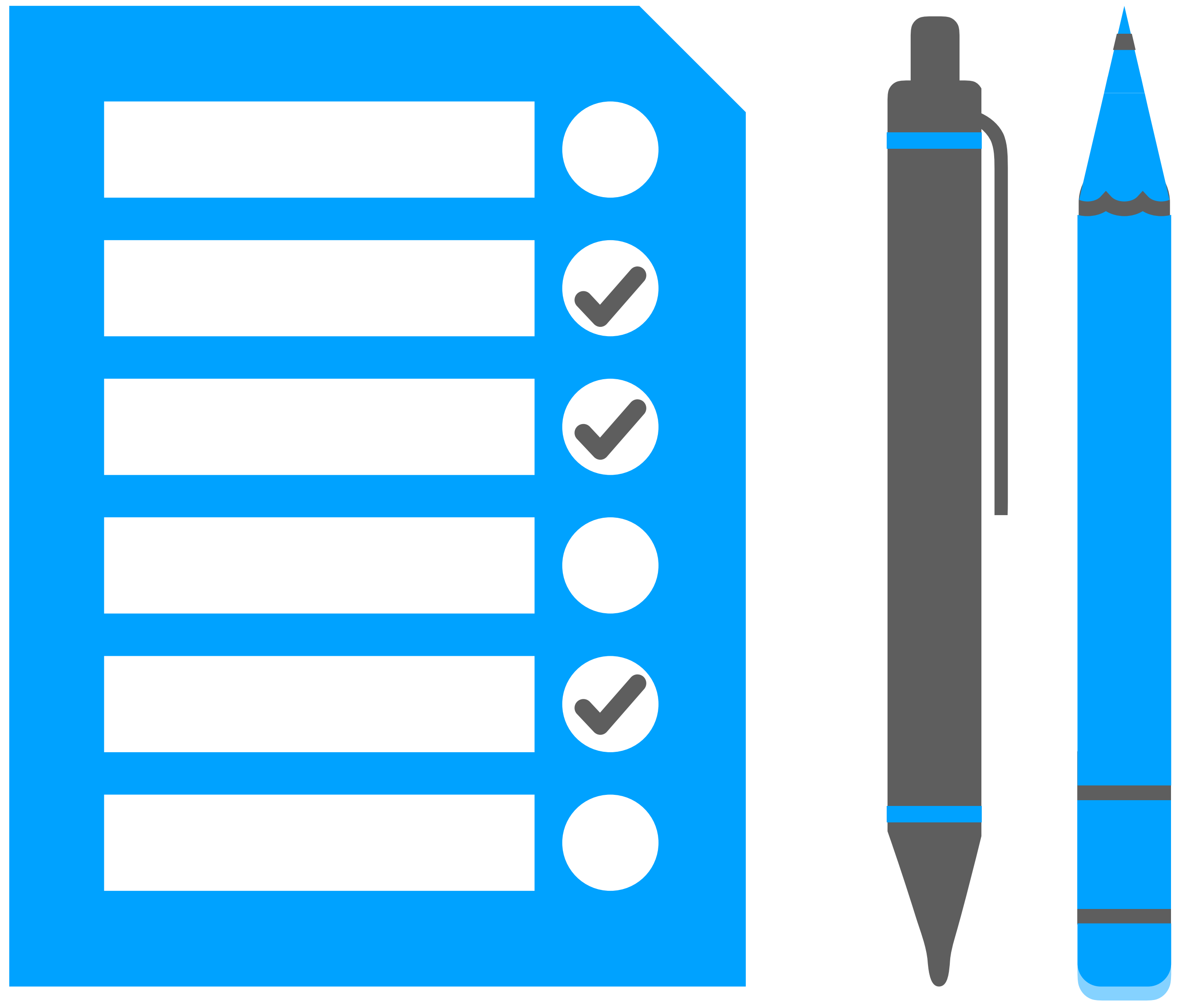
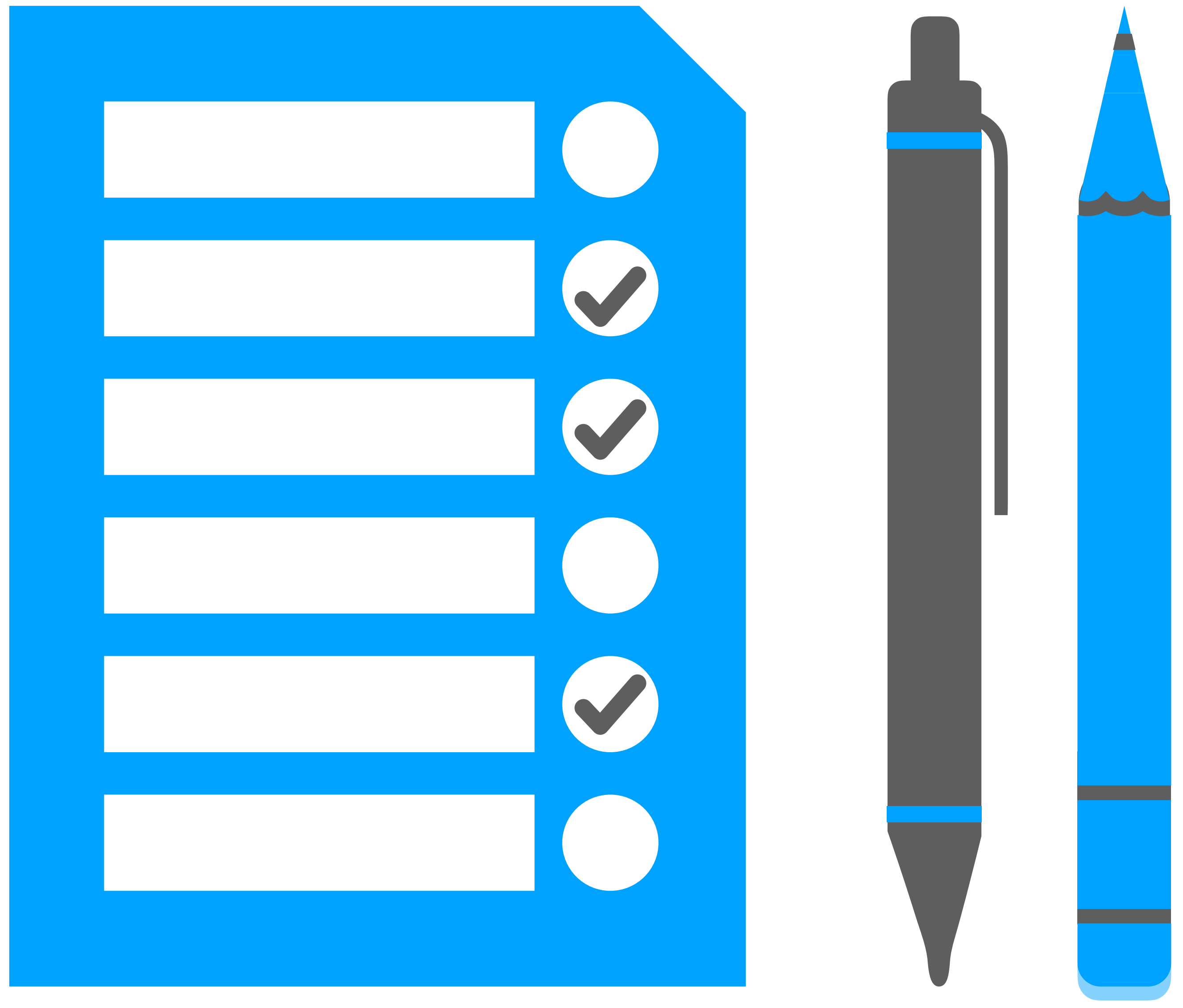
Type Casting



