***Dt : 3/11/2022***

***faq:***

***wt is the diff b/w***

***(i)getMessage()***

***(ii)printStackTrace()***

***(iii)toString()***

***(i)getMessage():***

***=>getMessage() method is used to display only msg from exception-object.***

***syntax:***

***String msg = obj.getMessage();***

***(ii)printStackTrace():***

***=>printStackTrace() method is used to display exception details like***

***msg,class\_name,method\_name and line\_no,from where the exception is raised.***

***syntax:***

***obj.printStackTrace();***

***(iii)toString():***

***=>toString() method is used to display msg and class\_name in exception***

***handling process.***

***syntax:***

***String d = obj.toString();***

***==================================================================***

***faq:***

***define "java.lang.NullPointerException"?***

***=>"java.lang.NullPointerException" is raised when we use NonPrimitive***

***datatype variables which are assigned with "null" values.***

***Ex-program:***

***package maccess;***

***public class DemoException7 {***

***public static String str;***

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

***int len = str.length();//raises NullPointerException***

***System.out.println("str : "+str.toString());***

***System.out.println("length of str : "+len);***

***}***

***}***

***o/p:***

***Exception in thread "main" java.lang.NullPointerException:***

***Cannot invoke "String.length()" because "maccess.DemoException7.str"***

***is null at maccess.DemoException7.main(DemoException7.java:5)***

***===================================================================***

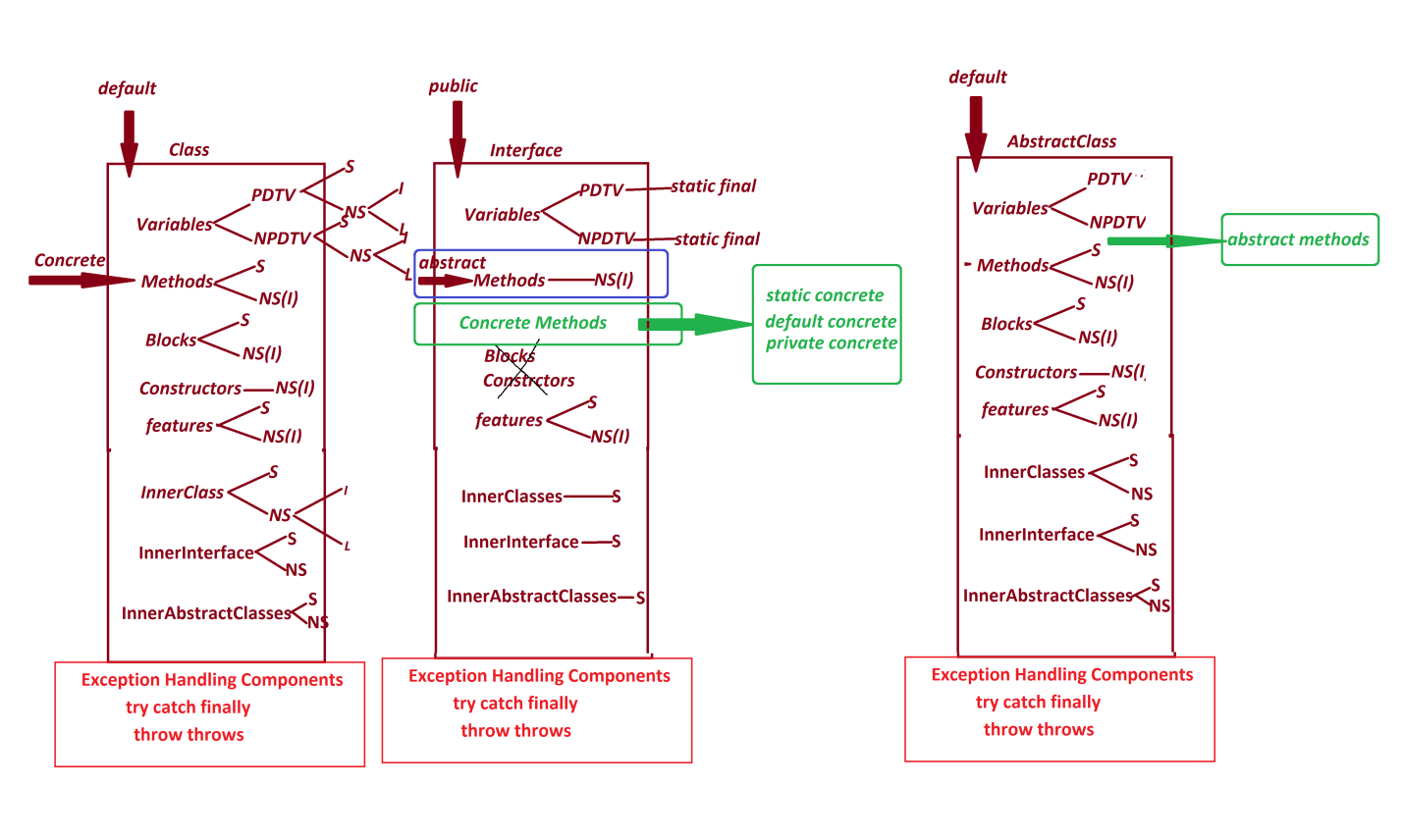
***faq:***

***define Encapsulation process?***

***=>The process of binding all the programming components into a Single***

***unit class is known as Encapsulation process.***

***Comparision Diagram:***

******

***====================================================================***

***faq:***

***define Abstraction process?***

***=>The process of hiding the background implementations which are not***

***needed by the end-user is known as Abstraction process.