7/7/2022

Run time Polymorphism (Dynamic binding)

→ RTP is also called as dynamic binding because the method call statement to a non static method is getting connected to the (binded) method defination at compile time but the decision is changing at run time.

method overriding

It is a type of run time polymorphism when both the Parent class as well as child class having same signature non static method at the time of execution with child object, child class method body overrides parent class method body. This is known as method overriding in java.

eg:

```
class parent
{
  void n()
  { sop(" parent class non static method");
  }
}
class child extends parent
{
  void n()
  { super.n(); // calling parent class method by using "super" keyword.
    sop(" child class non-static method");
  }
```

psvm ()

{ child ob = new child();

ob.h (); // child class method will execute

parent p = new parent();

p.h (); // since p is parent object
// parent class method will execute

}

}

Non premitive type casting.

→ Type casting means converting from one type to another type.

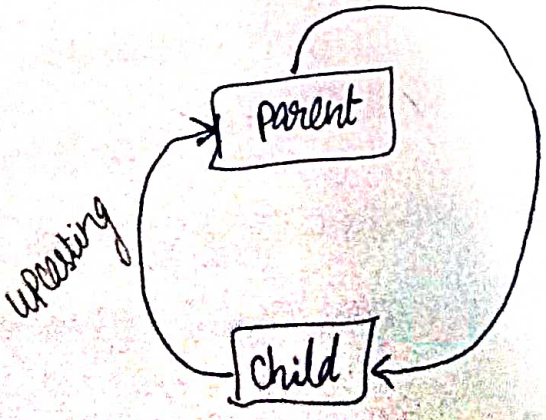
→ For performing premitive type casting the datatypes should be compatabile (similar)

→ For performing non-premitive type casting b/w the classes parent child relation should be there.

Non premitive type casting (2) types:

→ upcasting

→ down casting



Down casting the upcasted object.

Upcasting

→ Upcasting is the process of converting child type object into parent type

→ upcasting can be done implicitly.

→ once upcasted it is not possible to access child class members using the upcasted object. (overriden method can be accessed)

Down casting

→ Downcasting is the process of converting the upcasted object into child ~~caste~~ type again.

→ Down casting it cannot perform ~~dcp~~ implicitly, it should perform explicitly by use type caste operator.

→ It is not possible to perform downcasting without performing upcasting.

Note:

→ upcasting is performing to achieve generalization

→ downcasting is performing to achieve specialization.

→ If we tries to perform non-primitive type casting b/w unrelated classes (no parent child relation) we will get ~~class caste~~ classcastException.

eg:

```
class A
{ void example()
  { SOP("A class method");
    SOP("parent class is B");
  }
}
class B extends A
{
  void example()
  { SOP("B class method");
    SOP("child class is B");
  }
  void own()
{ SOP("child class method");
  SOP("hello from own method");
}
psvm()
{
// B ob = new B();
// ob.example(); // child class method will execute
   A a = new B(); // upcasting statement
    a.example();
  // a.own()// wiel produce compile time error because own() is not
    B obj = (B) a; // downcasting statement
    obj.own(); // wiel executing, because obj is a downcasted object.
  }
}
```