- Review casting primitive types
- Understand the need for casting
- Casting Reference Type

After today's session you should be able to:



- Understand the need for casting
- Perform casting for primitive types
- Perform Cast reference types
- Understand and avoid class

Type Casting

Converting one type to another

Implicitly or Explicitly

Types in Java

- Primitive Types
- Reference Type

Primitive Type

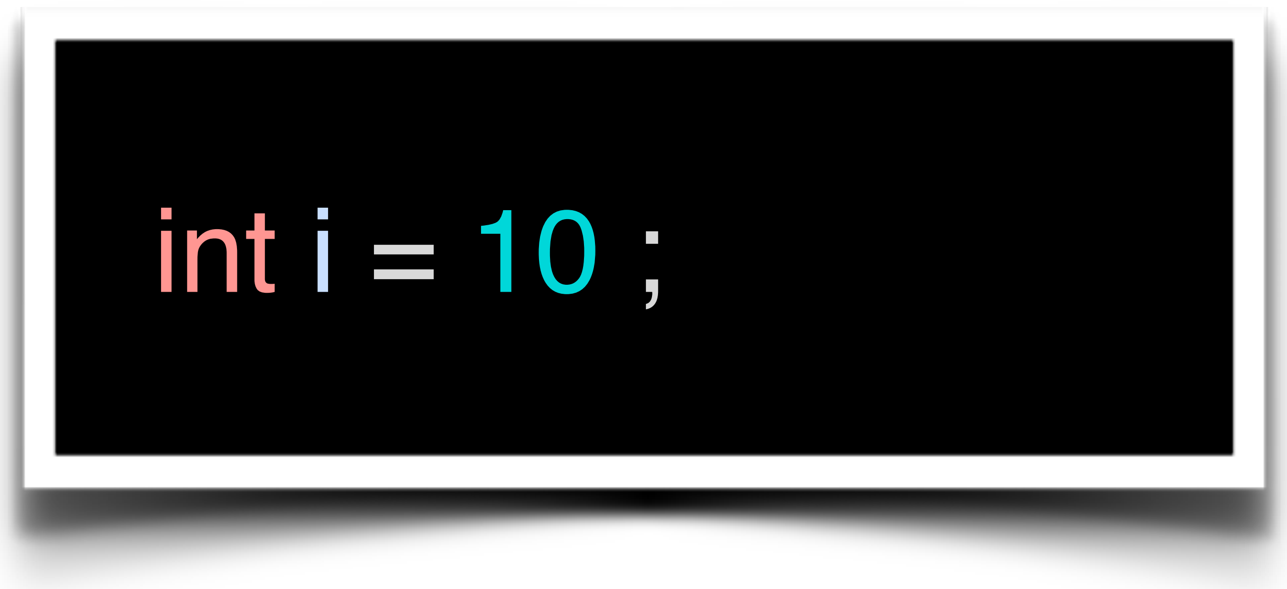
boolean

character

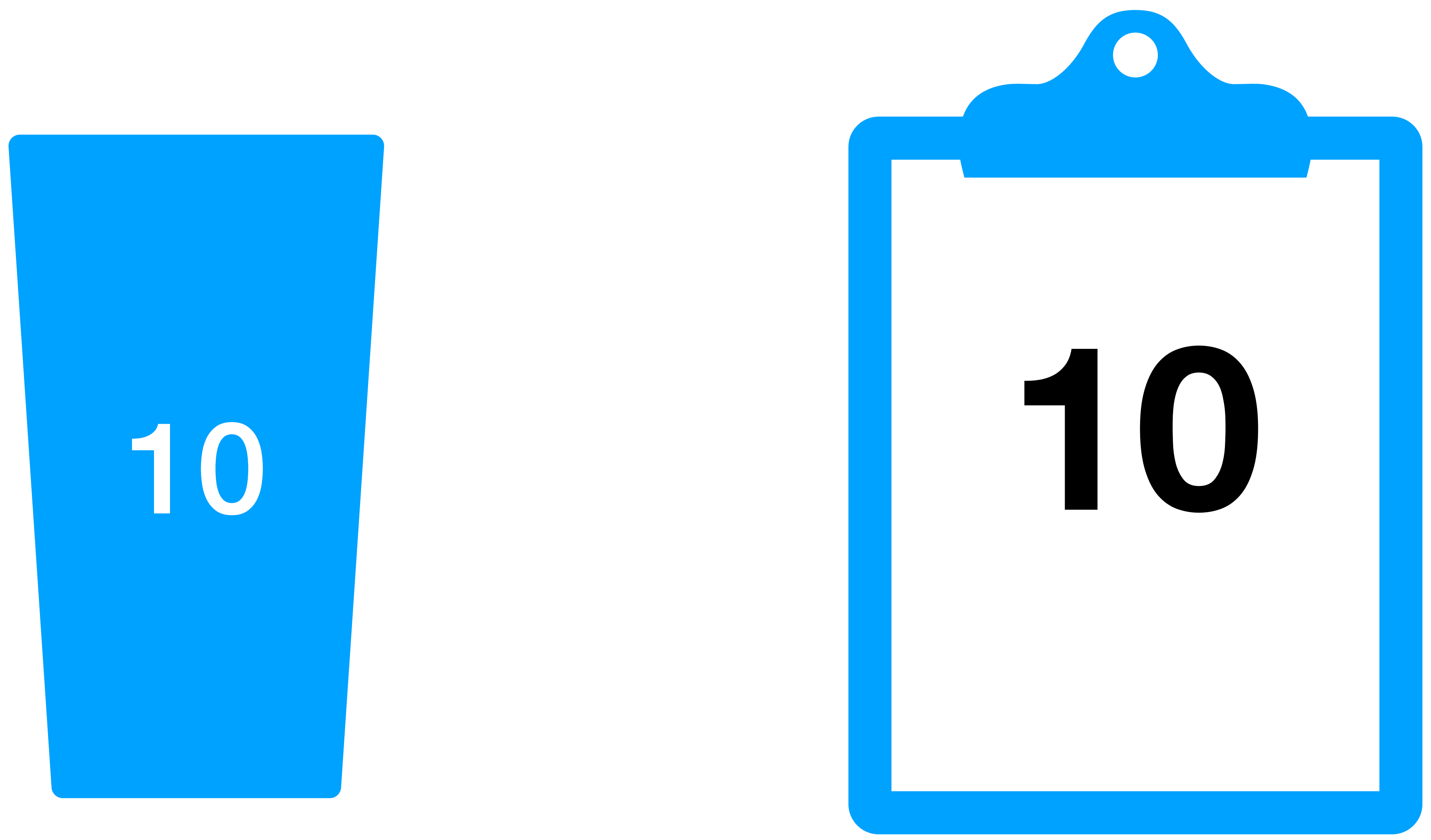
byte —> short—> int—> long—> float—> double

