***Dt : 14/9/2022***

***faq:***

***define Boxing process?***

***=>The process of binding primitive datatypes into WrapperClass***

***objects is known as Boxing Process.***

***Note:***

***=>we use Constructors to perform boxing process.***

***=>The following are the list of Constructors from WrapperClasses:***

***WrapperClass Constructors***

***Byte Byte(byte),Byte(String)***

***Short Short(short),Short(String)***

***Integer Integer(int),Integer(String)***

***Long Long(long),Long(String)***

***Float Float(float),Float(double),Float(String)***

***Double Double(double),Double(String)***

***Character Character(char)***

***Boolean Boolean(boolean)***

***Ex : DemoWrapperClass1.java***

***package maccess;***

***public class DemoWrapperClass1 {***

***@SuppressWarnings("removal")***

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

***//Boxing process***

***Integer ob1 = new ~~Integer~~(12);***

***Integer ob2 = new ~~Integer~~("14");***

***Float ob3 = new ~~Float~~(12.34F);***

***Float ob4 = new ~~Float~~(234.56);***

***Float ob5 = new ~~Float~~("23.34");***

***Character ob6 = new ~~Character~~('A');***

***Boolean ob7 = new ~~Boolean~~(true);***

***Boolean ob8 = new ~~Boolean~~("false");***

***System.out.println("====Display data from objects====");***

***System.out.println("ob1:"+ob1.toString());***

***System.out.println("ob2:"+ob2.toString());***

***System.out.println("ob3:"+ob3.toString());***

***System.out.println("ob4:"+ob4.toString());***

***System.out.println("ob5:"+ob5.toString());***

***System.out.println("ob6:"+ob6.toString());***

***System.out.println("ob7:"+ob7.toString());***

***System.out.println("ob8:"+ob8.toString());***

***}***

***}***

***o/p:***

***====Display data from objects====***

***ob1:12***

***ob2:14***

***ob3:12.34***

***ob4:234.56***

***ob5:23.34***

***ob6:A***

***ob7:true***

***ob8:false***

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

***faq:***

***define AutoBoxing process?***

***=>The Boxing process which is performed automatically is known as***

***AutoBoxing process.***

***Note:***

***=>In AutoBoxing process the Primitive DataType values are assigned***

***to Non-Primitive datatype variables.***

***Diagram:***

***Ex : DemoWrapperClass2.java***

***package maccess;***

***public class DemoWrapperClass2 {***

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

***//AutoBoxing process***

***Integer ob1 = 12;***

***Float ob2 = 12.34F;***

***Character ob3 = 'A';***

***Boolean ob4 = true;***

***System.out.println("====Display data from objects====");***

***System.out.println("ob1:"+ob1.toString());***

***System.out.println("ob2:"+ob2.toString());***

***System.out.println("ob3:"+ob3.toString());***

***System.out.println("ob4:"+ob4.toString());***

***}***

***}***

***o/p:***

***====Display data from objects====***

***ob1:12***

***ob2:12.34***

***ob3:A***

***ob4:true***

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

