UNDERSTANDING OBJECT REFERENCE VARIABLE

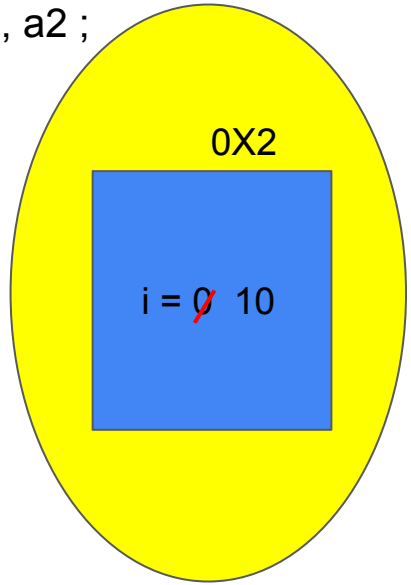
UNDERSTANDING REFERENCE VARIABLE

- We can copy the object reference from one reference variable to another reference variable, if they belong to the same family.

```
class Apple
{
  int i;
Apple(int i)
{
this.i=i;
}
}
```

```
class Carrot
{
   int x;
   Carrot(int x)
   {
    this.x=x;
}
}
```

```
Class Driver
{
    public static void main(String [] args)
   {
      Apple a1=new Apple(10) , a2 ;
      a2=a1;
      System.out.println(a1.i);
      System.out.println(a2.i)
}
}
```

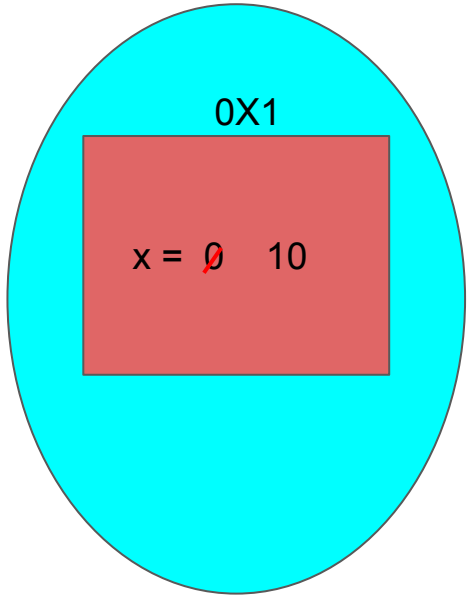


a1=0X2, a2=a1, therefore, a2=0X2
Two reference variables are pointing to the same object.

- An Object can be referred by any number of reference variable.
- We can verify weather 2 reference variables are pointing to the same or different object with the help of equality operator:-
    * If the equality operator returns ‘true’, it means other variables are pointing to the same object.
    * if it returns ‘false’, they are pointing to 2 different objects.

CASE 1:

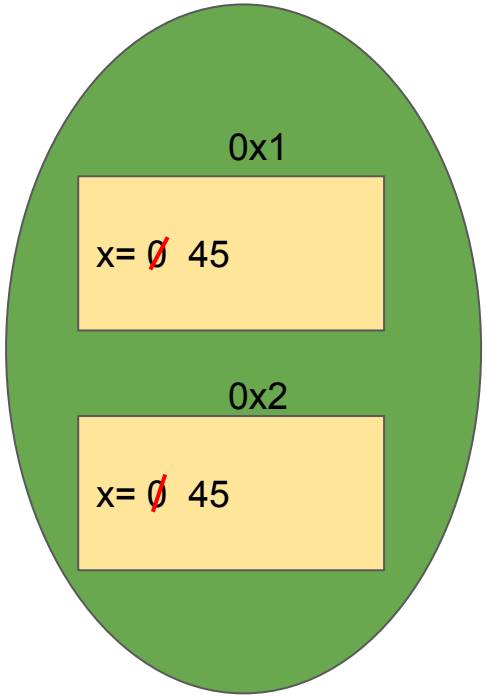
```
Carrot c1=new Carrot(77);
Carrot c2=c1;
System.out.println(c1.x);//77
System.out.println(c2.x);//77
System.out.println(c1==c2)//true
```



c1=0X2
c2=0X2

CASE 2

```
Carrot c1=new Carrot(45);
Carrot c2=new Carrot(45);
System.out.println(c1.x);//45
System.out.println(c2.x);//45
System.out.println(c1==c2);//false
```



c1=0x1

c2=0x2

- We can pass reference of an Object to method also.
- A method can return the reference of an Object.

