Date
Page

Object Type casting

Parent ↓   child ↓

⇒ Object o = new String ("durga");

StringBuffer sb = (StringBuffer) o;
A   b   C   d

$\boxed{A\ b = (c)\ d\ ;}$

we are converting d type object to c type and assigning it to A type ~~object~~ reference variable.

⇒ Here compiler is going to check 2 rule (thing) and JVM will check 1$^{\text{rule}}$ (thing)

Rule-1 Compile Time checking - 1

⇒ Compiler will check the relation between c and d i.e they must have some relationship Either parent to child child to parent or same
or get C.E

Rule-2 Compile Time Checking - 2

C must be of Same type of A or child type of A.
or get C.E

Rule -3

JVM Checking -3 :-

⇒ The runtime ~~of~~ object type of d must be same as ~~typ~~ object type of c or it must be child of c

R.E : class cast exception

Example

$$\begin{tikzcd}
 & & & \text{Object} \arrow[dll] \arrow[drr, no head] & & & \\
 & \text{Base 1} \arrow[dl] \arrow[dr, no head] & & & & \text{Base 2} \arrow[dl] \arrow[dr, no head] & \\
\text{~~der1~~ Der1} & & \text{Der2} & & \text{Der3} & & \text{Der 4}
\end{tikzcd}$$

```
class Main
{
    public static void main (String [] args)
           Parent              child
    {
        Base2 b = new Der4(); ✓
        Der4 d = (Der4) b ; ✓
        Base1 b1 = (Base1) b ; ✗ → CE
        Base2 b2 = (Base2) b ; ✓
        Object o = (Der3) b ; ✗ → RE
        Base2 b3 = (Base1) b ; ✗ — CE
    }
}
```

```
class Object
{

}
```
X

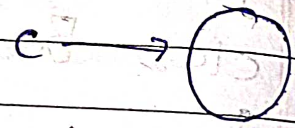
```
class Base1
{

}
class Base2
{
}
class Der1 extends Base1
{
}
class Der2 extends Base1
{
}
class Der3 extends Base2
{
}
Class Der4 extends Base2
{

}
```

$$\begin{tikzcd} \boxed{A} \text{ Parent} \\ \boxed{B} \text{ Child of A} \arrow[u] \\ \boxed{C} \text{ Child of C} \arrow[u] \end{tikzcd}$$

```
C c = new C();
```



(B) c → B type reference

Internal object type → C type runtime object

(A)(B) c) → A type reference

Internal object Type → C type runtime object

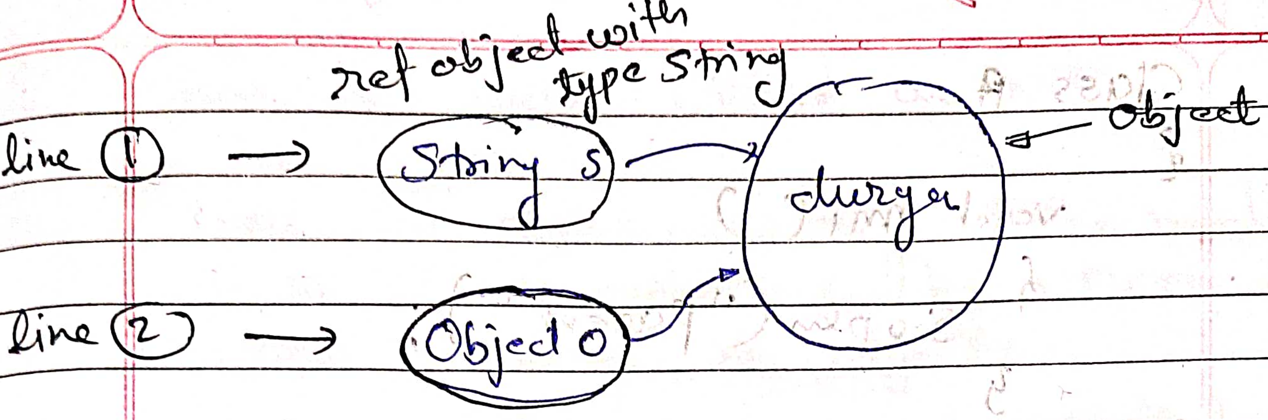
* Internal things in type casting

Example

```
String s = new String("durga");   — 1
Object o = (object) s;            — 2
```

*

⇒ In object type casting we are not going to creat any new object, for the existing object we are trying to provide a new Reference variable



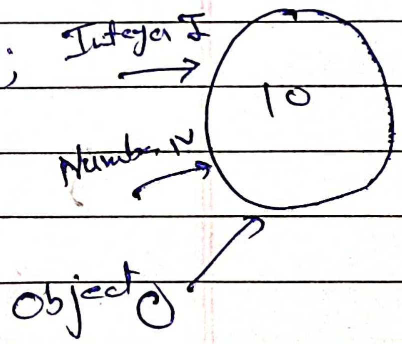
we are not creating a new object just we are changing the reference type of the object from String to Object

Program to check above thing!

```
class Test
{
        p.s.v.m (String [] args)
        {  String s = new String ("durga");
           Object o = (Object) s;
        }  s.o.p.ln (s == o); // true
}
```

Examples

```
  Integer I = new Integer (10);
Number N = (Number) I;
Object O = (Object) N;
```



```
check
  ↘ s.o.p.ln (I == O); // true ;
  s.o.p.ln (N == O); // true .
  s.o.p.ln (I == N); // true
```

Date
Page

```
Class A
{
        void m1( )
        {   s.opln("parent");
        }
}
class B extends A
{
        void m2( )
        {
          s.opln("child");
        }
}
class Test
{
    p.s.v.m( String [] args)
    {
         B b = new B();
         b.m1(); → ✓
         b.m2(); → ✓
         (A) b.m1(); ✓
         (A)b.m2(); ✗   C.E
    }
```

parent class reference

child class runtime object

```
A a = new B();
  a.m1();
```

can call parent class method

```
A a = new B();
a.m2();
```

cannot call child class method with parent class reference for method hiding Method resolution is always taken care by compiler based on Reference Object

Example with Method overriding

* Note; In Method overriding Method resolution is always taken care by JVM based on run time ~~boo~~ object

~~for~~ method hiding

```
class A                              static void m1 ( )
{     ~~void m1 ( )~~
      { sopln ("A") ;
}     }

Class B extends A
 {
      ~~void m1 ( )~~                ~~Static void m1( )~~
      { sopln ("B") ;
 }    }

class C extends B
{                                    ~~Static void m1( )~~
      void m1 ( )
      { sopln ("C") ;
      }
}

Class Test
{
      p.s.v.m(String [] args).
      {
Ans for
Method Hiding
           C c = new C( ) ;
C    <--   c.m1 ( ) ; // C
           B b = new C ( ) ; // ((B)c) .m1() ;
B    <--   b.m1 ( ) ; // C
           A a = new C ( ) ; // ((A)c) .m1() ;
A    <--   a.m1 ( ) ; // C
```