Be Careful , Bear Shouldn’t Ingest Large Fluffy Dog

Primitive Variable

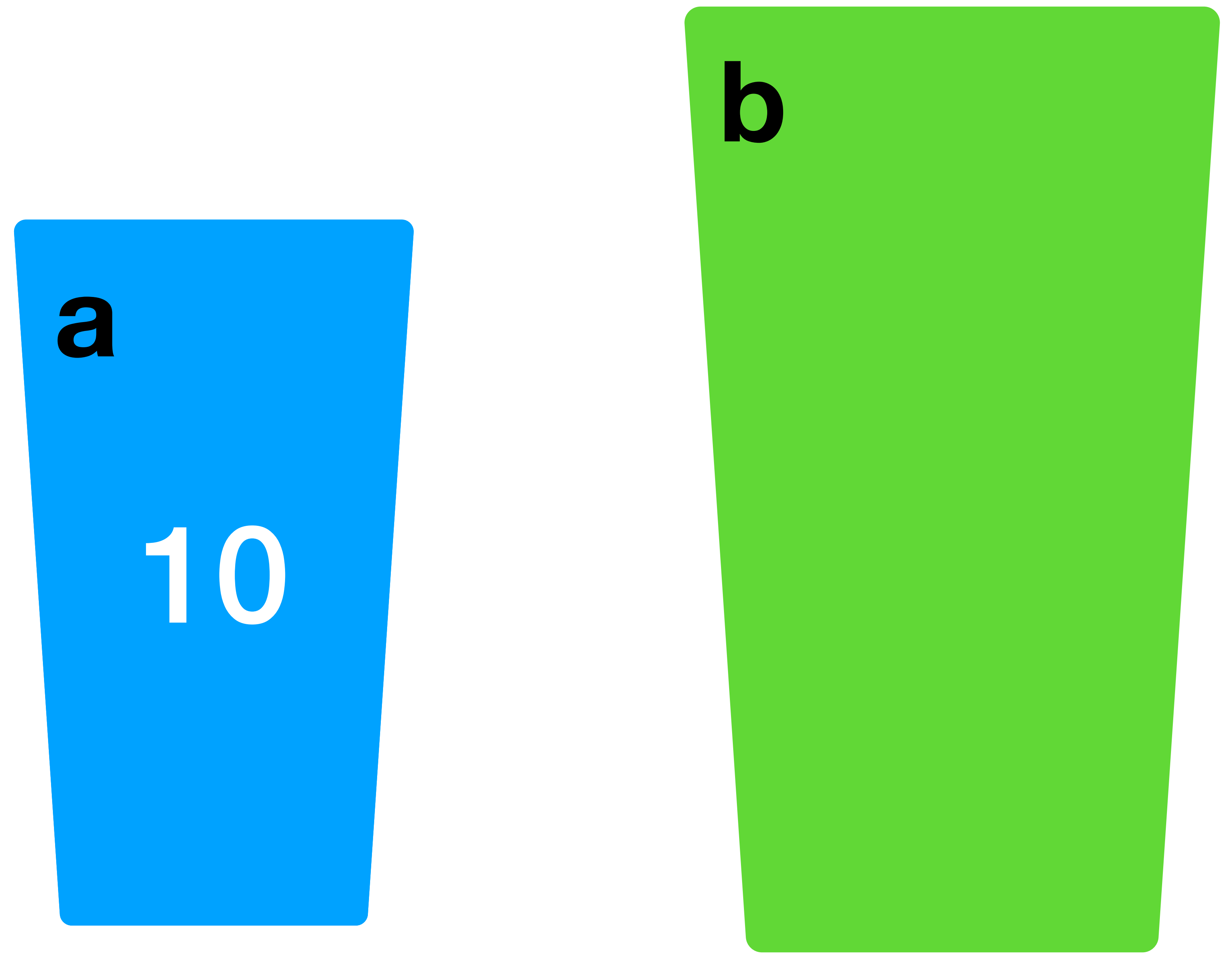


```
int i = 10 ;
```



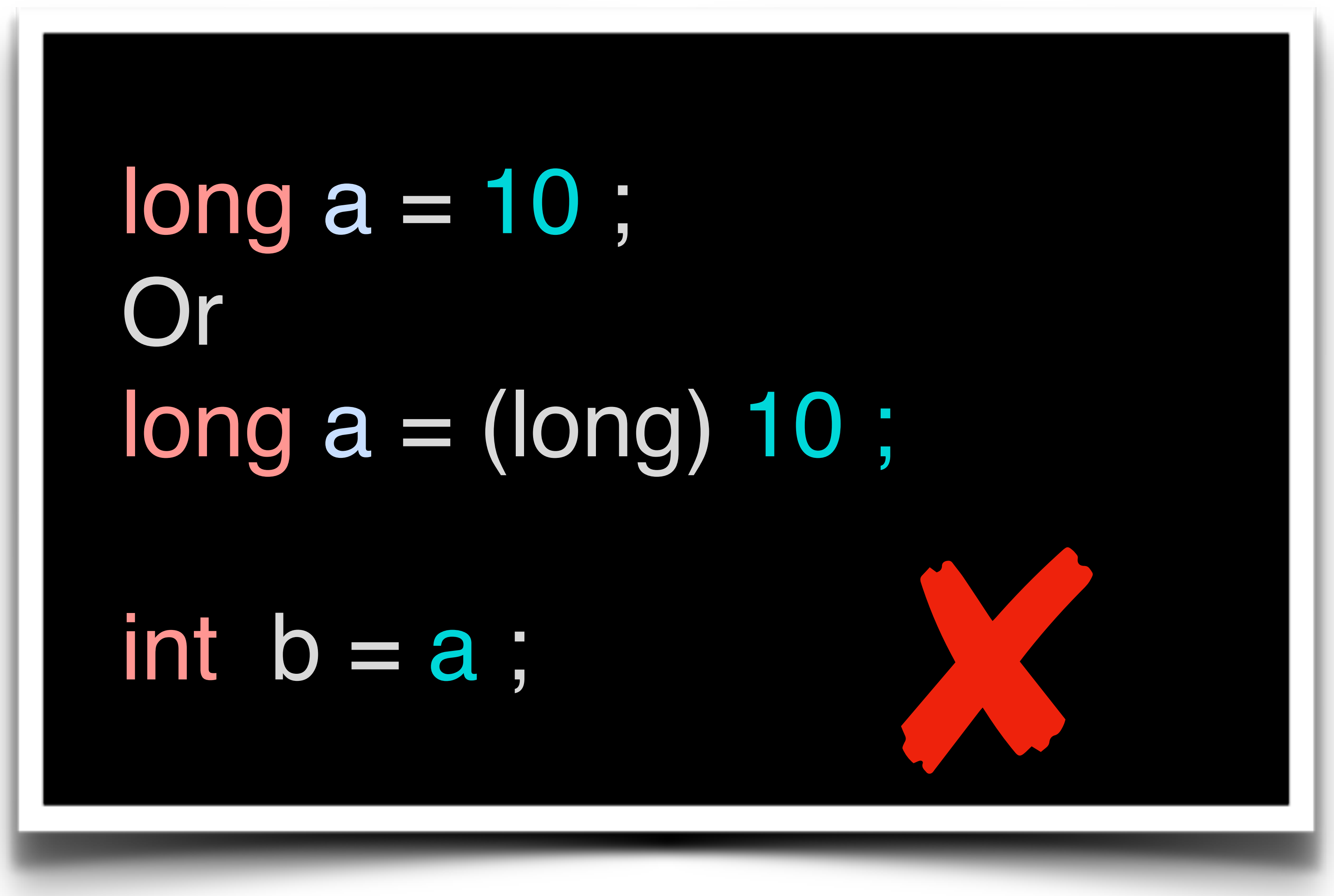
Primitive Variable

```
int a = 10 ;
long b = a ;
```