NON PRIMITIVE TYPE CASTING

- The process of converting one non-primitive data type into another non-primitive data type is known as non-primitive data type casting
- It is also known as Derived Type casting.

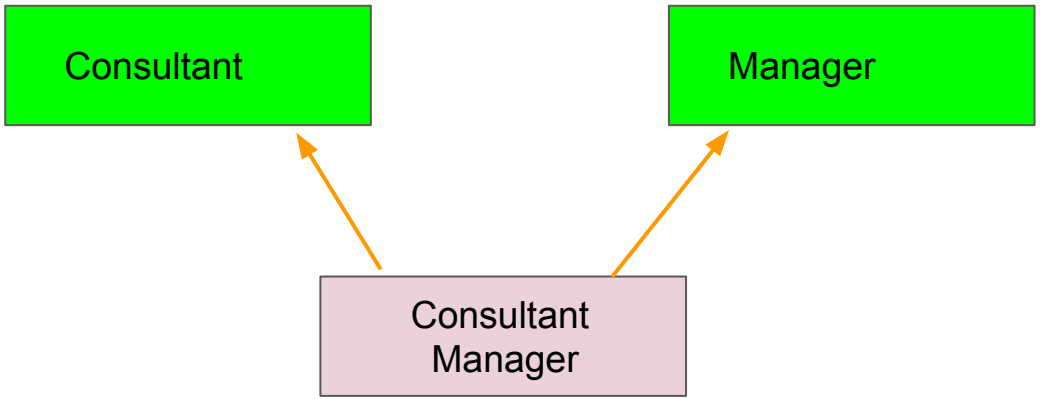
RULES TO ACHIEVE NON PRIMITIVE TYPE CASTING :

Non primitive type casting can be achieved only in the following situation:-

CASE 1:-There must be Is-A relation (Parent and child) exist between two reference.

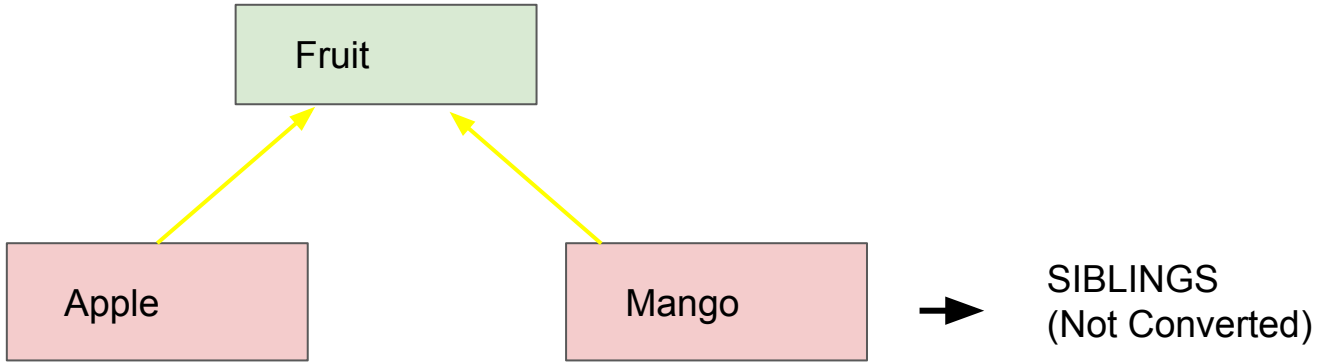
- Fruit can be converted to Apple and Apple can be converted to fruit.
- Vegetable can be converted to Carrot and Carrot can be converted to Vegetable.
- But Fruit and apple can’t be converted to Vegetable and carrot
- Vegetable and carrot can not be converted to Fruit and Apple.

CASE 2:- We can convert one type to another type, even though they don’t have ‘is-a’ relationship, but have a common child.(Multiple Inheritance)



- Consultant can be converted to Consultant Manager and Consultant Manager can be converted to Consultant.
- Manager can be converted to Consultant Manager and Consultant Manager can be converted to Manager.
- Consultant can convert to Manager .
- Manager can convert to Consultant.

CASE 3:-We cannot convert one reference type to another reference type, if they do not have ‘is-a’ relationship or they are not the parent of common child.