***faq:***

***define UnBoxing process?***

***=>The process of taking primitive datatypes outof WrapperClass***

***objects is known as UnBoxing process.***

***Note:***

***=>we use the following methods to perform UnBoxing process:***

***public byte byteValue();***

***public short shortValue();***

***public int intValue();***

***public long longValue();***

***public float floatValue();***

***public double doubleValue();***

***public char charValue();***

***public boolean booleanValue();***

***Ex : DemoWrapperClass3.java***

***package maccess;***

***public class DemoWrapperClass3 {***

***@SuppressWarnings("removal")***

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

***//Boxing process***

***Integer ob1 = new ~~Integer~~(12);***

***Float ob2 = new ~~Float~~(12.34F);***

***Character ob3 = new ~~Character~~('A');***

***Boolean ob4 = new ~~Boolean~~(true);***

***//UnBoxing process***

***int i = ob1.intValue();***

***float f = ob2.floatValue();***

***char ch = ob3.charValue();***

***boolean bl = ob4.booleanValue();***

***System.out.println("====Display the data===");***

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

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

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

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

***}***

***}***

***o/p:***

***====Display the data===***

***i:12***

***f:12.34***

***ch:A***

***bl:true***

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

***faq:***

***define AutoUnBoxing process?***

***=>The UnBoxing process which is performed automatically is known***

***as AutoUnBoxing process.***

***Note:***

***=>In AutoUnBoxing process we assign Non-Primitive datatype***

***variables to Primitive datatype variables.***

***Ex : DemoWrapperClass4.java***

***package maccess;***

***public class DemoWrapperClass4 {***

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

***//AutoBoxing process***

***Integer ob1 = 12;***

***Float ob2 = 12.34F;***

***Character ob3 = 'A';***

***Boolean ob4 = true;***

***//AutoUnBoxing process***

***int i = ob1;***

***float f = ob2;***

***char ch = ob3;***

***boolean bl = ob4;***

***System.out.println("====Display the data===");***

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

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

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

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

***}***

***}***

***o/p:***

***====Display the data===***

***i:12***

***f:12.34***

***ch:A***

***bl:true***

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

***Note:***

***=>All WrapperClass objects are automatically 'Immutable Objects'***

***=>Frameworks(ORM tools) will accept the data only in the form of***

***objects,because of this reason we have to make primitive***

***datatypes available in the form of objects.***

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