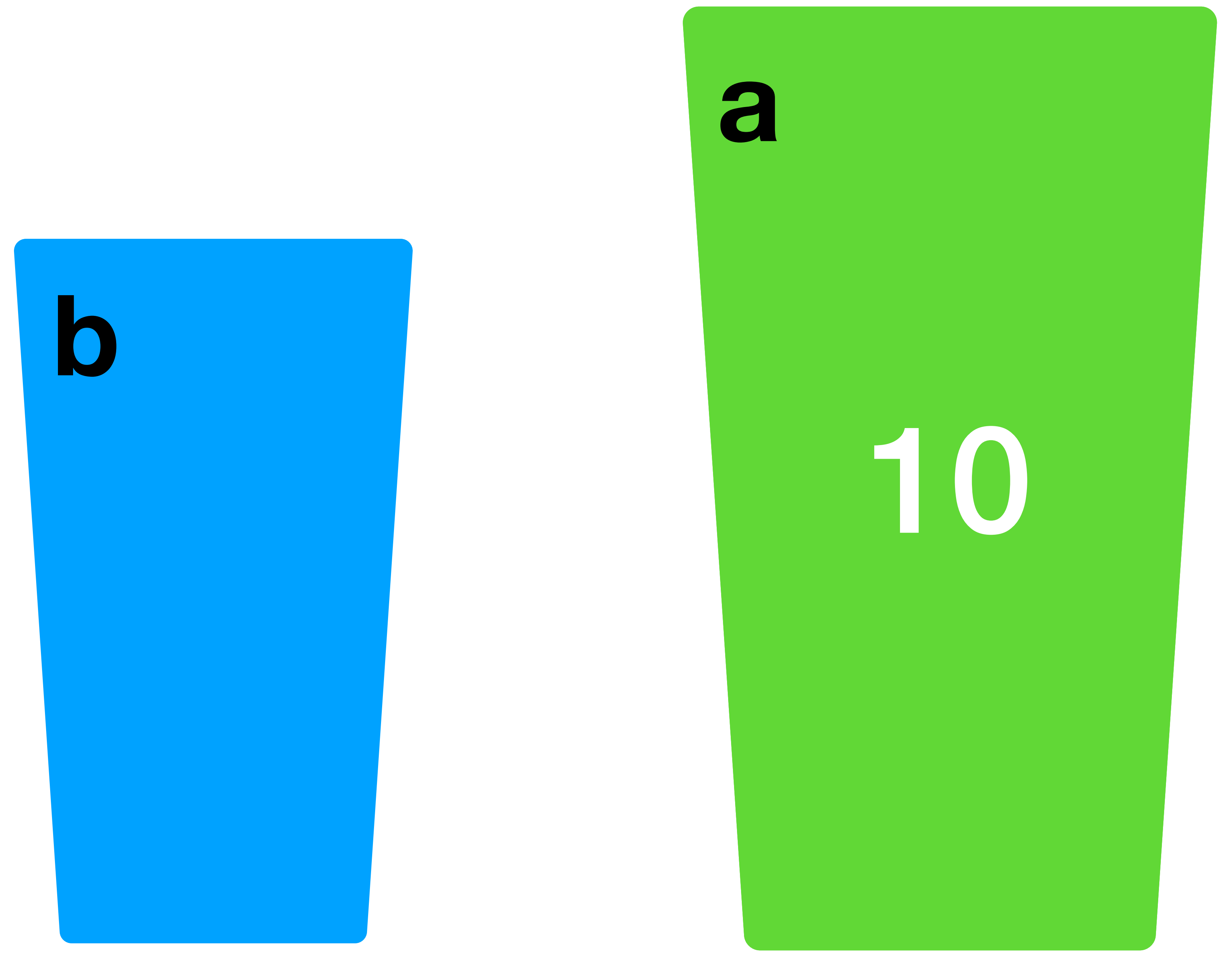
Widening Primitive Conversions happen implicitly