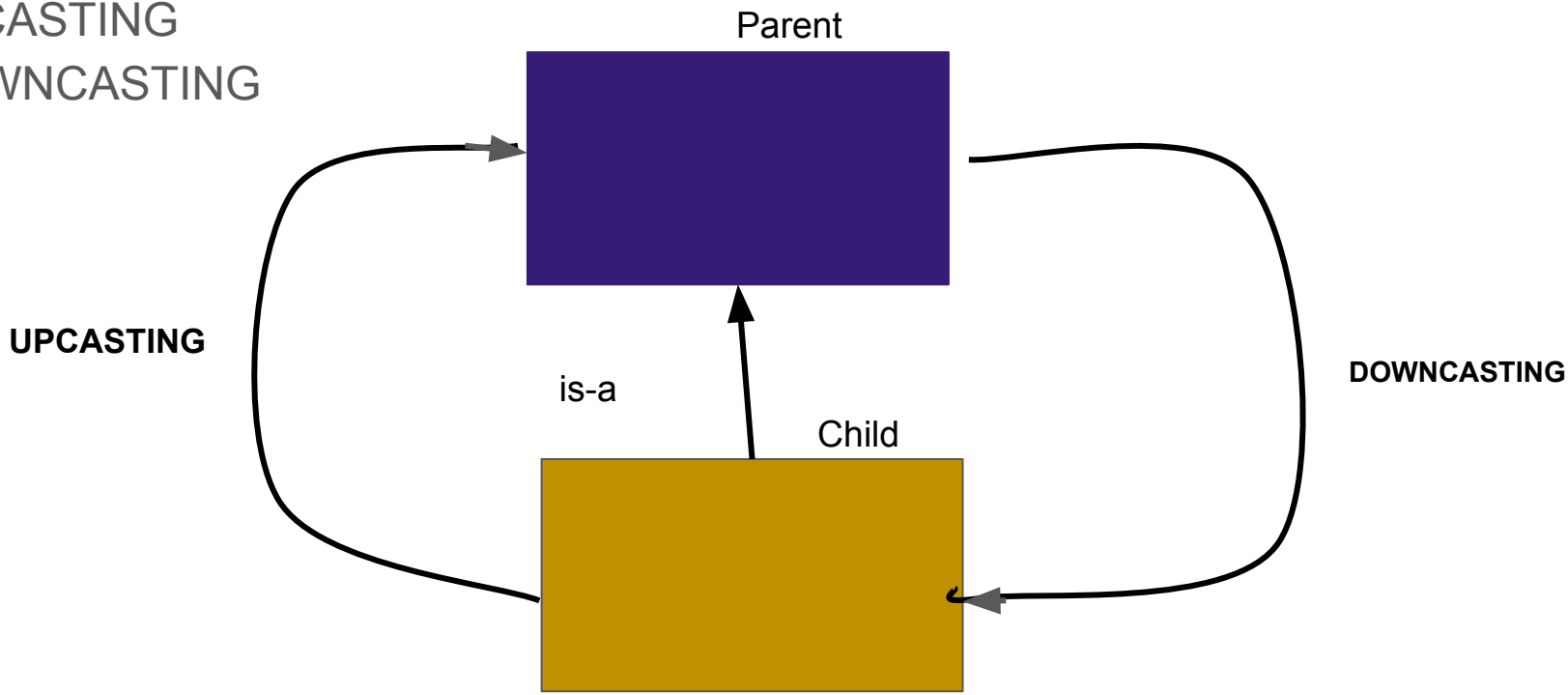


- Fruit can be converted to Apple and Mango
- Mango and Apple can be converted into Fruit.
- But Apple cannot be converted into Mango .
- Mango cannot be converted into Apple.

