Tuesday, November 16, 2021 10:31 AM

Manual array comparison ----------

We first compare length of the two arrays …
Boolean arraydecision = true;
If the length is not same ---- decision made that arrays are not equal
If they are equal then we have to check each element
For(i=…arr.length)
If(! (arr[i].equals(otherarr[i]))
----decision is made that arrays are different
---no need to continue the loop break !!!

check the arraydecision

API array comparison
boolean result =Arrays.equals(arr1,arr2)

Polymorphism ---
Overloading , Overriding

NO dynamic polymorphism in static methods !!!
All non-static methods are resolved by RTT of the calling object .

Keywords = final , abstract

Final class cannot be subclassed !!!! No class can extend a final class.

final method cannot be overridden !!!

abstract class MUST be subclassed !! This class cannot be instantiated , We cannot create object of abstract class

Abstract method does not have a definition / does not have a method body !!! Abstract method must be overriden

If a method is abstract then the class in which the method is written must be abstract !!!!
If the subclass of the abstract class
a. implements the abstract method then the subclass is CONCRETE class, its object can be created
b. NOT implement the abstract method then the subclass has to become ABSTRACT , its object cannot created

---

WHY ABSTRACT ??? Abstract AAKHIR KYU !!!!

Abstract is USED for dynamic polymorphism !!!

HW ----

Class Shop ---- User of Product hierarchy

Main

---- create an array of Product

```
Product[] shopping = new Product[4];
    Shopping[0] = new Toy("doll", "non-battery", 300);
    Shopping[1] = new Toy("car" ,"battery" ,200);
    Shopping[2] = new Dress("sherwani","blue","L", 2000);
    ….
```

--- ask the user which products should be added in the array

--- call showBill pass the Product[]

```
public static void showBill ( Product[] arr)
{
    Call the getFinalCost method of each product
    Make the total and show the desc of each product and the individual cost
    And show total
}
```

Product

Write two abstract methods getDesc and getFinalCost

Initially write following subclasses

Toy

name, category-battery/nonbattery , cost

Dress

Name, material , size , price

INTERFACES ----------

Interfaces are similar to abstract classes

--------- We cannot create objects of interfaces (no instantiation)

-------- It is used for OCP -- in user of hierarchy

-------- It is used to force a subclass to OVERRIDE certain methods

differences

Interfaces have ONLY abstract methods + default methods

Interfaces can have only public static final properties !!!

TO become a subclass of abstract class we used "extends" keyword
TO become a s subclass of interface we use "implements" keyword

One class can extend from only one abstract class
One class can implement multiple interfaces

HW -------

Write an interface Sellable
getSellingPrice()
setDiscount(double)

Class CD implements Sellable
{
Name, music/video , cost , duration , artist (properties , 2 constructors, getters, setters, toString)
}

Class Laptop implements Sellable
{
Brand, RAM, HDD, CPU-type , Cost
}

Class User
Main
create 3/4Sellable objects show details (toString)
Show the sellingprice of each Sellable object
