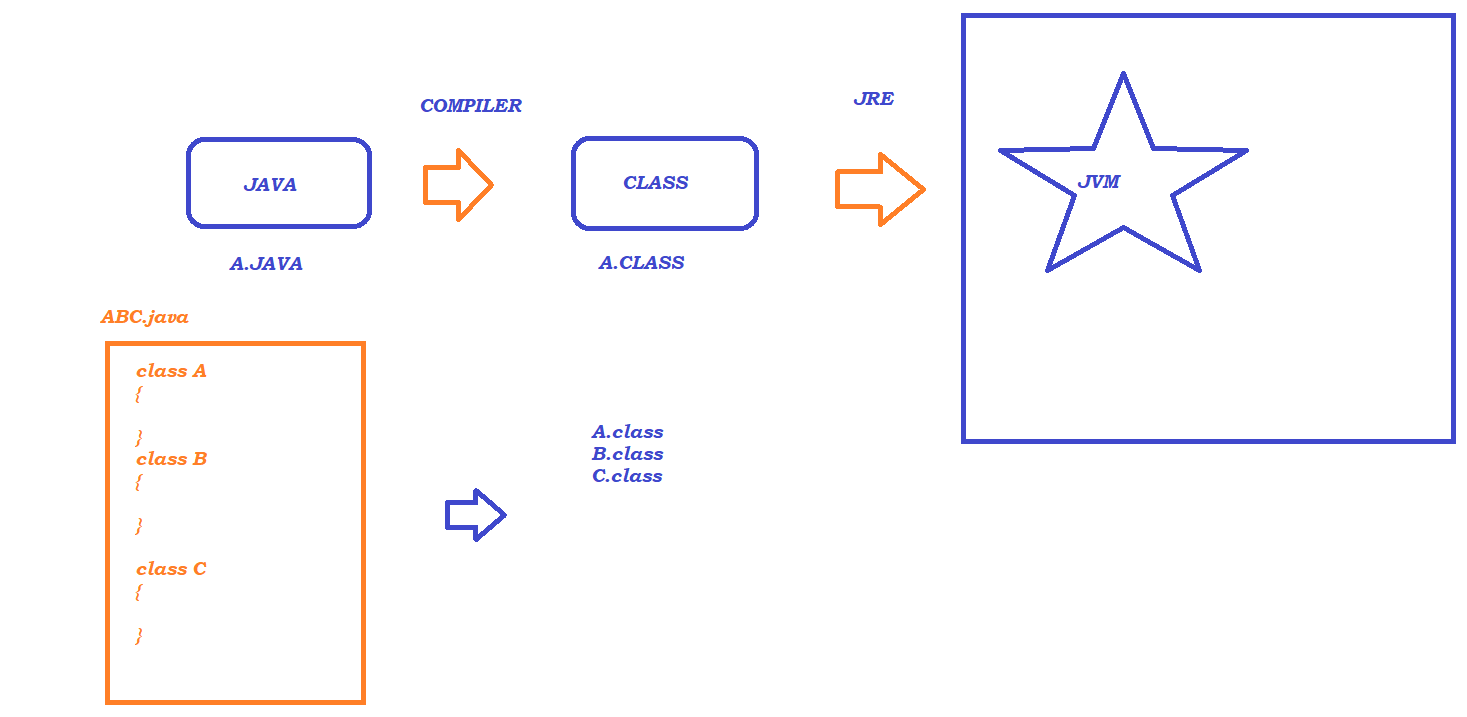
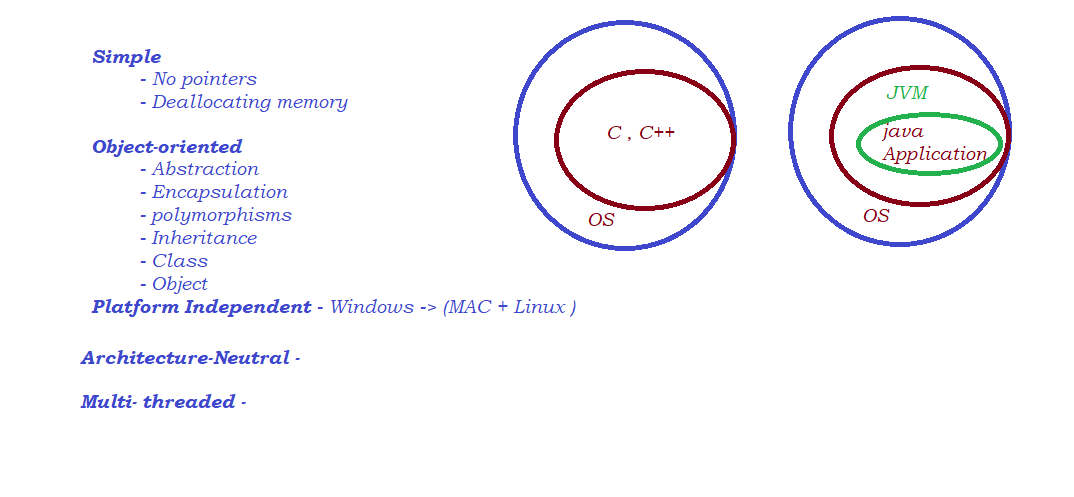
Java :

Execution –

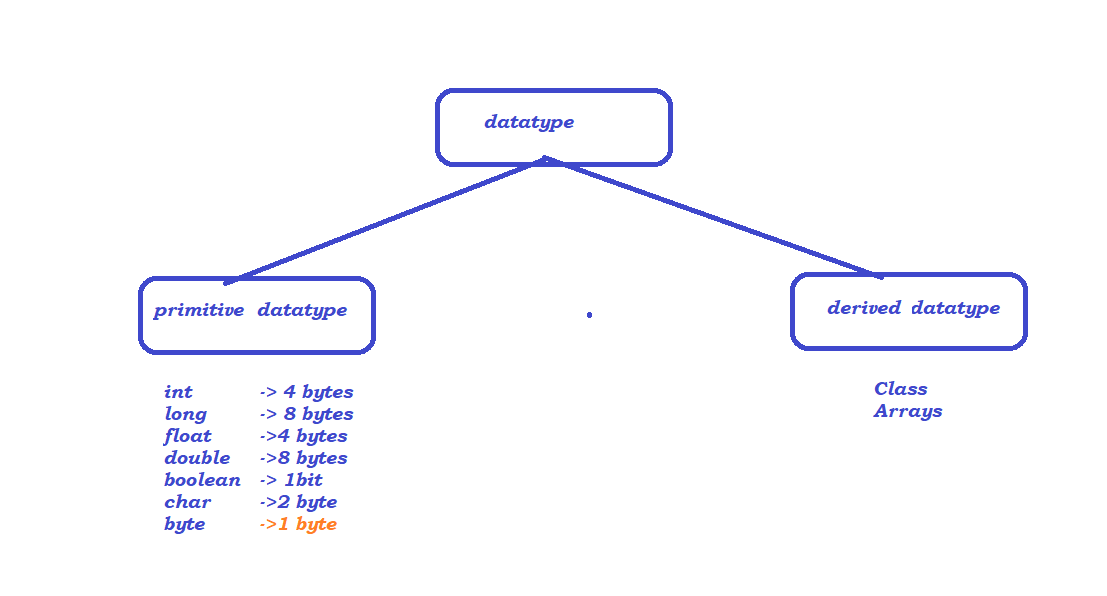


Features :