TYPES OF NON-PRIMITIVE TYPE CASTING

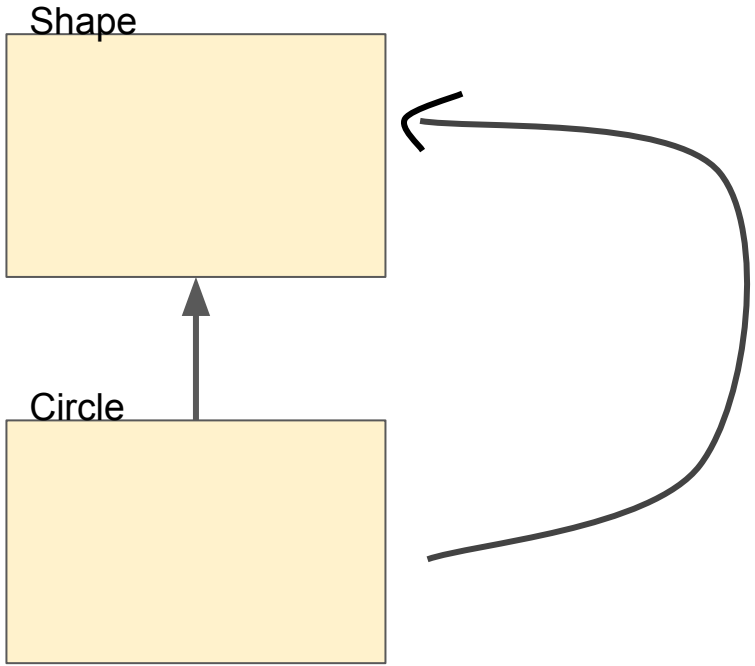
It is broadly categorized into 2 types:-

1) UPCASTING
2) DOWNCASTING



UPCASTING

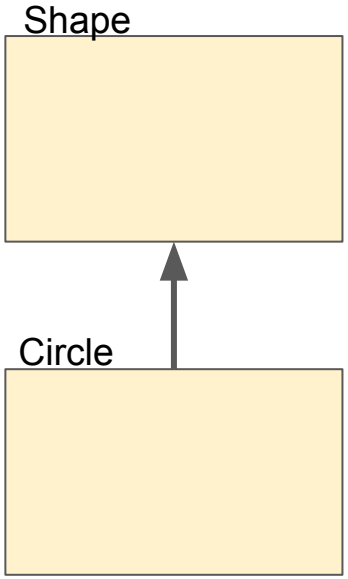
The process of converting child class reference type into parent type class reference type, is known as Upcasting.



NOTE :

- The upcasting is implicitly done by the compiler.
- It is also known as auto upcasting.
- Upcasting also done by explicitly with the help of typecast operator.
- Once the reference is upcasted we can’t access the members of child.

EXAMPLE



Circle c=new Circle();
Shape s=c;// UPCASTING

Note:-By using reference variable ‘s’ we can only access only the members of Shape.(Parent class)

WHY DO WE NEED UPCASTING ?

●It is used to achieve generalization.

●It helps to create generalized container so that the reference of any type of object can be stored.

EXAMPLE :

Case 1:-

```
Prime p ;
p= new Mini();//CTE
p = new Auto();//CTE
p = new Prime();//CTS
```

Description:-

- ‘p’ is not a generalized variable.
- Inside ‘p’ we can store only reference of Prime Object
- We can not store the reference of Mini and Auto

Case 2:-

```
Ola a;//generalized variable
a=new Mini();//CTS
a=new Prime();//CTS
a=new Auto();//CTS
```

Description:-

- ‘a’ is generalized variable as it belongs to superclass/parent class Ola type. Therefore,
- We can store the reference of its any type of its child object inside ‘a’.
- This is possible because of upcasting.

