***Dt : 23/9/2022***

***faq:***

***can we perform Overriding process for standard main() method?***

***=>we cannot perform Overriding process for standrad main()***

***method because the main() method comes under static method***

***category.***

***faq:***

***Can we perform Overloading process for standard main() method?***

***=>Yes,we can perform Overloading process for standard main()***

***method.***

***faq:***

***can we pass parameters to standard main() method?***

***=>Yes,we can pass parameters to standard main() method while***

***execution command,because standard main() method call is in***

***execution command.***

***syntax:***

***java Class\_name arg1 arg2 arg3 ...***

***Ex : DemoMain.java***

***package maccess;***

***import java.util.Scanner;***

***public class DemoMain***

***{***

***public static void main(String[] args) //Standard main***

***{***

***Scanner s = new Scanner(System.in);***

***System.out.println("====main(String[])====");***

***for(String str : args)***

***{***

***System.out.println(str);***

***}//end of loop***

***System.out.println("------------------------");***

***System.out.println("Enter the value of k:");***

***DemoMain.main(s.nextFloat());//method call***

***System.out.println("------------------------");***

***System.out.println("Enter the value of z:");***

***DemoMain.main(s.nextInt());//method Call***

***s.close();***

***}***

***public static void main(float k)***

***{***

***System.out.println("====main(float)====");***

***System.out.println("The value k:"+k);***

***}***

***public static void main(int z)***

***{***

***System.out.println("====main(int)====");***

***System.out.println("The value z:"+z);***

***}***

***}***

***o/p:***

***E:\Demo137>javac DemoMain.java***

***E:\Demo137>java DemoMain nit hyd java 2022***

***====main(String[])====***

***nit***

***hyd***

***java***

***2022***

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

***Enter the value of k:***

***12.34***

***====main(float)====***

***The value k:12.34***

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