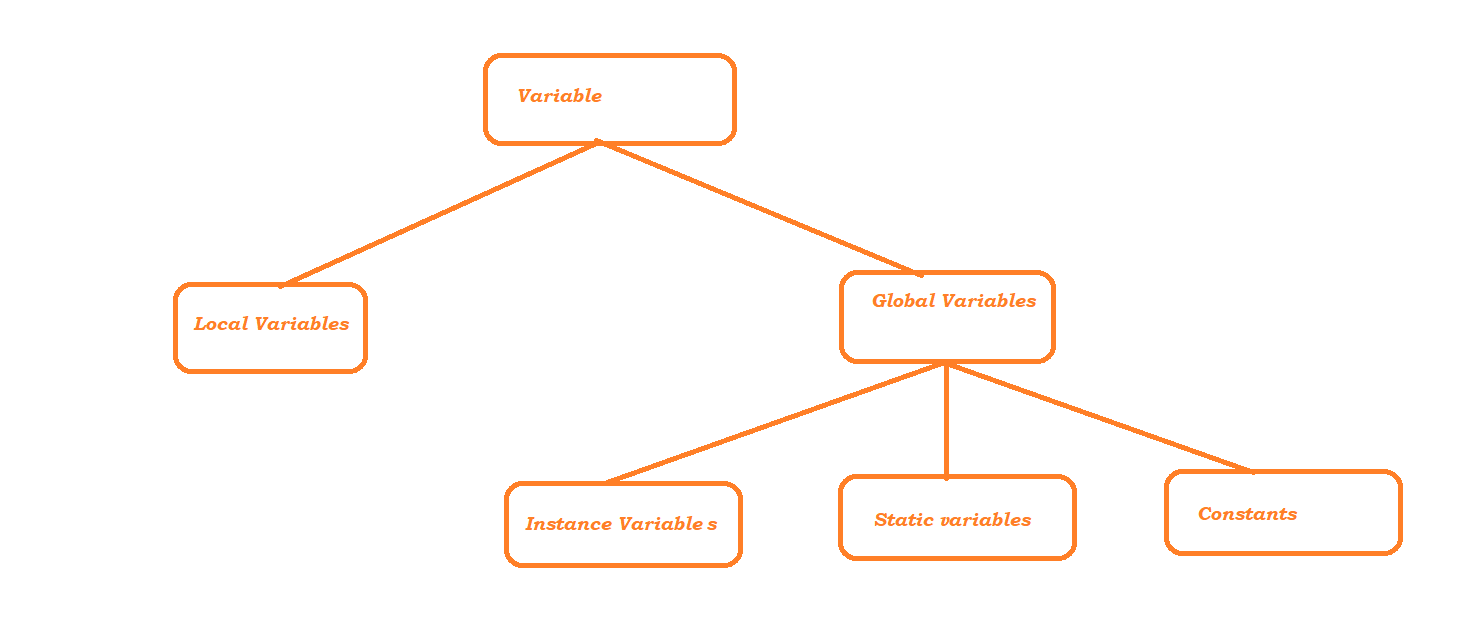
---------



Data Type :



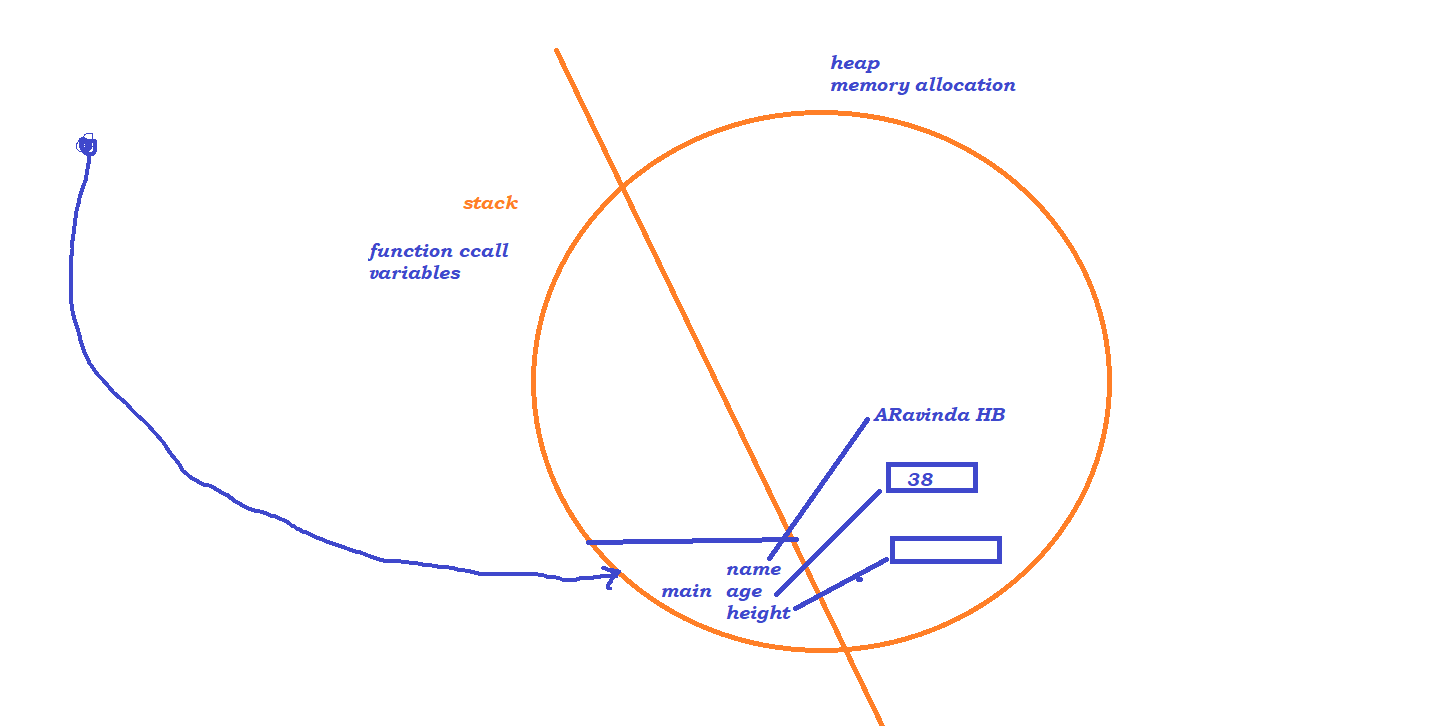
Variables :