***faq:***

***define Annotation?***

***=>The tag based information which is added to the programming***

***components(Interface,class,method,variable) is known as Annotation.***

***=>we use "@" symbol to represent annotation.***

***=>These annotations will provide information to compiler at***

***Compilation stage or information to execution control at execution***

***stage.***

***=>The following are some important annotations from CoreJava:***

***(i)@SuppressWarnings***

***(ii)@Override***

***(i)@SuppressWarnings:***

***=>@SuppressWarnings annotation will provide information to***

***compiler to close the raised Wranings.***

***(ii)@Override:***

***=>@Override annotation will also provide information to***

***compiler to check the method is Overriding method of not.***

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

***Dt : 15/9/2022***

***"Object" class:***

***=>"Object" class is from java.lang package and which is***

***ParentClass or SuperClass of all the classes declared inthe***

***application.***

***=>The following are some important methods of Object class:***

***1.hashCode()***

***2.equals()***

***3.clone()***

***4.wait()***

***5.notify()***

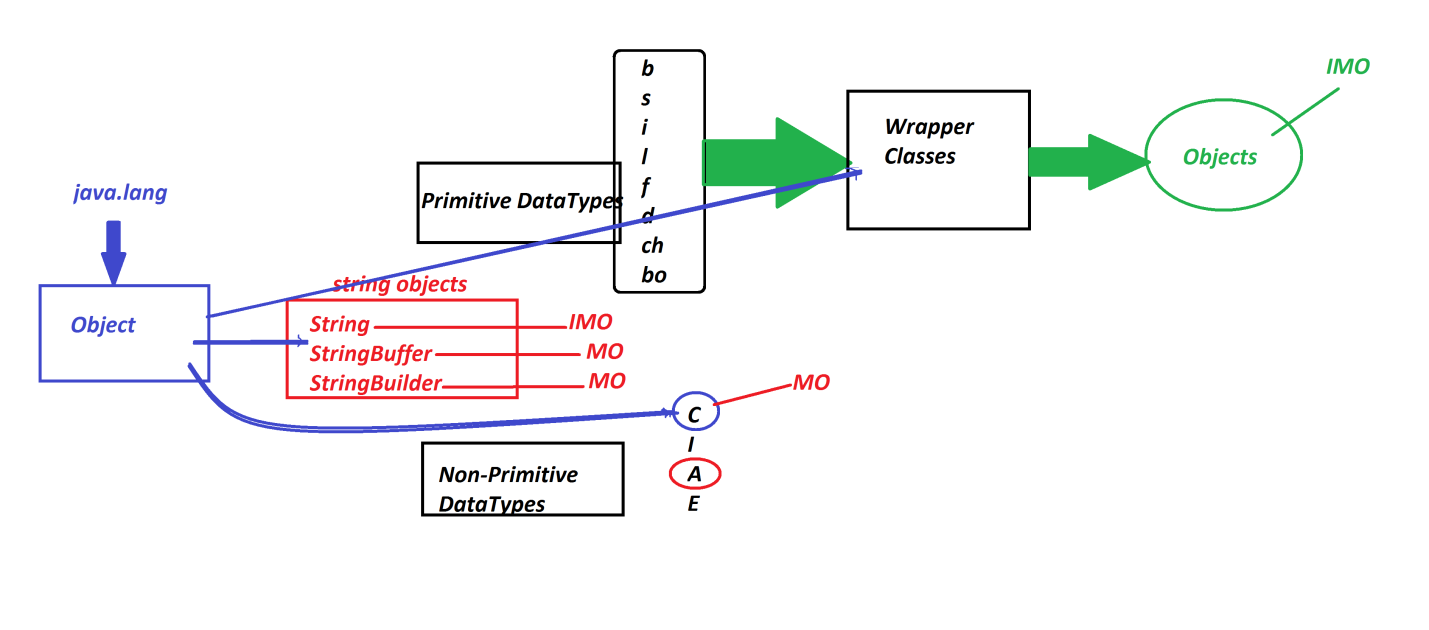
***6.notifyAll()***

***7.toString()***

***8.finalize()***

***9.getClass()***

***Diagram:***

******

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

***\*imp***

***Arrays in Java:***

***=>Array is a sequenced collection of elements of same data***

***type.***

***=>Array is a Sequences collection of Similer objects.***

***(Objects generated from same class are similer objects)***

***Types of Arrays:***

***=>Arrays are categorized into two types:***

***1.Single Dimensional Arrays***

***2.Multi-Dimensional Arrays***

***1.Single Dimensional Arrays:***

***=>The Arrays which are represented using single dimension are***

***known as Single-D Arrays or 1-D Arrays.***

***syntax:***

***Class\_name arr\_var[] = new Class\_name[size];***

***Ex-program-1:***

***wap to read and display Integer WrapperClass objects using Array?***

***Ex : DemoArray1.java***

***package maccess;***

***import java.util.\*;***

***public class DemoArray1 {***

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

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

***System.out.println***

***("Enter the size of Array to hold Integer Objects");***

***int size = s.nextInt();***

***Integer a[] = new Integer[size];***

***System.out.println("Enter "+size+" Integer objects:");***

***for(int i=0;i<a.length;i++)***

***{***

***a[i] = new ~~Integer~~(s.nextInt());***

***//Adding Integer object to Array Object***

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

***System.out.println("===Display Using Old for loop===");***

***for(int i=0;i<a.length;i++)***

***{***

***System.out.print(a[i].toString()+" ");***

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

***System.out.println("\n==Display Using Extended for(Java5)====");***

***for(Integer k : a)***

***{***

***System.out.print(k.toString()+" ");***

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

