Monday, November 15, 2021 10:37 AM

Object super class, all classes are subclasses of Object class

-Object class methods ---- clone, equals, tostring, hashcode, getClass, notify,notifyAll , wait

Superclass ref = new subClass() ; // Upcasting

((Subclass)superclassref).subclassMethod(); //downcasting

Polymorphism ----
Poly = many
Morph= forms
Many forms of a METHOD !!!!

When is the method RESOLVED !!!
It may be resolved at compile time = Static POLYMORPHSIM / Compile time Polymorphism
It may be resolved at run time = Dynamic POLYMORPHISM / Run time Polymorphism

Static Polymorphism = METHOD OVERLOADING !!! CONSTRUCTOR OVERLOADING

Keep the name of the method SAME , but change the PARAMETER LIST } ALL the methods can be in ONE CLASS
Example ----- we can show the example of Student constructor overloading
Substring method of String
println method in sysout

Dynamic Polymorphism = METHOD OVERRIDING , Always done within a hierarchy

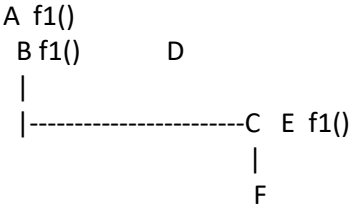
The signature /prototype of the method including Parameter list , name of method REMAINS SAME !!!
Class in which the TWO forms of the method are defined CHANGES

@Override = annotation
This annotation will tell compiler --- the method that is annotated must **have same signature like the super class** method with same name !!! (PLEASE CHECK THIS)

In JAVA all NON STATIC methods are by default virtual !!!!!

In JAVA all non static methods are polymorphic ---- the method is RESOLVED as per the RUNTIME type of the object !!!!

A obj = new B();



_____

Overriding is ONLY applicable to non static methods in a hierarchy !!!

Overriding is not applicable to static methods !!!

_____

Overloading	Overriding
Method name is same for all forms/definitions	method name is same for all forms/definitions
Method parameters are different for each form	Method parameters are same for all forms
Can be done within a single class	Cannot be done in single class---at least 2 classes in a hierarchy
Applicable to static and non static methods	Applicable to ONLY non-static methods
Compile time polymorphism	Runtime polymorphism

_____

Object class has a method

public String toString()

This method returns the package qualified classname @ hex address in RAM

If we want to return some information about the class properties in the toString then we should override toString!!!

Object toString (classnmae@address)
|
|
Pen toString (peninfo)

_____

HW1 -- Override toString of following classes and check if that toString is called at runtime .

1. Circle
2. Point
3. Book
4. MyDate
5. Product
6. Person
7. Student
8. Employee
9. Patient
10. InternshipStudent

Class TestToString

Main

//pass all 10 objects of above classes one by one to check() and observe that toString of runtime type is called

Public void check(Object obj)

{

```
        Sysout (  obj.toString()  )
    }
```

Object class --------

```
 public boolean  equals ( Object obj )

    It compares two references and returns true if they are same else false
    {
        If ( this == obj )  return true;
         else return false;

    }
```

HW2

Override equals method for the following ----------

1. Circle (third)
2. Point (second)
3. Book (LAST)
4. MyDate (first)
5. Product (fourth)
6. Person (fifth)
7. Student (sixth)
8. Employee (seventth
9. Patient (eighth
10. InternshipStudent (nineth)

Write a User

```
    Main

    Public static void checkEqaulity(Object obj1, Object obj2)
    {


        If (obj1.equals(obj2) )
            Sysout  they are equal
        Else
           sysout they are not equal

    }
```