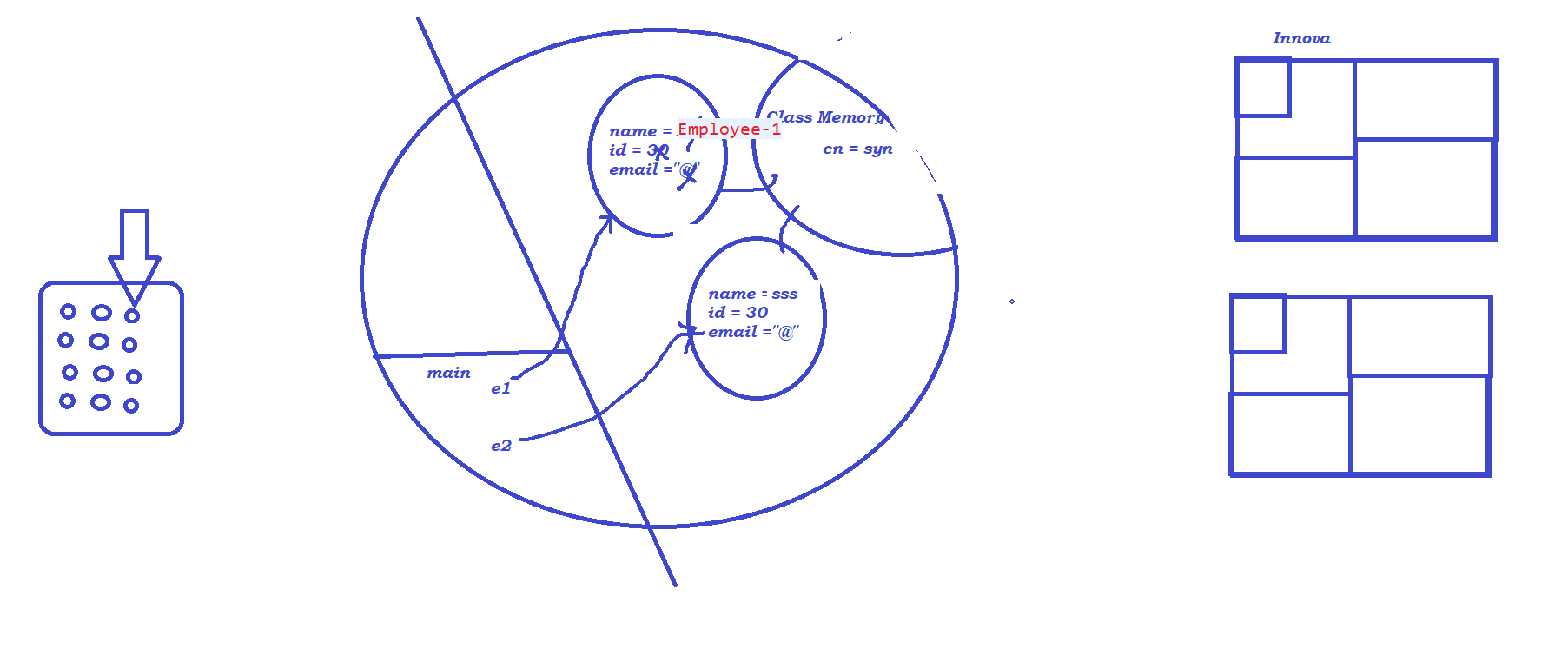


String name = “Aravinda HB”