***Enter the value of z:***

***234***

***====main(int)====***

***The value z:234***

***E:\Demo137>***

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

***faq:***

***define Command line arguments program?***

***=>The process of passing arguments to standard main() method***

***while execution command is known as Command line arguments program.***

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

***\*imp***

***Types of Inheritances:***

***=>Inheritances are categorized into the following:***

***1.Single Inheritance***

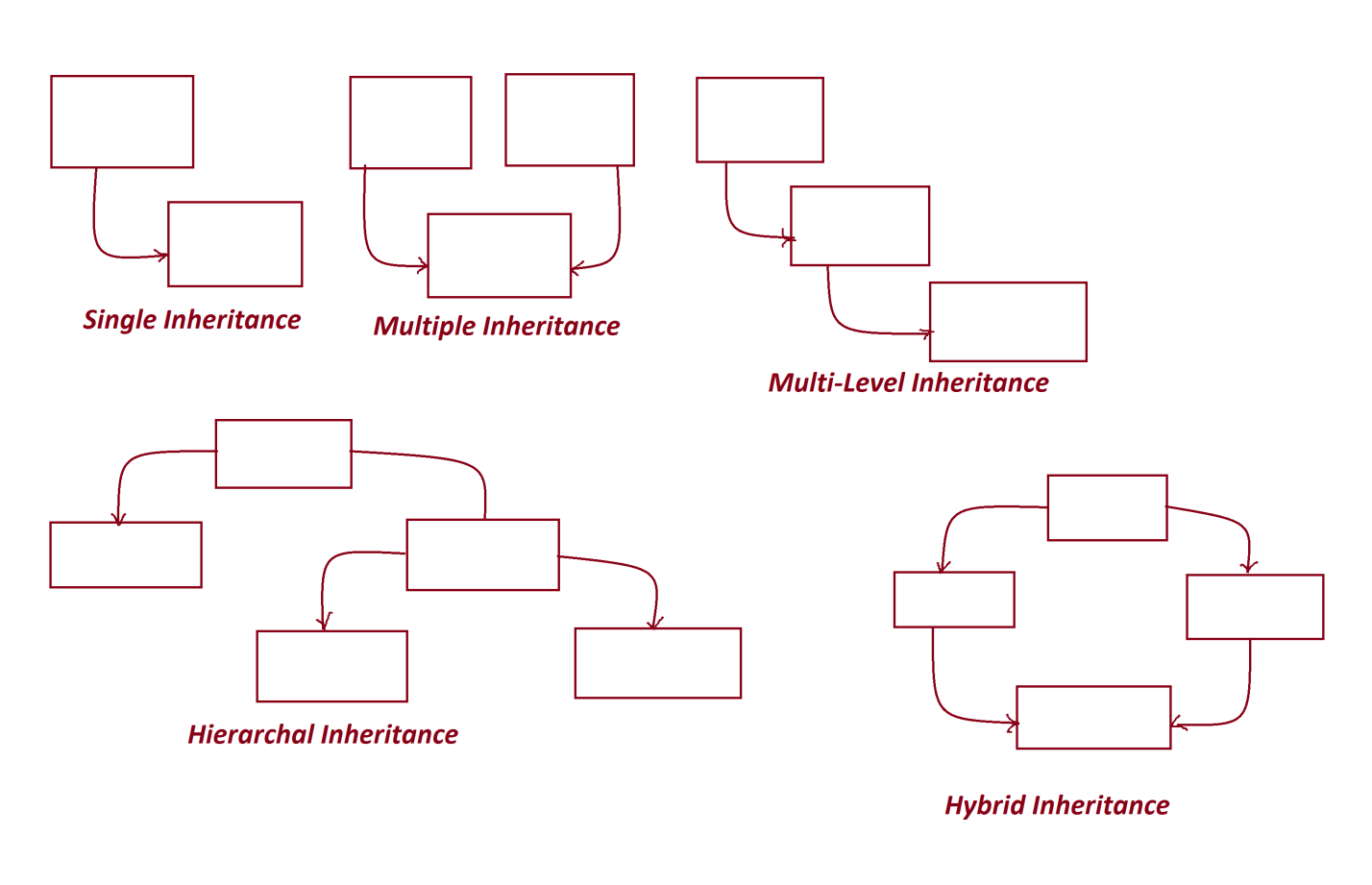
***2.Multiple Inheritance***

***3.Multi-level Inheritance***

***4.Hierarchal Inheritance***

***5.Hybrid Inheritance***

***Diagrams:***

******

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

***=>In realtime the Inheritance process is categorized into two***

***types:***

***(a)Single Inheritance***

***(b)Multiple Inheritance***

***(a)Single Inheritance:***

***=>The process of extracting the features from one class at-a-time***

***is known as Single Inheritance process.***

***Ex:***

***above programs***

***(b)Multiple Inheritance:***

***=>The process of extracting the features from more than one***

***Class at-a-time is known as Multiple Inheritance process.***

***Note:***

***=>Multiple Inheritance process using classes not available in Java,***

***because it leads to replication of programming components and raises***

***ambiguity,the ambiguity state application will generate Wrong***

***results.***

***=>In Java,Multiple Inheritance process can be performed using***

***Interfaces.***

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

***\*imp***

***Interfaces in Java:***

***=>From Java8 version onwards the interface is a collection of***

***Variables,abstract methods and Concrete methods.***

***faq:***

***define abstract methods?***

***=>The methods which are declared without method body are known as***

***abstract methods.***

***structure of abstract methods:***

***return\_type method\_name(para\_list);***

***faq:***

***define concrete methods?***

***=>The methods which are declared with method body are known as***

***Concrete methods.***

***Structure of Concrete methods:***

***return\_type method\_name(para\_list)***

***{***

***//body***

***}***

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

