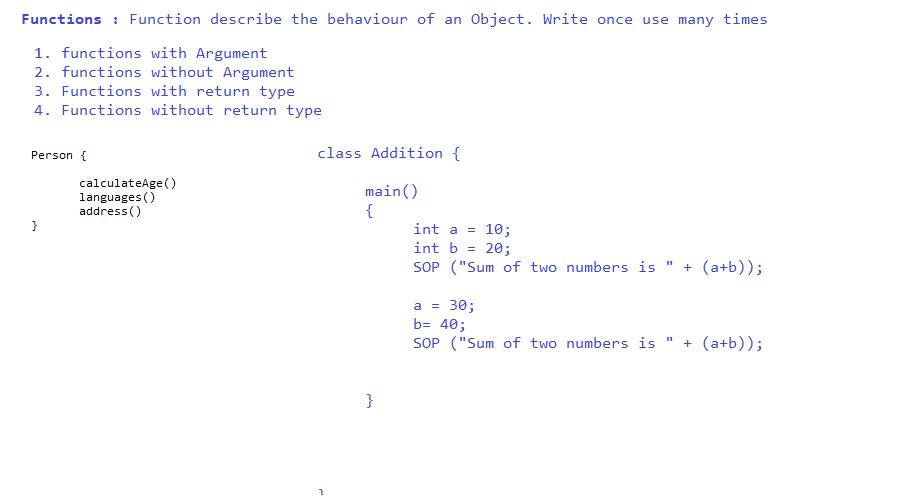


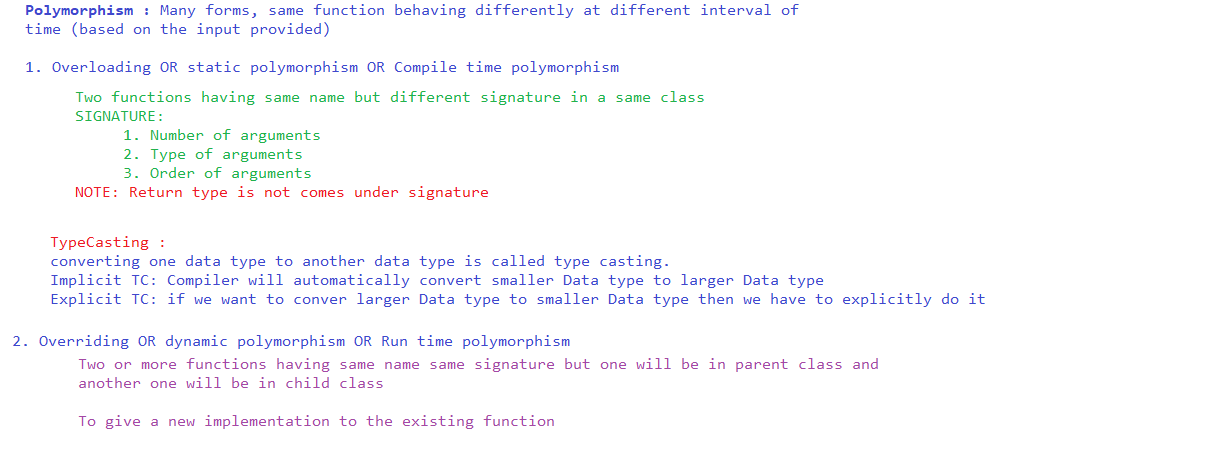


static variables demo :