***

***=>we use Interfaces and Abstract classes to construct Abstraction process.***

***======================================================================***

***\*imp***

***PolyMorphism in Java:***

***=>The process in which the programming component having more than one***

***form is known as PolyMorphism.***

***Poly - Many***

***Morphism - Forms***

***=>PolyMorphism is categorized into two types:***

***1.Dynamic PolyMorphism***

***2.Static PolyMorphism***

***1.Dynamic PolyMorphism:***

***=>The PolyMorphism(Many-Forms) at execution stage is known as Dynamic***

***PolyMorphism or Runtime PolyMorphism.***

***Ex:***

***Method Overriding process***

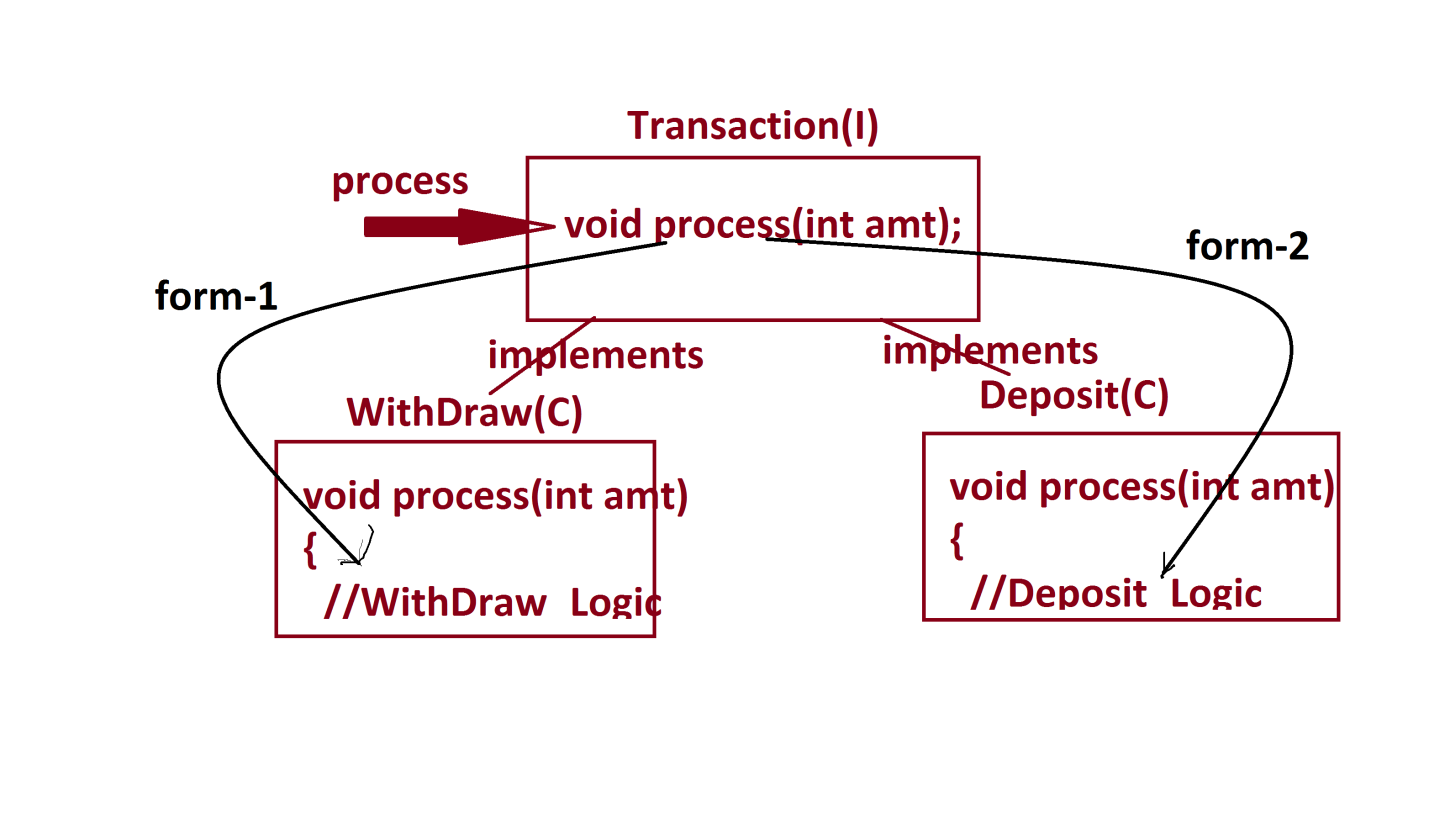
***Note:***

***=>Through Method Overriding process we can have more than one form at***

***execution stage,because of this reason Method Overriding process comes***

***under Dynamic PolyMorphism or Runtime PolyMorphism.***

***Diagram:***

******

***------------------------------------------------------------***

***2.Static PolyMorphism:***

***=>The PolyMorphism(Many-Forms) at compilation stage is known as static***

***PolyMorphism or Compiletime PolyMorphism.***

***Ex:***

***Method Overloading process***

***Note:***

***=>Through Method Overloading process we can have many forms for a same***

***method by differentiating Para\_list or Para\_type at compiltaion stage,***

***because of this reason Method Overloading process comes under Static***

***PolyMorphism or Compiletime Polymorphism.***

***dis(int,int)***

***dis(int,int,int)***

***dis(int,float)***

***=========================================================***

***\*imp***

***=>The compiler at compilation stage will control the following keywords:***

***1.static***

***2.private***

***3.final***

***1.static:***

***=>The following are the static programming components:***

***(a)static variables***

***(b)static methods***

***(c)static blocks***

***(d)static classes***

***(e)static interfaces***

***(f)static abstract classes***

***=>There is no concept of static constructors in Java.***