Primitive Variable



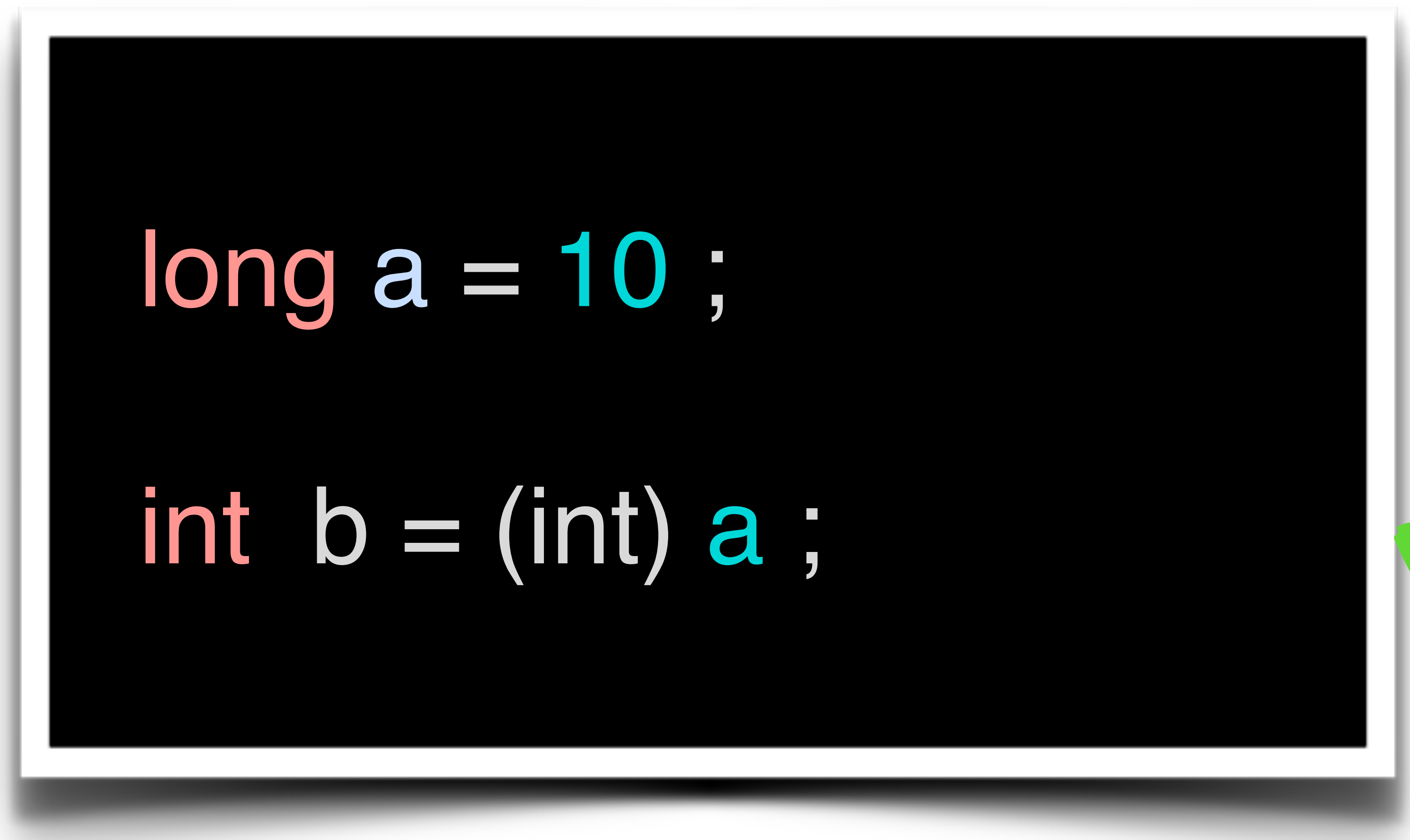
```
long a = 10 ;
Or
long a = (long) 10 ;

int  b = a ;
```