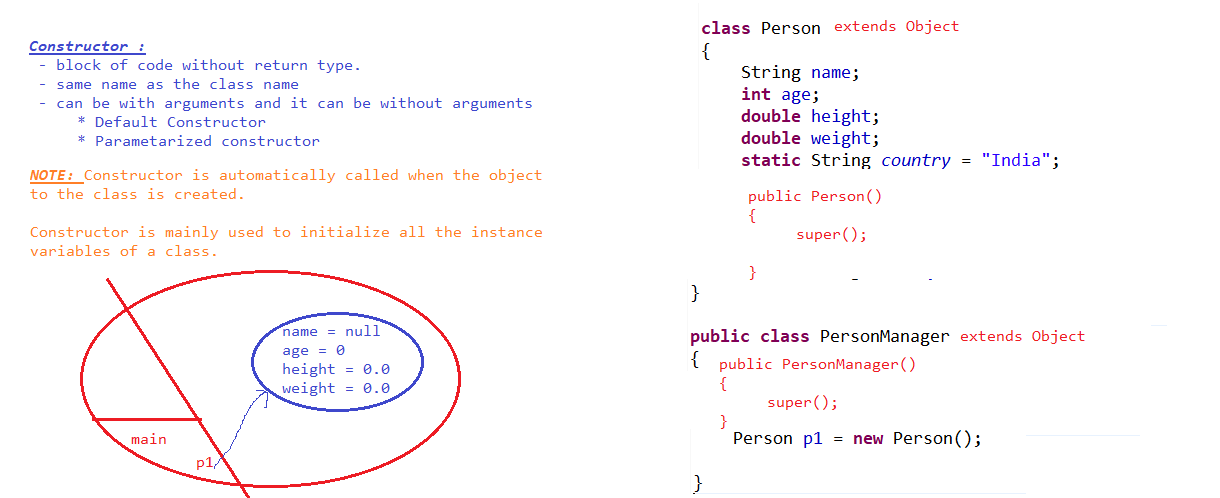


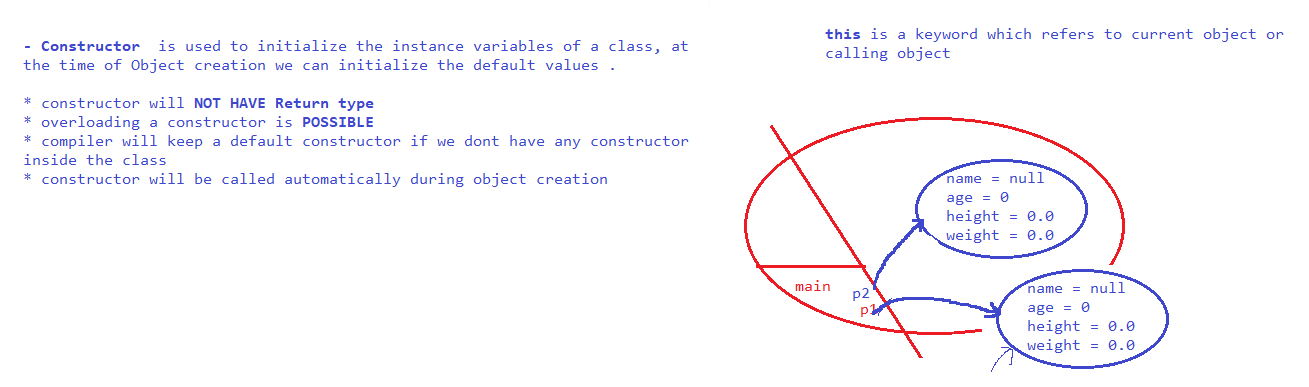
**Functions OR methods**

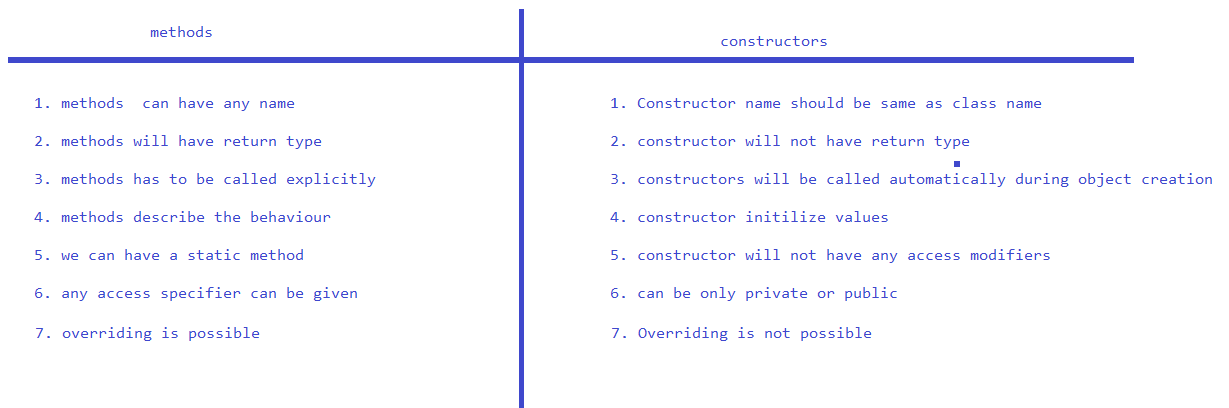
****



Constructors :







Overriding :

with toString method.