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

***}***

***}***

***o/p:***

***Enter the size of Array to hold Integer Objects***

***5***

***Enter 5 Integer objects:***

***11***

***10***

***23***

***22***

***21***

***===Display Using Old for loop===***

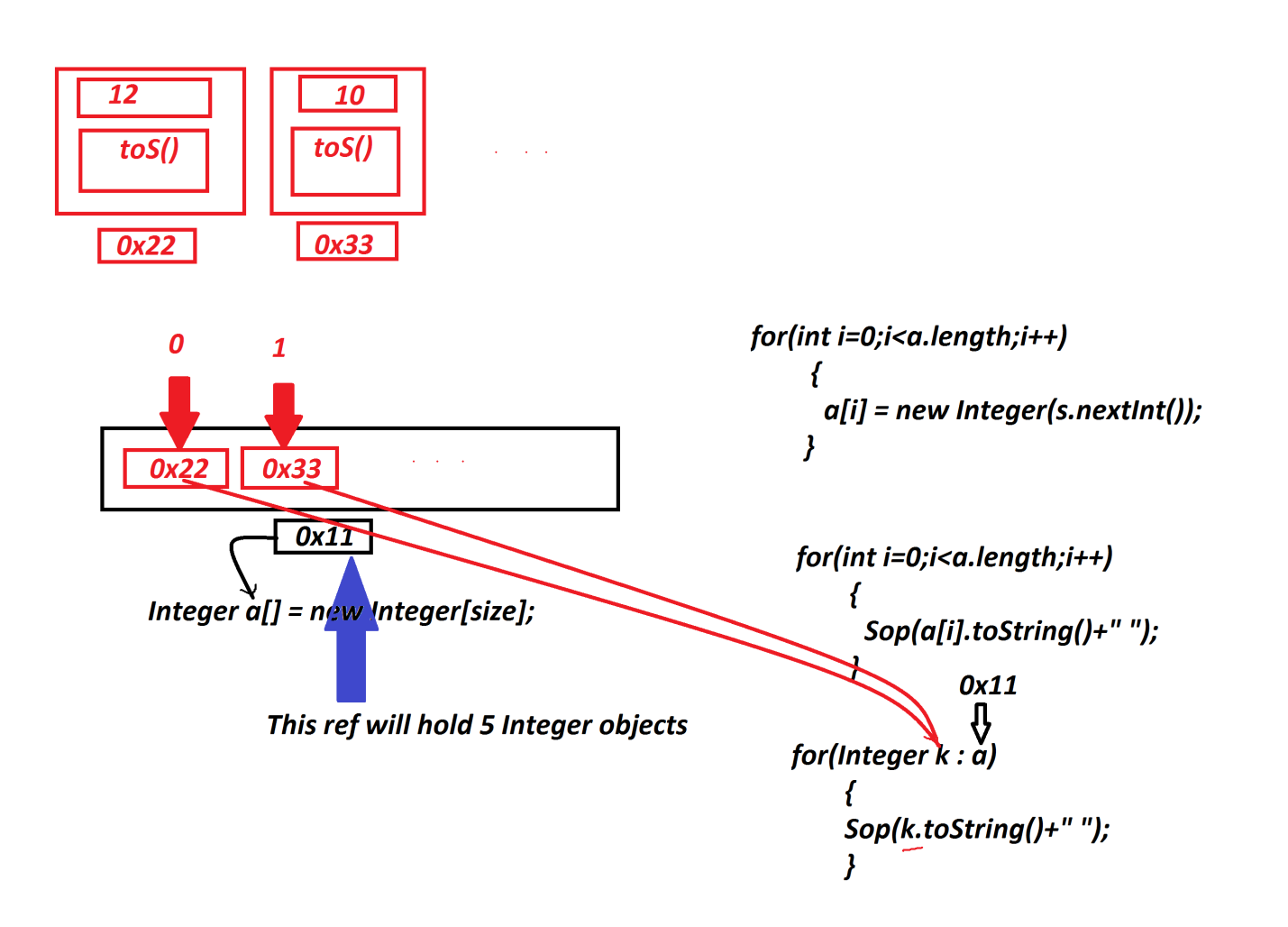
***11 10 23 22 21***

***==Display Using Extended for(Java5)====***

***11 10 23 22 21***

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

***Diagram:***

******

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

***define 'Extended for'?***

***=>'Extended for' introduced by Java5 version and which is***

***auto-retrieval loop used to retrieve elements from Arrays***

***based on Array\_name or Container\_name.***

***syntax:***

***for(Class\_name var\_name : Container\_name)***

***{***

***//Loop\_body***

***}***

***Note:***

***=>This 'Extended for' is also known as 'Enhanced for loop'***

***or for-each loop.***

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

***faq:***

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

***(i)length***

***(ii)length()***

***=>'length' keyword is used to find the length of Array and***

***'length()' method is used to find the length of String.***

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

***Ex-Program-2:***

***wap to read and display String objects using Array?***

***DemoArray2.java***

***package maccess;***

***import java.util.\*;***

***public class DemoArray2 {***

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

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

***System.out.println***

***("Enter the size of Array to hold String Objects");***

***int size = Integer.parseInt(s.nextLine());***

***String a[] = new String[size];***

***System.out.println("Enter "+size+" String objects:");***

***for(int i=0;i<a.length;i++)***

***{***

***a[i] = new String(s.nextLine());***

***//Adding Integer object to Array Object***

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

***System.out.println("===Display Using Old for loop===");***

***for(int i=0;i<a.length;i++)***

***{***

***System.out.print(a[i].toString()+" ");***

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

***System.out.println("\n==Display Using Extended for(Java5)====");***

***for(String k : a)***

***{***

***System.out.print(k.toString()+" ");***

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

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

***}***

***}***

***o/p:***

***Enter the size of Array to hold String Objects***

***5***

***Enter 5 String objects:***

***java***

***program***

***nit***

***hyd***

***task***

***===Display Using Old for loop===***

***java program nit hyd task***

***==Display Using Extended for(Java5)====***

***java program nit hyd task***

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