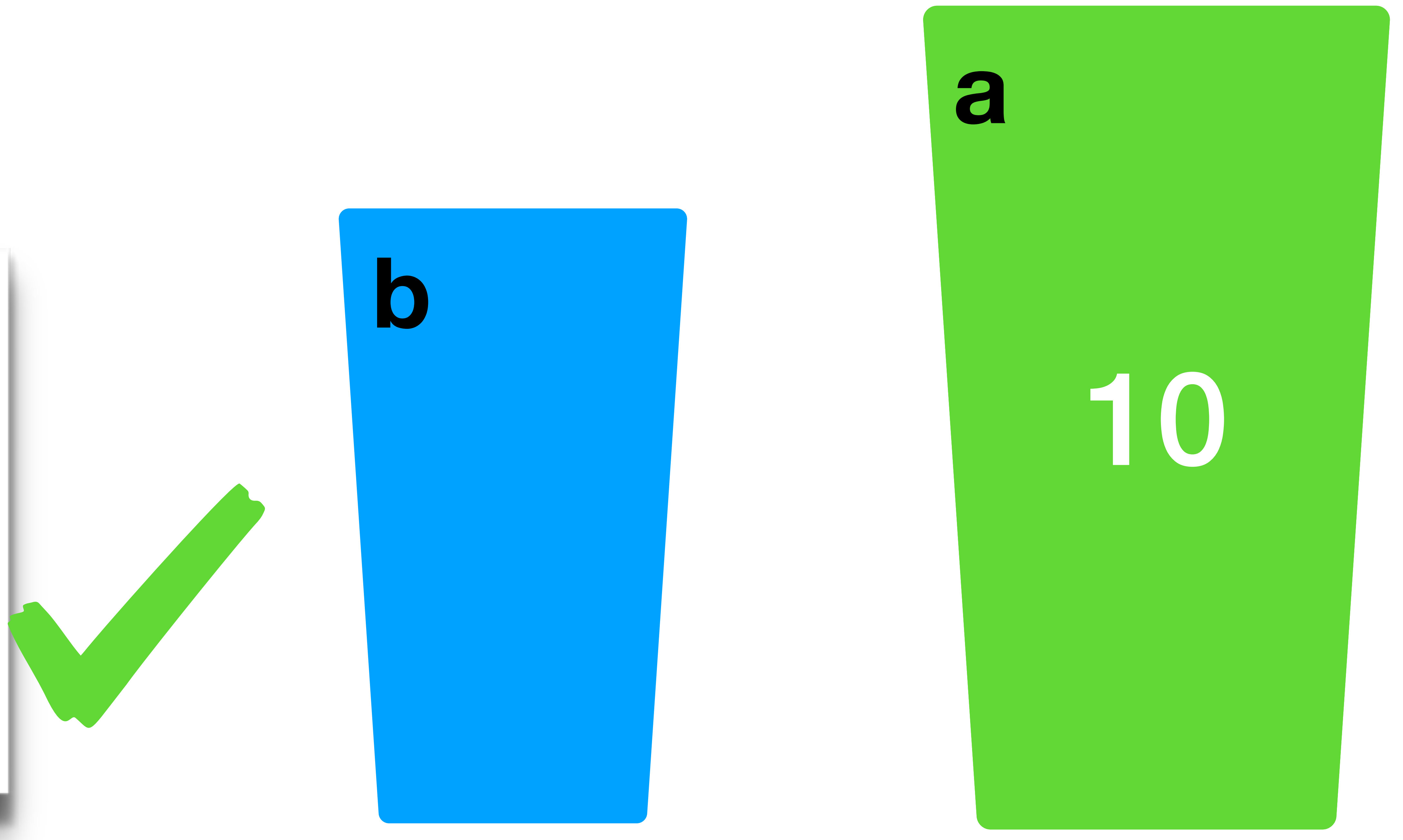
Narrowing Primitive Conversions has to be explicit cast

Primitive Variable



```
long a = 10 ;

int  b = (int) a ;
```



Narrowing Primitive Conversions has to be explicit cast

Type of casting

- **Upcasting**