Static initialization Block

static

{

}

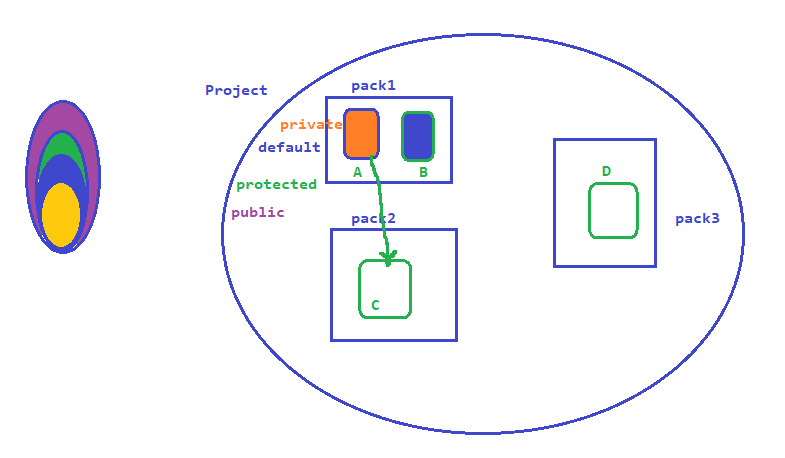
Instance initialization Block

{

}

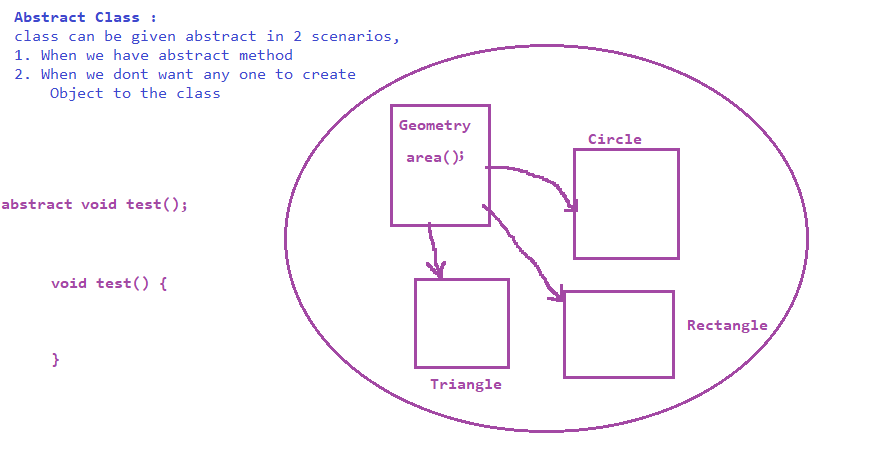
Access Specifiers

* private - private members are accessible only inside the class by all other members
* default – default members are accessible by all the classes within the same package
* protected – protected members are accessible by all the classes within the same package, outside the package it is accessible only if there is a inherited class (child class )
* public - wider access, it can be accessible by all the classes with in the project



Access modifiers

* static – class member, without creating a object we can access these members
* final –
  + variables : it acts as a constant value, no one can change / alter the value
  + methods : we cannot override
  + class : we cannot make it as a parent class
* abstract

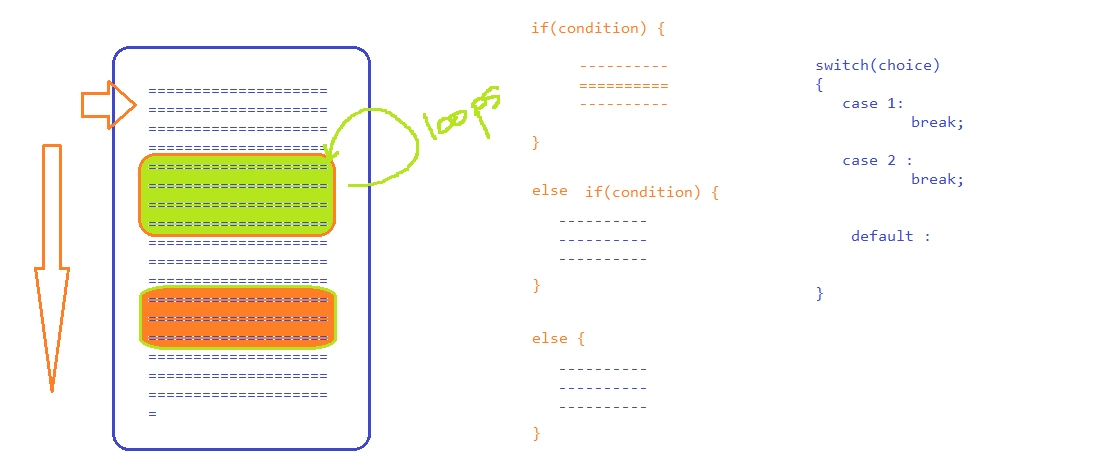


* + methods: when there is no method definition we declare method as abstract
  + class :
    - when we have any abstract method
    - when we want to restrict creating objects to the class
* synchronized –TBD

Looping and Conditional Statements :

