Type Casting

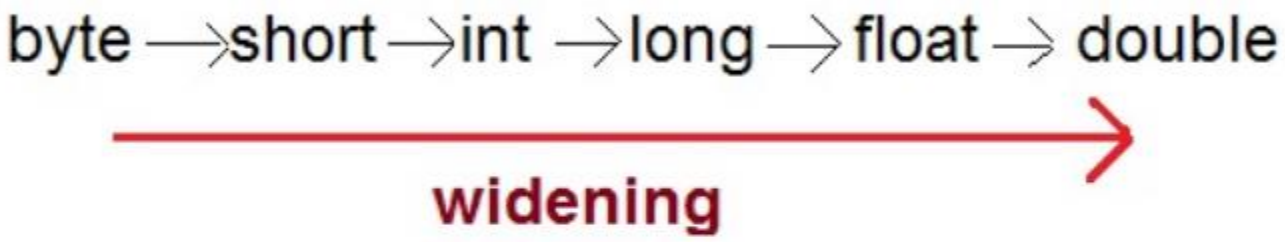
Table 1: List of Java's primitive data types

Type	Size in Bytes	Range
byte	1 byte	-128 to 127
short	2 bytes	-32,768 to 32,767
int	4 bytes	-2,147,483,648 to 2,147,483, 647
long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	approximately ±3.40282347E+38F (6-7 significant decimal digits) *Java implements IEEE 754 standard*
double	8 bytes	approximately ±1.79769313486231570E+308 (15 significant decimal digits)
char	2 byte	0 to 65,536 (unsigned)
boolean	not precisely defined*	true or false

Type Casting

In Java, type casting is classified into two types,

- Widening Casting(Implicit)



- Narrowing Casting(Explicitly done)

```
int i = 100;

long l = (long) i;

float f = (float) i;
```

```
int i = 100;

long l = i;  //no explicit type casting required

float f = i;  //no explicit type casting required
```

```
int  i  = 100;

long a1  =  i;

long a2  =  (long) i;
```

```
int     a = 100;

char  b = (char)a;
```

✓

```
int     a = 100;

char  b = a;
```

✗

```
double d = 100.04;

long l = (long)d;  //explicit type casting required

int i = (int)l;        //explicit type casting required
```

Object Creation

```
class A
{

}
class B extends A
{

}
```

A

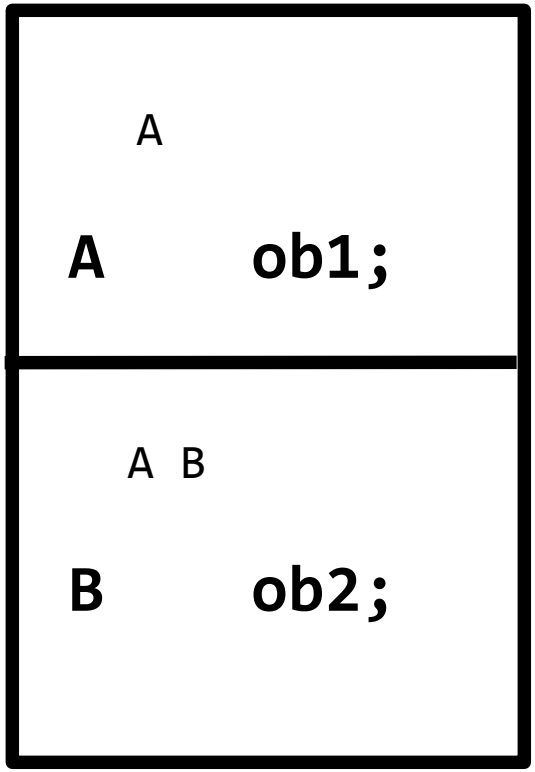
new A();

A B

new B();

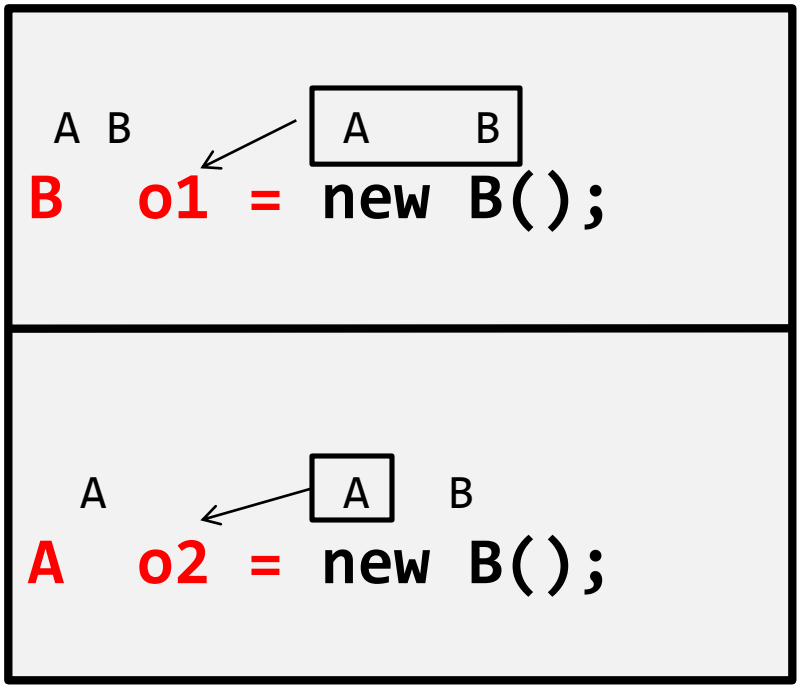
Reference Creation

```
class A
{

}
class B extends A
{

}
```