Casting from a subclass to a superclass is called upcasting. Typically, the upcasting is implicitly performed by the compiler.

- **Downcasting**

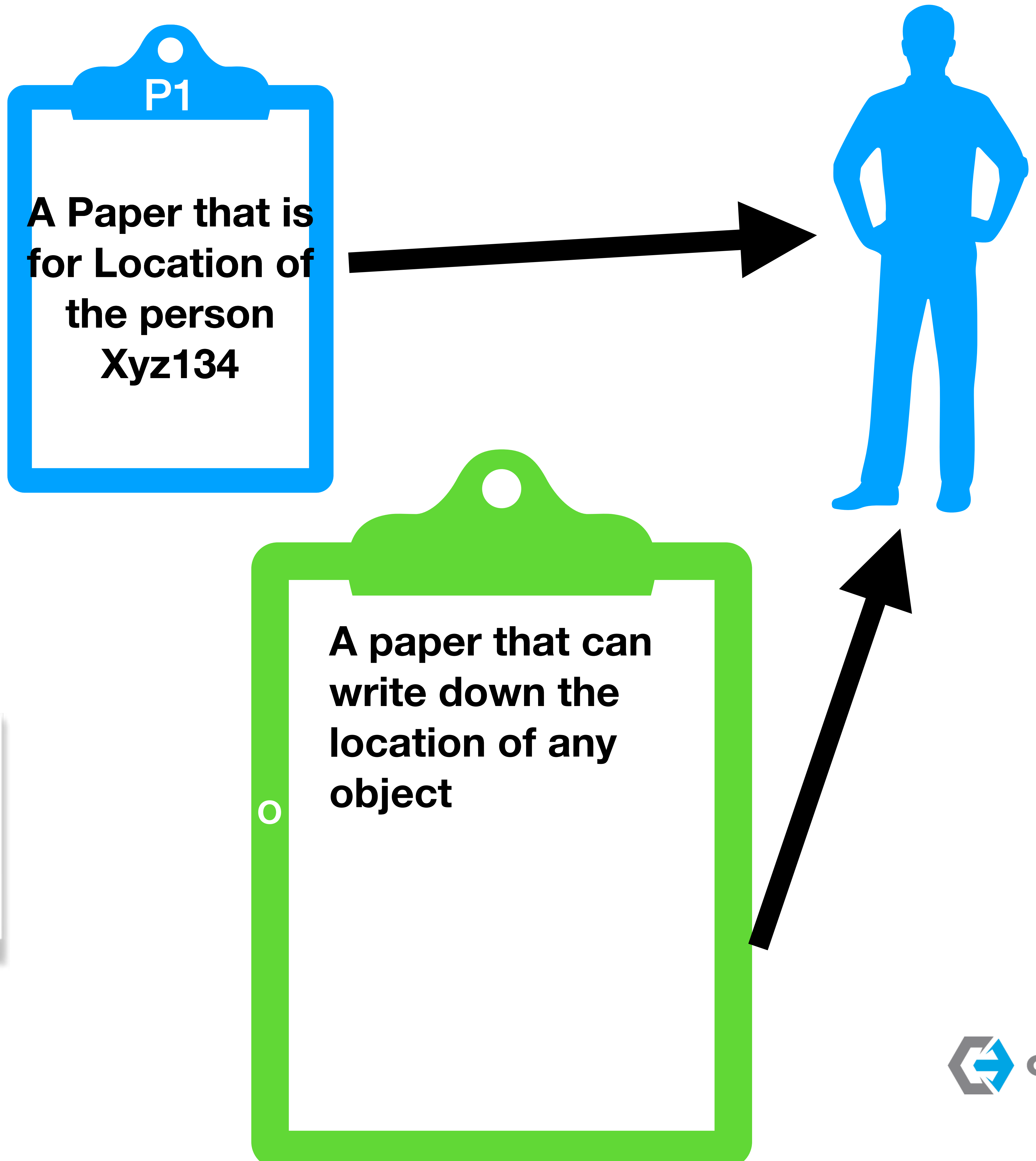
Casting from a superclass type to subclass type is called downcasting. The upcasting is has to be performed by the programmer explicitly.