***Coding rules of Interfaces:***

***Rule-1 : we use 'interface' keyword to declare interfaces.***

***syntax:***

***interface Interface\_name***

***{***

***//Interface\_body***

***}***

***Rule-2 : The programming components which are declared within the***

***interface are automatically "public".***

***Rule-3 : Interfaces can be declared with both Primitive DataType***

***Variables and Non-Primitive DataType variables.***

***Rule-4 : The variables which are declared in interfaces are***

***automatically static variables and final variables.***

***Note:***

***(i)Static variables in interfaces will get the memory within the***

***interface and canbe accessed with Interface\_name.***

***(ii)final variables must be initialized with values and once***

***initialized cannot be modified.***

***(final variables are also known as constant variables and Secured***

***Variables)***

***Dt : 24/9/2022***

***Rule-5 : The methods which are declared within the interface are***

***automatically abstract methods.***

***Note:***

***=>'abstract' keyword is not manditory to declare abstarct methods.***

***=>There is no concept of static abstract methods.***

***Rule-6 : Interfaces cannot be instantiated,which means we cannot***

***create object for Interfaces.***

***Note:***

***=>In Java,Interfaces are known as abstract components.***

***Rule-7 : Interfaces are linked to classes through 'implements'***

***keyword and the classes are known as implementation classes.***

***Rule-8 : Implementation classes must construct body for abstract***

***methods of interfaces***

***Rule-9 : The interfaces canbe declared with any number of abstract***

***methods without retriction.***

***Rule-10 : The implementation classes must construct body for all***

***abstract methods of Interface.***

***Rule-11 : The implementation classes can also be declared with***

***Non-implemented methods.***

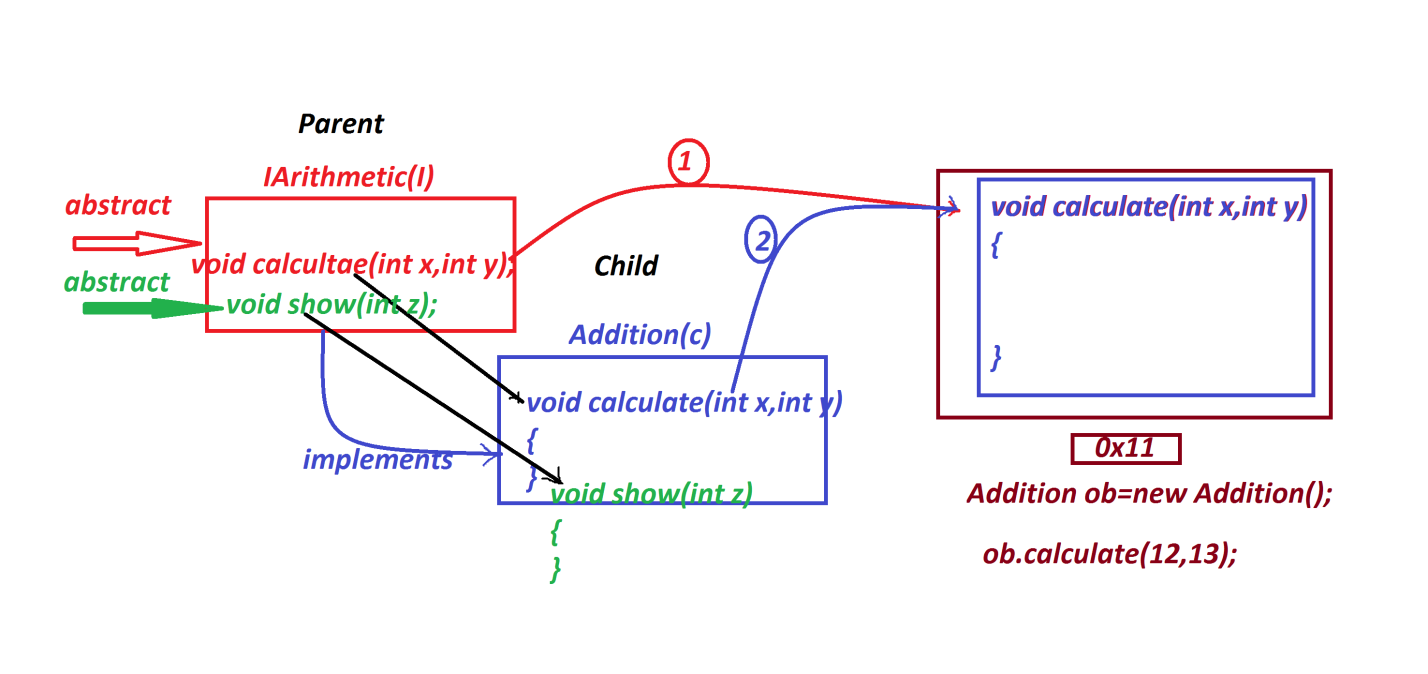
***Rule-12 : There is no concept of declaring Blocks and Constructors***