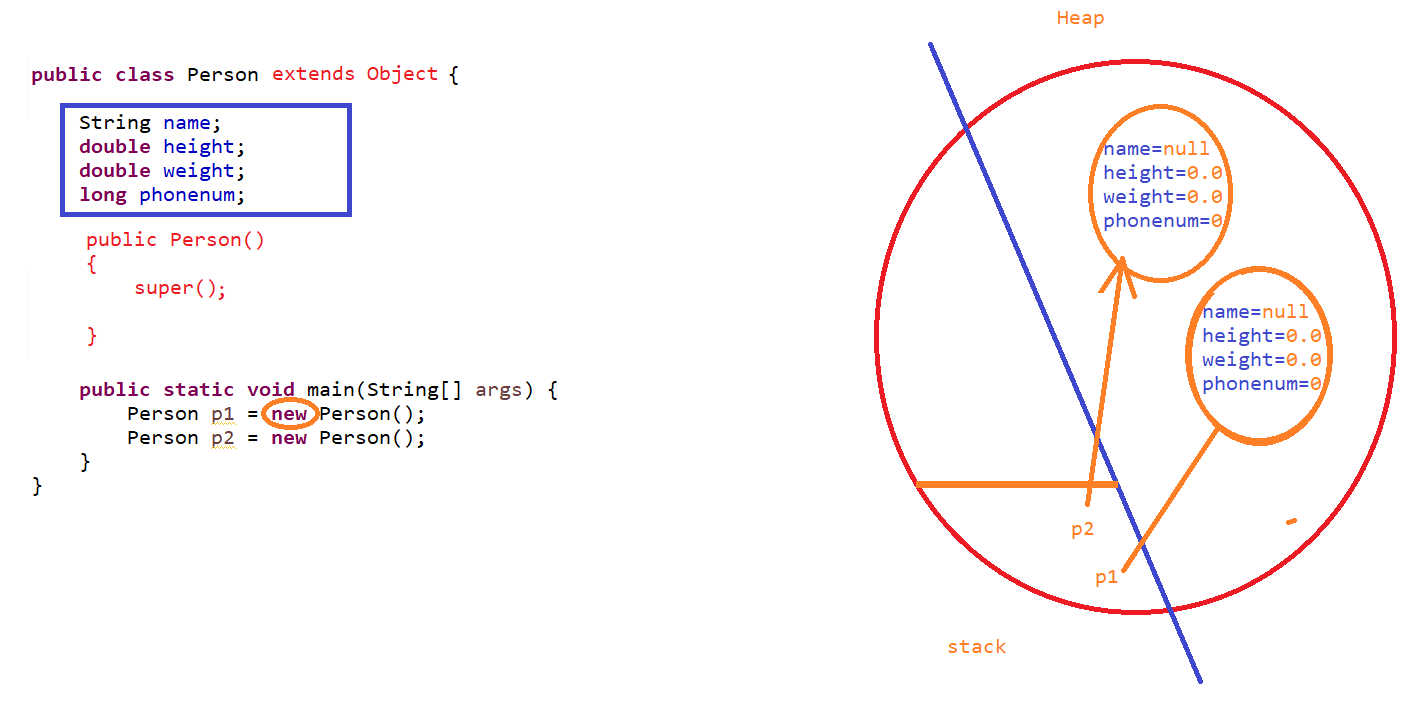
int age = 38

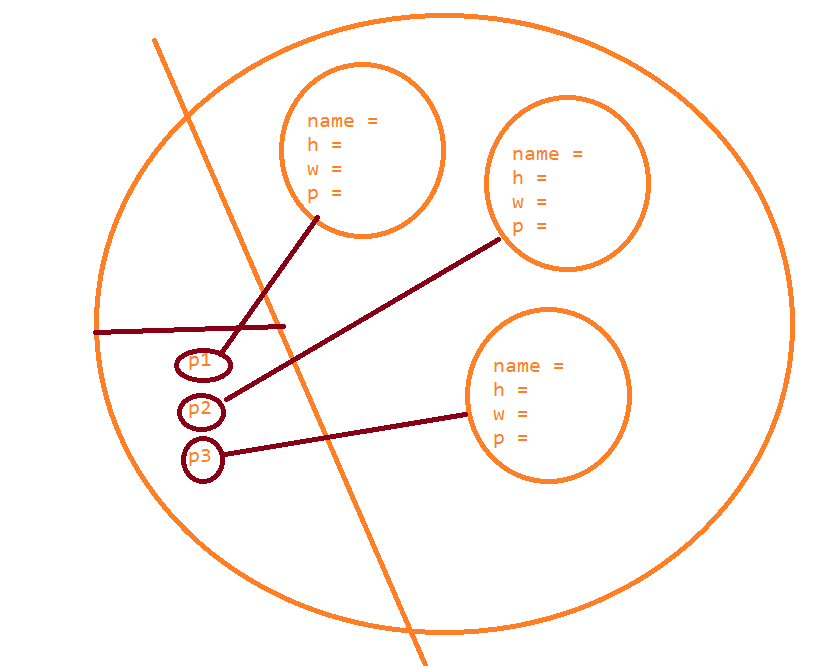
double height = 5.8



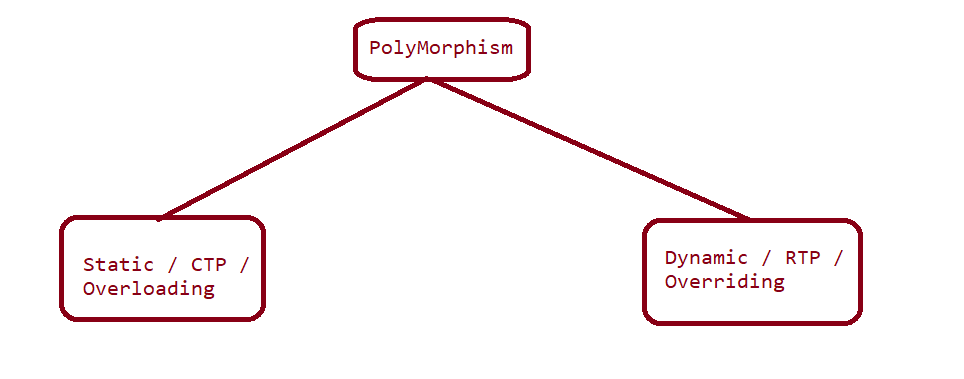


Constructors:





Polymorphism :



Initialization Blocks :

1. Static Initialization Block

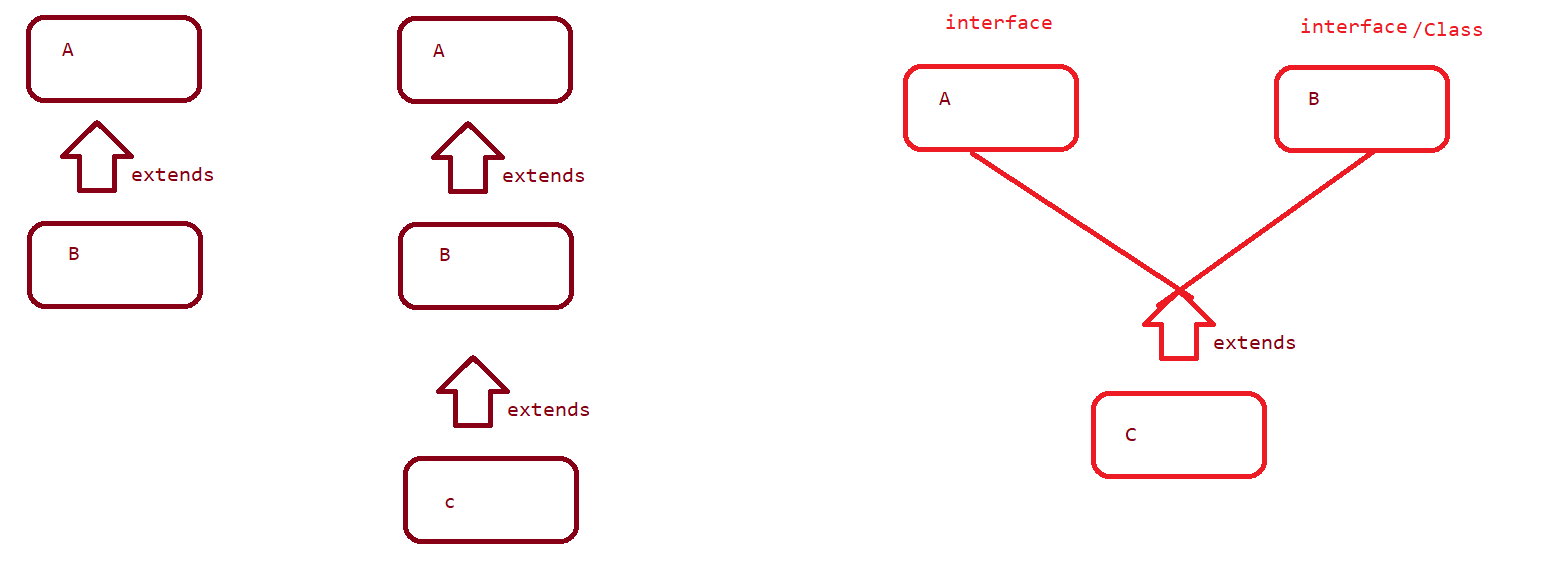
2. Instance Initialization Blocks

Abstract Classes:

Class with implementation gap

Interfaces

Inheritence

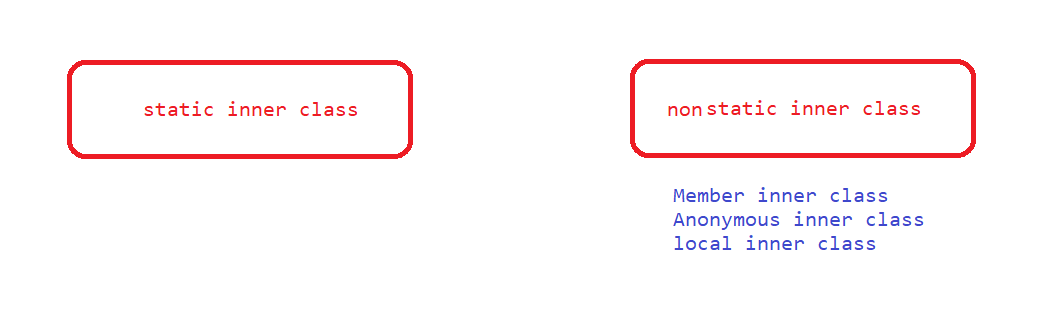


looping and conditional Statements

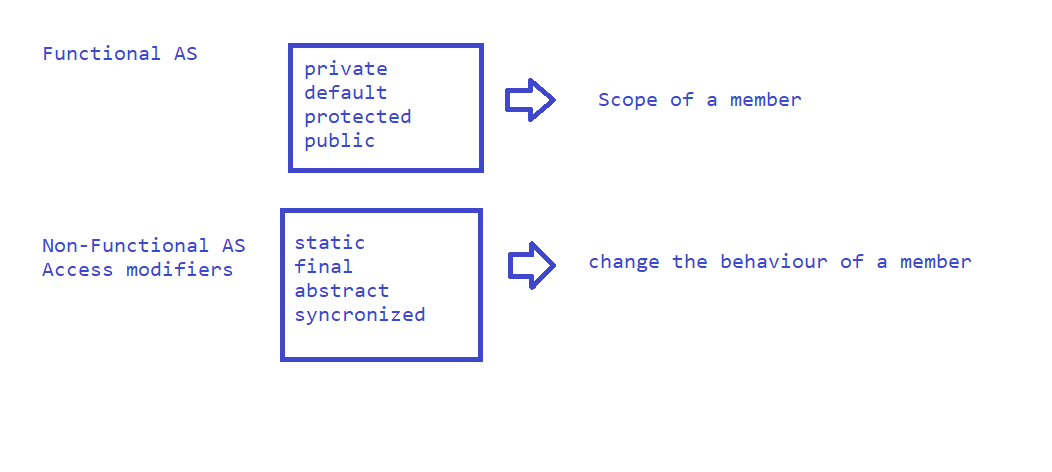
* for
* while
* do while
* foreach
* if
* if,else
* if elseif else
* switch