***Ex-program-3:***

***wap to read and display multiple Products?***

***Product.java***

***package test;***

***public class Product {***

***public String code,name;***

***public float price;***

***public int qty;***

***public Product(String code,String name,float price,int qty)***

***{***

***this.code=code;***

***this.name=name;***

***this.price=price;***

***this.qty=qty;***

***}***

***public String toString()***

***{***

***return code+"\t"+name+"\t"+price+"\t"+qty;***

***}***

***}***

***DemoArray3.java(MainClass)***

***package maccess;***

***import java.util.\*;***

***import test.Product;***

***public class DemoArray3 {***

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

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

***System.out.println***

***("Enter the size of Array to hold Product Objects");***

***int size = Integer.parseInt(s.nextLine());***

***Product a[] = new Product[size];***

***System.out.println("Enter "+size+" ProductDetails :");***

***for(int i=0;i<a.length;i++)***

***{***

***System.out.println("===Product-"+(i+1)+"====");***

***System.out.println("Enter the ProdCode:");***

***String code = s.nextLine();***

***System.out.println("Enter the ProdName:");***

***String name = s.nextLine();***

***System.out.println("Enter the ProdPrice:");***

***float price = Float.parseFloat(s.nextLine());***

***System.out.println("Enter the ProdQty:");***

***int qty = Integer.parseInt(s.nextLine());***

***a[i] = new Product(code,name,price,qty);***

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

***System.out.println("===Display Using Old for loop===");***

***for(int i=0;i<a.length;i++)***

***{***

***System.out.println(a[i].toString()+" ");***

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

***System.out.println("==Display Using Extended for(Java5)====");***

***for(Product k : a)***

***{***

***System.out.println(k.toString()+" ");***

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

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

***}***

***}***

***o/p:***

***Enter the size of Array to hold Product Objects***

***3***

***Enter 3 ProductDetails :***

***===Product-1====***

***Enter the ProdCode:***

***A111***

***Enter the ProdName:***

***Mouse***

***Enter the ProdPrice:***

***123.45***

***Enter the ProdQty:***

***12***

***===Product-2====***

***Enter the ProdCode:***

***A234***

***Enter the ProdName:***

***CDDDR***

***Enter the ProdPrice:***

***234.56***

***Enter the ProdQty:***

***12***

***===Product-3====***

***Enter the ProdCode:***

***A564***

***Enter the ProdName:***

***FDDDR***

***Enter the ProdPrice:***

***456.23***

***Enter the ProdQty:***

***23***

***===Display Using Old for loop===***

***A111 Mouse 123.45 12***

***A234 CDDDR 234.56 12***

***A564 FDDDR 456.23 23***

***==Display Using Extended for(Java5)====***

***A111 Mouse 123.45 12***

***A234 CDDDR 234.56 12***

***A564 FDDDR 456.23 23***

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