Class Casting

```
class A
{

}
class B extends A
{

}
```



Class Casting

```
class A
{
}
class B extends A
{
}
```

```
A s1 = new B();

B s2 = (B)s1;
```

Inheritance

Up-casting

```
A s1 = new B();

B s2 = (B)s1;
```

Downcasting

A class **Object** may be under
same-class reference or **base**-class reference

```
B  r1 = new B();

A  r2 = new B();
```

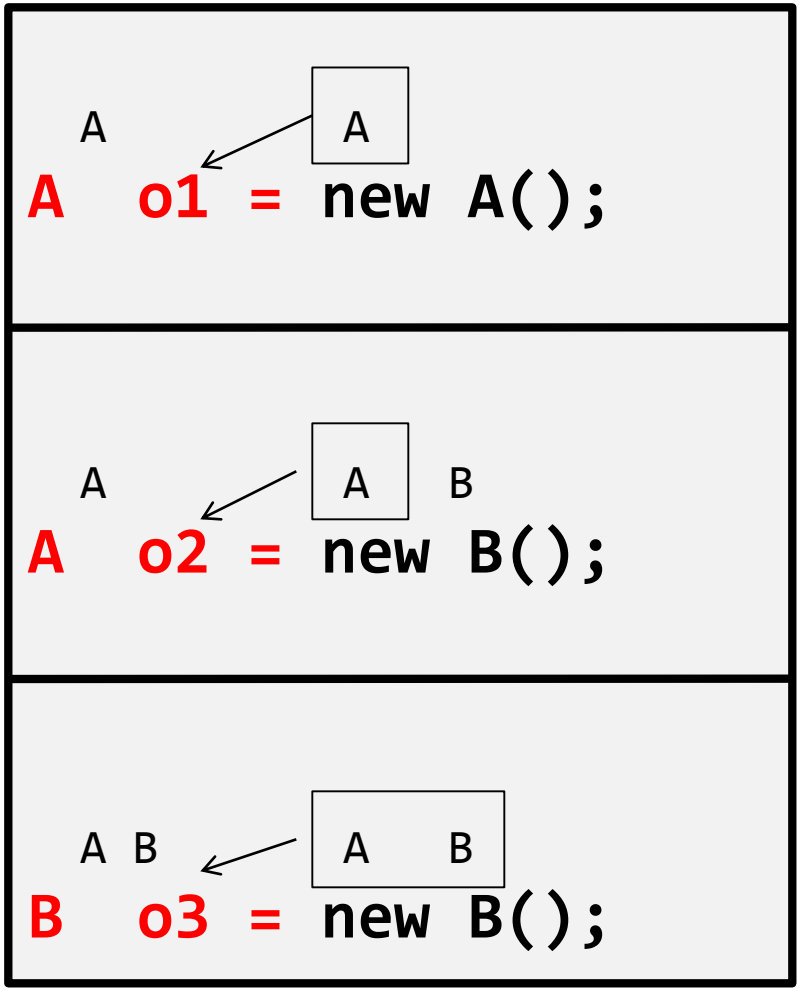
A class **Reference** may contain
same-class Object or **sub**-class Object

```
A  r1 = new A();

A  r2 = new B();
```

```
A   r1 = new B();

A   r2 = new B();
```

Inheritance

```
class A
{

}
class B extends A
{

}
```



```java
class A{

    int x;
    void test(){
        System.out.println(" X  : "+x);
    }

}

class B extends A{

    int y;
    void show(){
        System.out.println(" X  : "+x+" Y  : "+y);
    }

}
```

```
B ob1 = new B();

A ob2 = new B();

A ob3 = (A)new B();

A ob4 = ob1;

A ob5 = (A)ob1;
```

```
B ob1 = new B();

A ob2 = ob1;

A ob3 = (A)ob1;

B ob4 = ob2;

B ob5 = (B)ob2;
```

```
B ob1 = new B();

A ob2 = ob1;
```

incompatible types: A cannot be converted to B

(Alt-Enter shows hints)

```
B ob4 = ob2;

B ob5 = (B)ob2;
```

```
class A
{
}
class B
{
}
class C
{
}
```

O A

```
new A();
```

O B

```
new B();
```

O C

```
new C();
```

```
class A
{
}
class B
{
}
class C
{
}
```

```
A o1 = new A();

B o2 = new B();

C o3 = new C();
```

Tightly Coupled

```
class A
{
}
class B
{
}
class C
{
}
```

```
A o1 = new A();

B o2 = new B();

C o3 = new C();
```

```
class A
{
}
class B
{
}
class C
{
}
```

```
                    O    A
Object o1 = new A();

                    O    B
Object o2 = new B();

                    O    C
Object o3 = new C();
```

Loosely Coupled

```
class A
{
}
class B
{
}
class C
{
}
```

```
                    O    A
Object o1 = new A();

                    O    B
Object o2 = new B();

                    O    C
Object o3 = new C();
```

Inheritance

```
class A
{
}
class B
{
}
class C
{
}
```

```
                    O    A
Object s1 = new A();

A      s2 = (A)s1;
```

Inheritance

```
class A
{
}
```

Up-casting

```
Object s1 = new A();

A       s2 = (A)s1;
```

Down-casting

https://www.javatpoint.com/java-tutorial

- Java Object Class
  - Java OOPs Concepts
  - Naming Convention
  - Object and Class
  - Constructor
  - static keyword
  - this keyword
- Java Inheritance
  - Inheritance(IS-A)
  - Aggregation(HAS-A)

Java Polymorphism

- Method Overloading
- Method Overriding
- Covariant Return Type
- super keyword
- Instance Initializer block
- final keyword
- Runtime Polymorphism
- Dynamic Binding
- instanceof operator

Java Abstraction

- Abstract class
- Interface
- Abstract vs Interface