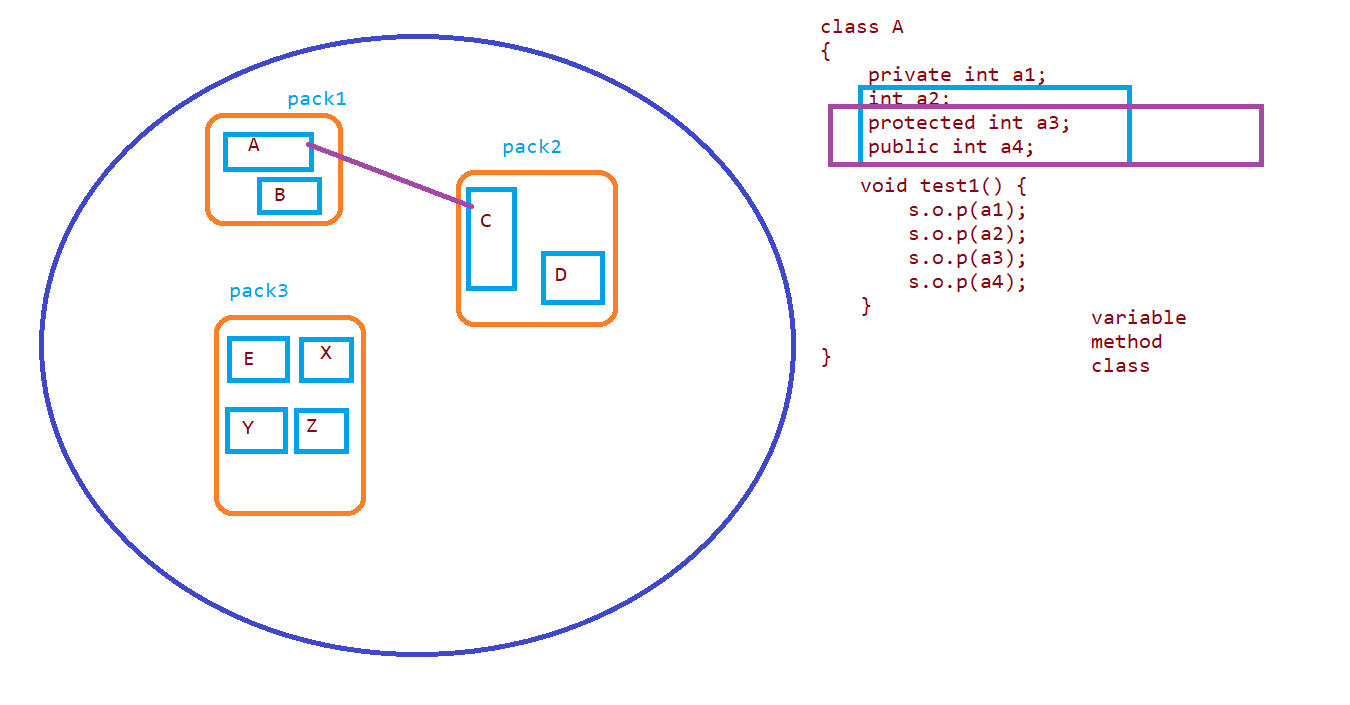
Inner Classes

Anonymous inner class

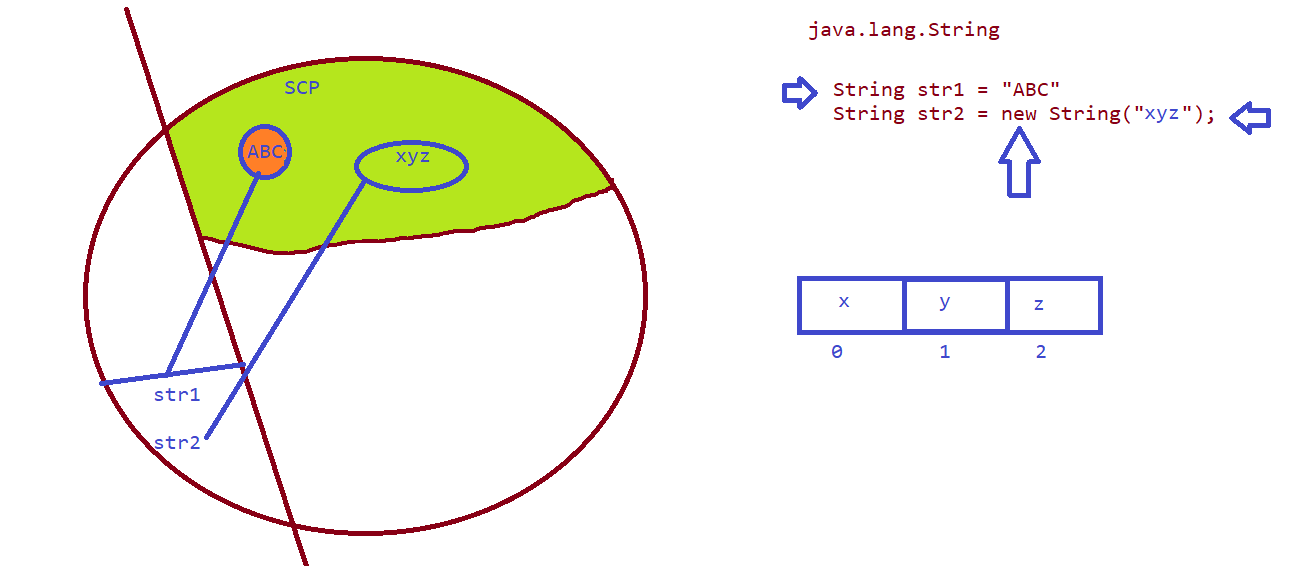


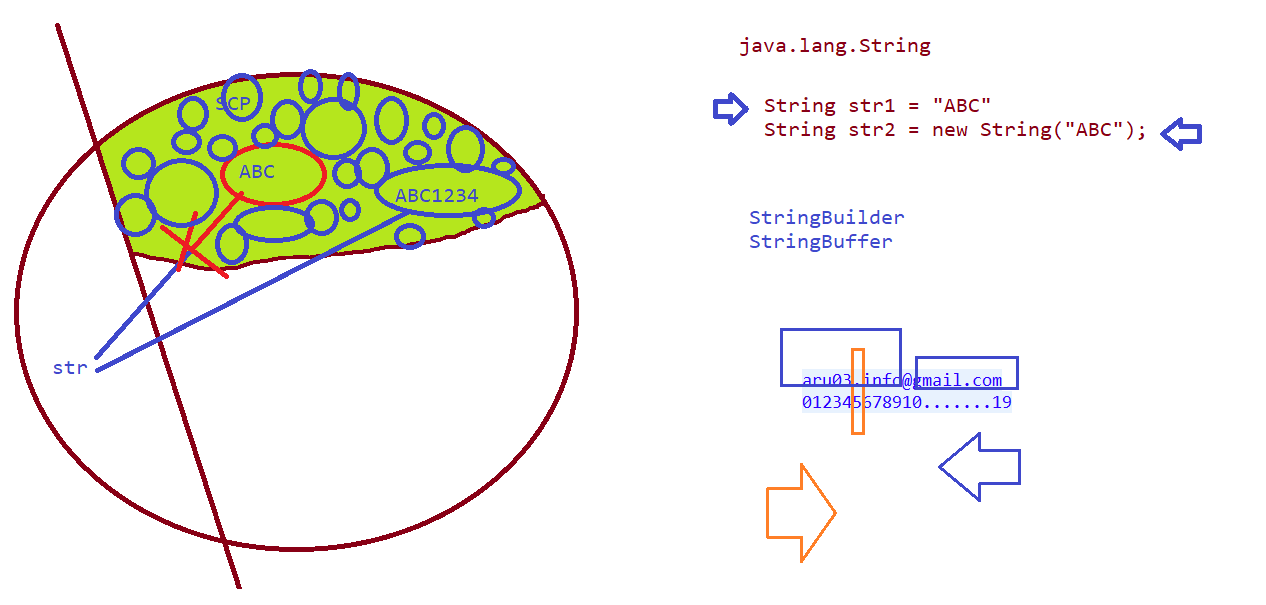
Access specifiers and modifiers :