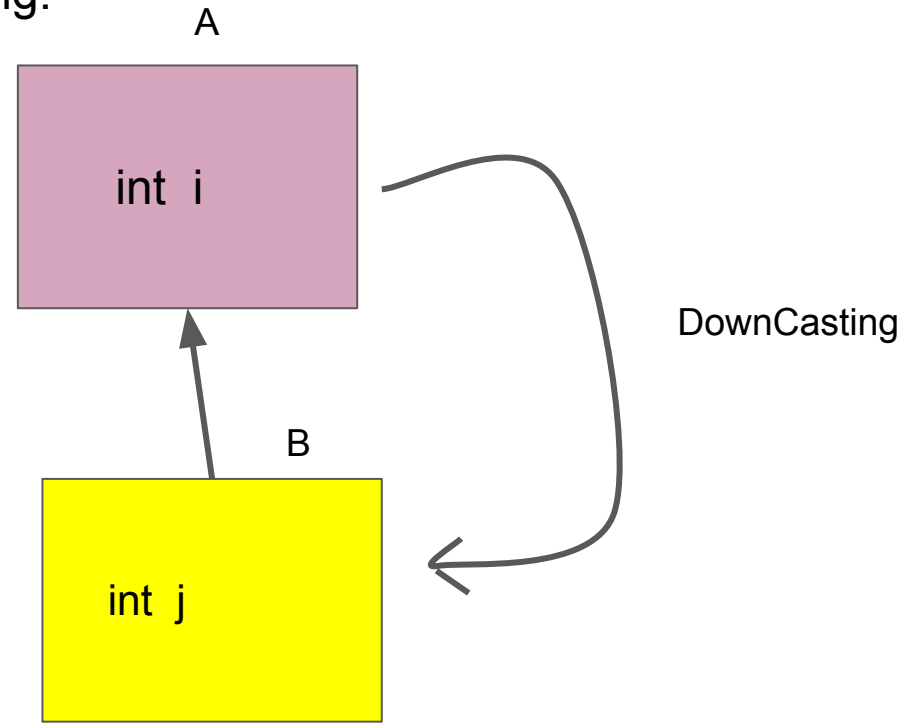
DISADVANTAGE OF UPCASTING

There is only one disadvantage of upcasting ,that is, once the reference is upcasted it’s child members can not be used.

Note:-In order to Overcome this problem we should go for DownCasting

DOWNCASTING

The process of converting parent class reference type into child class reference type is known as DownCasting.



Note:

- DownCasting is never done implicitly by the compiler.
- DownCasting can be done explicitly by the programmer with the help of typeCast operator.

PURPOSE OF DOWNCASTING

- If the reference is upcasted we cannot use the reference of subclass.
- To use the members of subclass we need to downcast the reference to subclass.

STEPS TO ACHIEVE DOWNCASTING :

STEP 1 :

Create a object for child

Child c = new Child();

STEP 2 :

Upcast Child reference type to parent class reference type.

Parent p = c ;

STEP 3 :

Downcast Parent class reference type to Child class reference with the help of Typecast operator.

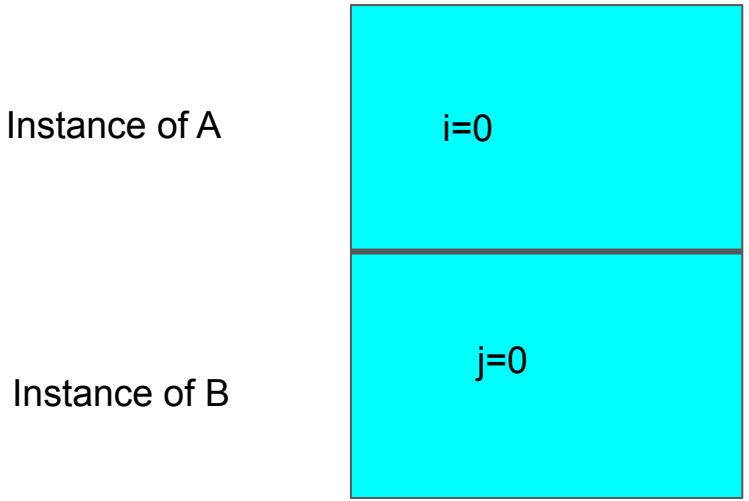
c = (Child)p;

Example

CASE 1:

```
A a=new B();//upcasting
System.out.println(a.i);//CTS
System.out.println(a.j);//CTE

B b=(B) a;//downcasting
System.out.println(b.j);//CTS
```



Case 2:-

A a=new A();

System.out.println(a.i);// CTS and RTS

System.out.println((B)a);//CTS but RUN TIME EXCEPTION

- While downcasting the object reference ‘a’ do not have the instance of the child class in which we are downcasting, then it will give RUN TIME EXCEPTION.
- In Case 2 we get runtime exception which is called ClassCastException.

CLASSCASTEXCEPTION

- It is a RuntimeException.
- Whenever we try to improperly cast a parent type reference to child type reference then we will get a ClassCastException.

When classCastException will occur :

When we try to directly convert parent type reference into child type reference.

```
Child c = (Child)new Parent(); //ClassCastException
```

instanceof operator :

- It is a binary operator
- It is used to test if an object is of given type.
- The return type of this operator is boolean.
- If the specified object is of given type then this operator will return true else it return false.

Syntax to use instanceOf operator :

(Object) instanceof (type)

EXAMPLE : new String() instanceof Object ;

NOTE :

It is always best practice to check whether the object is of given type or not to avoid ClassCastException.