***in Interfaces.***

***Rule-13 : One Interface can have the features of another Interface***

***using 'extends' keyword***

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

******

***Ex-1 :***

***IArithmetic.java***

***package test;***

***public interface IArithmetic {***

***public abstract void calculate(int x,int y);***

***public abstract void show(int z);***

***}***

***Addition.java***

***package test;***

***public class Addition implements IArithmetic{***

***public void calculate(int x,int y)//Implemented method***

***{***

***System.out.println("====calculate(x,y)====");***

***System.out.println("Sum:"+(x+y));***

***}***

***public void show(int z)//Implemented method***

***{***

***System.out.println("====show(z)====");***

***System.out.println("The value z:"+z);***

***}***

***public void dis(int p)//Non-Implemented method***

***{***

***System.out.println("====Non-implemented dis(p)====");***

***System.out.println("The value p:"+p);***

***}***

***}***

***DemoInterface1.java(MainClass)***

***package maccess;***

***import test.\*;***

***public class DemoInterface1 {***

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

***//IArithmetic ob = new IArithmetic();//CompilationError***

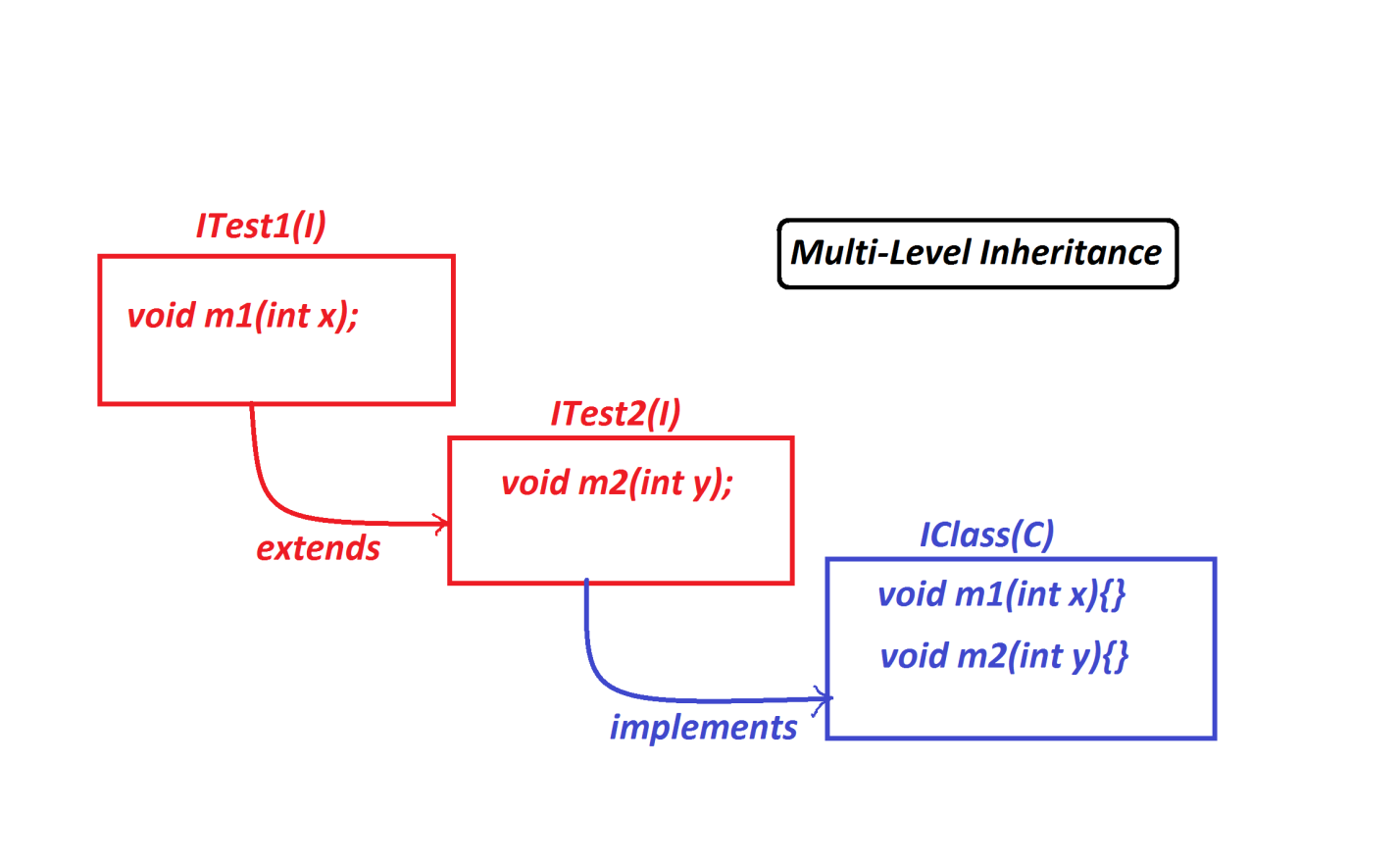
***Addition ob = new Addition();//implementation object***

***ob.calculate(12, 13);***

***ob.show(123);***

***}***

***}***

******

***Ex-2:***

***ITest1.java***

***package test;***

***public interface ITest1 {***

***public abstract void m1(int x);***

***}***

***ITest2.java***

***package test;***

***public interface ITest2 extends ITest1{***

***public abstract void m2(int y);***

***}***

***IClass.java***

***package test;***

***public class IClass implements ITest2{***

***public void m1(int x) {***

***System.out.println("====m1(x)=====");***

***System.out.println("x:"+x);***

***}***

***public void m2(int y) {***

***System.out.println("====m2(y)=====");***

***System.out.println("y:"+y);***

***}***

***}***

***DemoInterface2.java(MainClass)***

***package maccess;***

***import test.\*;***

***public class DemoInterface2 {***

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

***IClass ob = new IClass();***

***ob.m1(11);***

***ob.m2(12);***

***}***

***}***

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