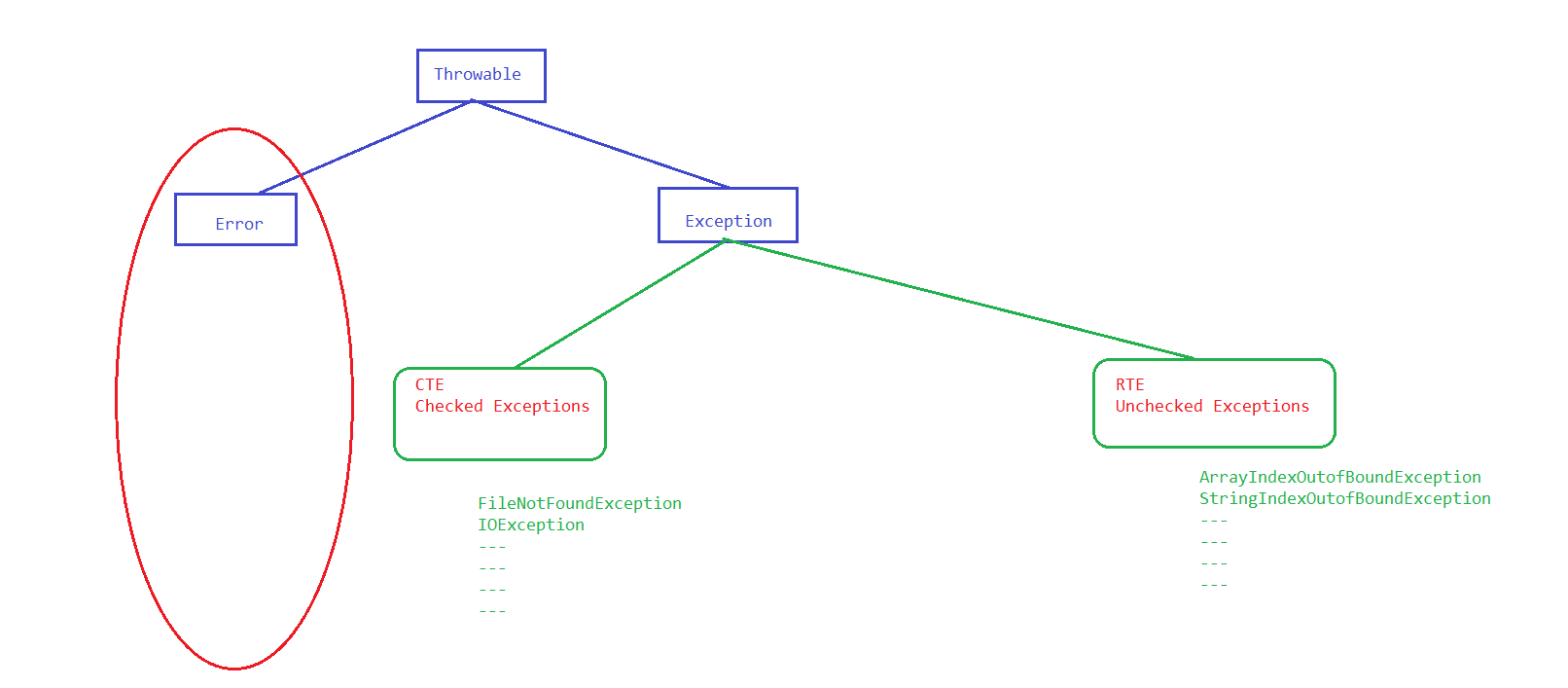


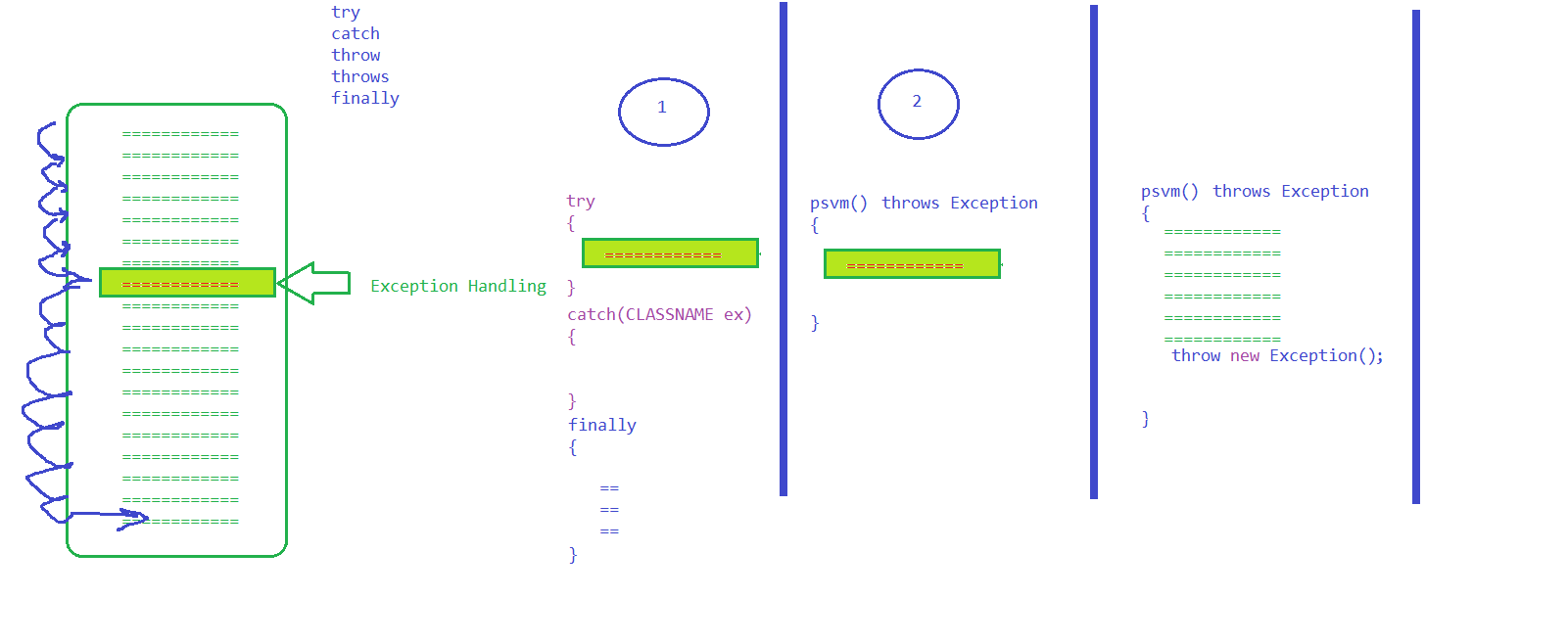
Strings :



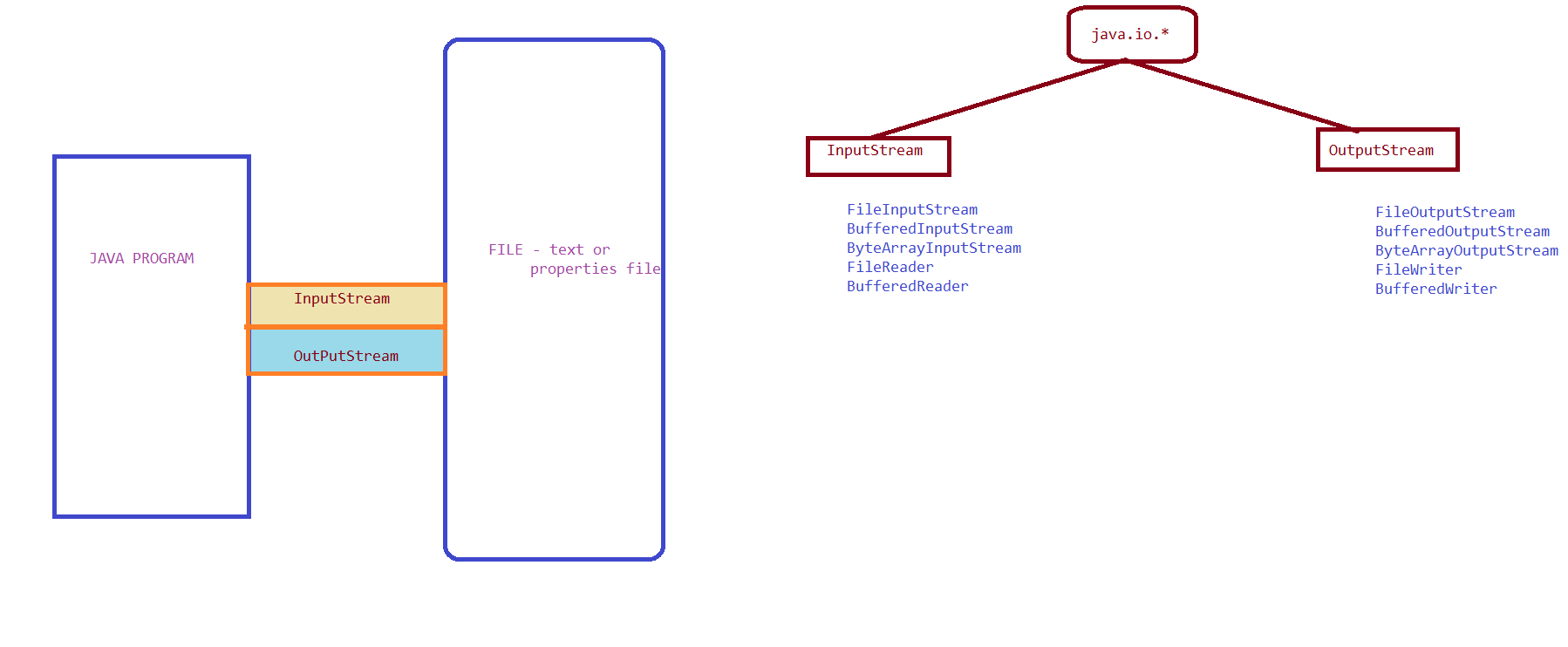


Exceptions:

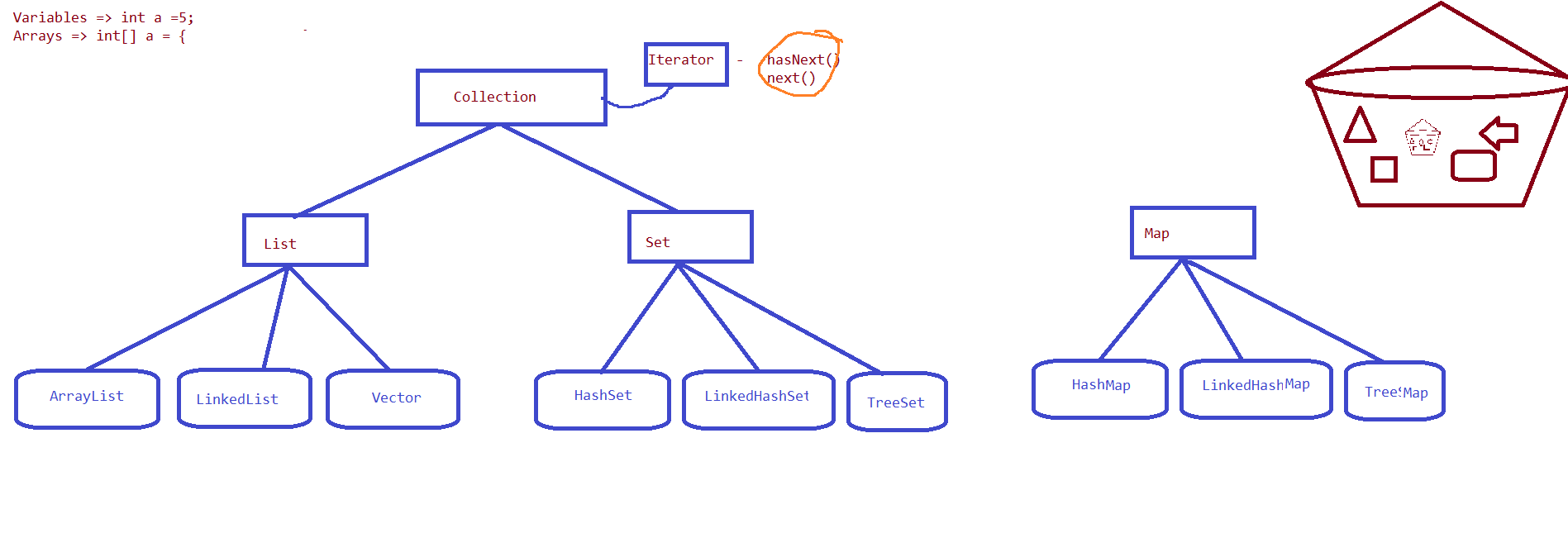




File Handling



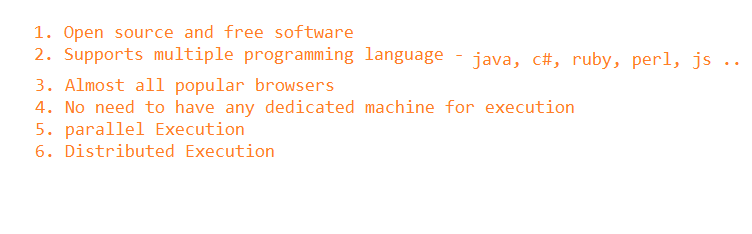
Collections:



Generics :

Parameterized Collections

Selenium:



* Identification
  1. id
  2. name
  3. classname
  4. tagname
  5. linktext
  6. partiallinktext
  7. css
  8. xpath
* CSS Selector :

1. htmltag[attribute = ‘value’] - input[placeholder='Enter address']
2. htmltag[id = ‘value’] OR htmltag#idvalue OR #idvalue
   1. input#autocomplete OR #autocomplete
3. htmltag[class = ‘value’] OR htmltag.classValue OR .classvalue
   1. input.form-control OR .form-control
4. Parent to child
   1. expression of parent > child expression OR TAG
   2. a#loginButton > div

* XPATH
  1. **Basic xpath**
     + //htmltag[@attribure=’value’]
  2. **Xpath using functions**
     + text() - > Exact text of an element
     + EX: //label[text()='Street address']
     + contains(arg1, arg2)
       - arg1 – attribute or text function
       - arg2 – partial value of attribute or text function
       - Attribute : //htmltag[contains(@attribute,’partialvalue’)]
         * EX: //img[contains(@src,'timer')]
       - Function: //htmltag[contains(functionname,’partialvalue’)]
         * EX: //label[contains(text(),'Keep')]
     + starts-with(arg1, arg2)
       - arg1 – attribute or text function
       - arg2 – starting value of attribute or text function
       - Attribute :
         * //\*[starts-with(@id,'ext-gen')]
       - Function:
         * //h3[starts-with(text(),'About')]
  3. **Logical operators**
     + AND -> //htmltag[@att1=’val1’ and @att2=’val2’]
       - //td[(@class='current day' or @class='wd day' or @class='we day') and text()='3']
     + OR ->//htmltag[@att1=’val1’ or @att2=’val2’]
     + NOT -> htmltag[@att1=’val1’ and not @att2=’val2’]
       - //td[text()='6' and not(@class='past day')]
  4. **Traverse from parent to child**
     + **Immediate Child:** //a[@id='loginButton']/div
     + **Somewhere :** //td[@id='loginButtonContainer']//div[text()='Login ']
  5. **Traverse from child to parent**
     + Whenever we see dependent and independent elements in the application then we go with traversing from child to parent
     + STEP-1 : Write an xpath for independent element
       - **//th[text()='Directed by']**
     + STEP-2 : Put the complete expression inside a square bracket(which is written in STEP-1 except // )
       - **//[th[text()='Directed by']]**
     + STEP-3 : Specify the parent HTML TAG
       - **//tr[th[ [text()='Directed by']]**
     + STEP-4 : Traverse to the dependent Element
       - **EX : //tr[th[text()='Directed by']]//a**
  6. **Axes Functions**
     + **Traversing between following-sibling**
     + **Traversing between preceding-sibling**
     + **Traversing till the beginning of the Page**
     + **Traversing till the end of the Page**
     + **Traversing to parent**
     + **Traversing to child**
     + **Traversing to ancestor**