* for
* while
* do,while
* foreach / Extended for loop

Conditional statements



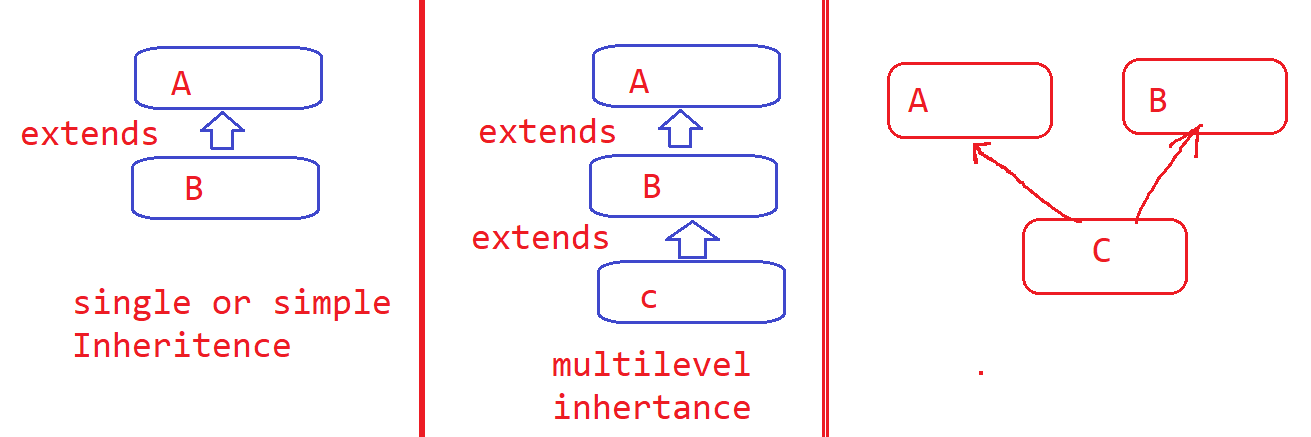
Strings :

- String functions

- String Buffer

- String Builder

Inheritance :



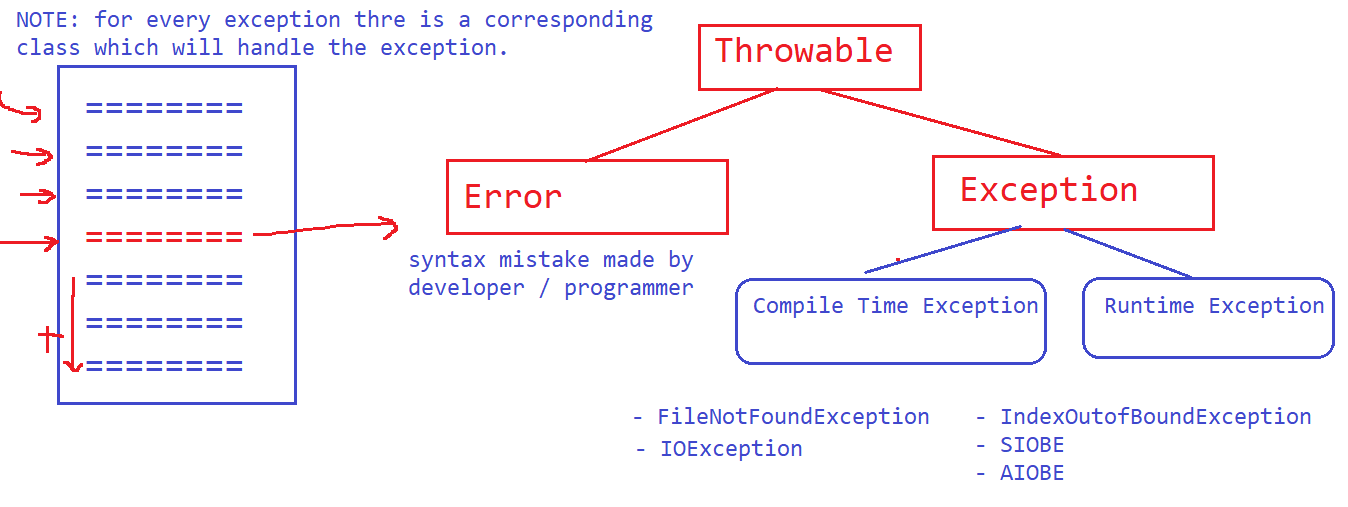
To achieve multiple inheritance we use interfaces.

NOTE : Interfaces are similar to classes, but interfaces are 100% pure abstract class.

Interfaces:

whatever method we declare inside interface is abstract by default and whatever the variable we declare inside the public static final by default

Exception Handling



Exception Handling is a mechanism using which we can ask compiler to continue the execution till the last line even though there is a abnormal statement.

To handle exception we have to use try block and catch block

* try : abnormal statements can be kept inside try block
* catch : corresponding class which can handle the exception .
  + for one try we can keep multiple catch blocks
  + there should not be any valid java code between try and catch
* throws : whenever we don’t want to handle the exception from the place where it has occurred then we use thows. using throws, exception will go to the called place (Function call)
* throw : when we want to raise exception explicitly
* finally : to execute all the time ( exception raised or not raised )

File Handling in JAVA :

OPS Concept :