Reference Variable Upcasting

```
Person p1 = new Person() ;

Object o = p1 ;
Or
Object o = (Object) p1 ;
```

Child Reference casted to Parent reference happen implicitly



P1
A Paper that is for Location of the person Xyz134
o
A paper that can write down the location of any object

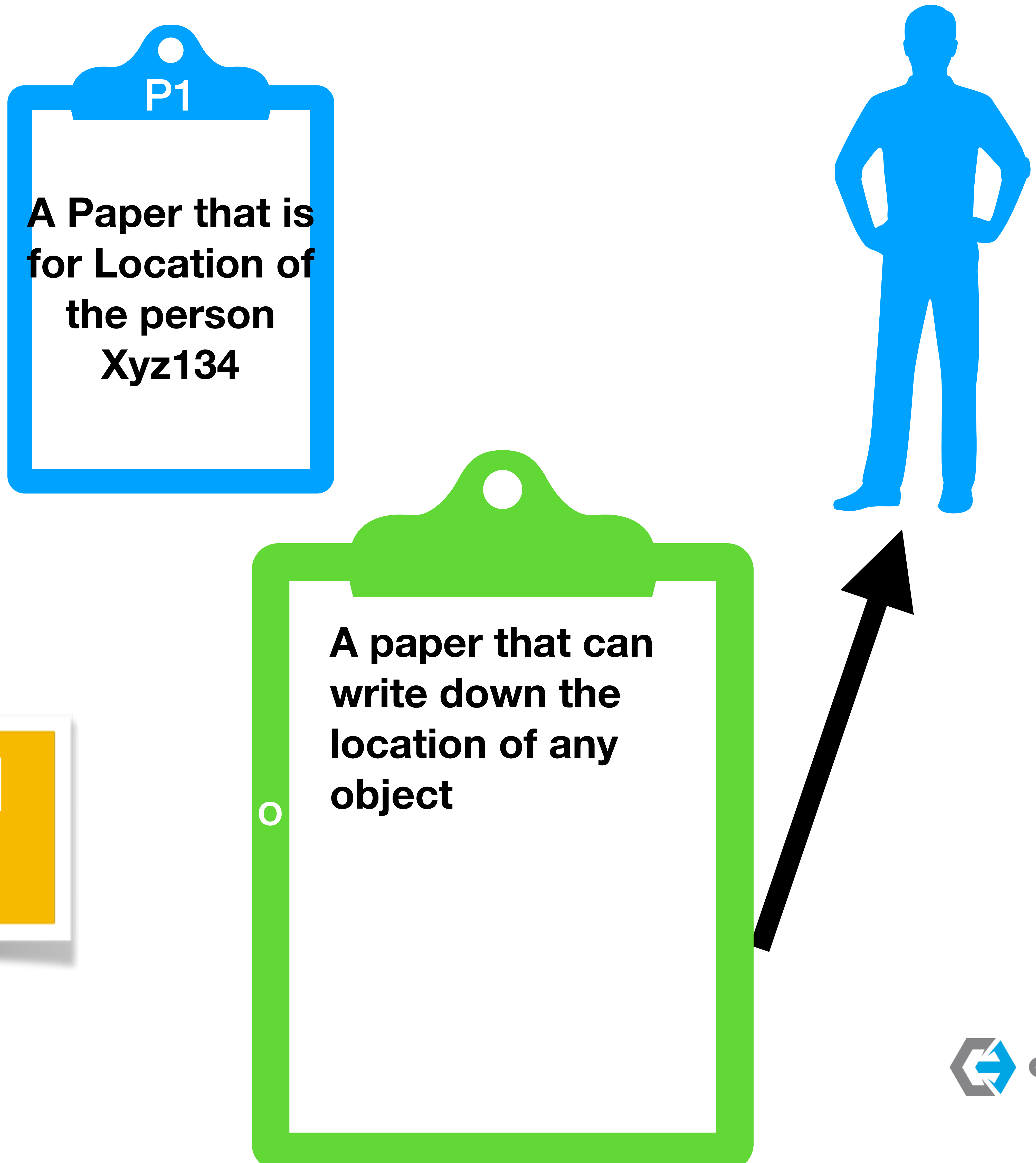
CYBERTEK

Reference Variable Downcasting

```
Object o = new Person() ;

Person p1 = o;
```

Parent Reference casted to Child reference has to be explicit

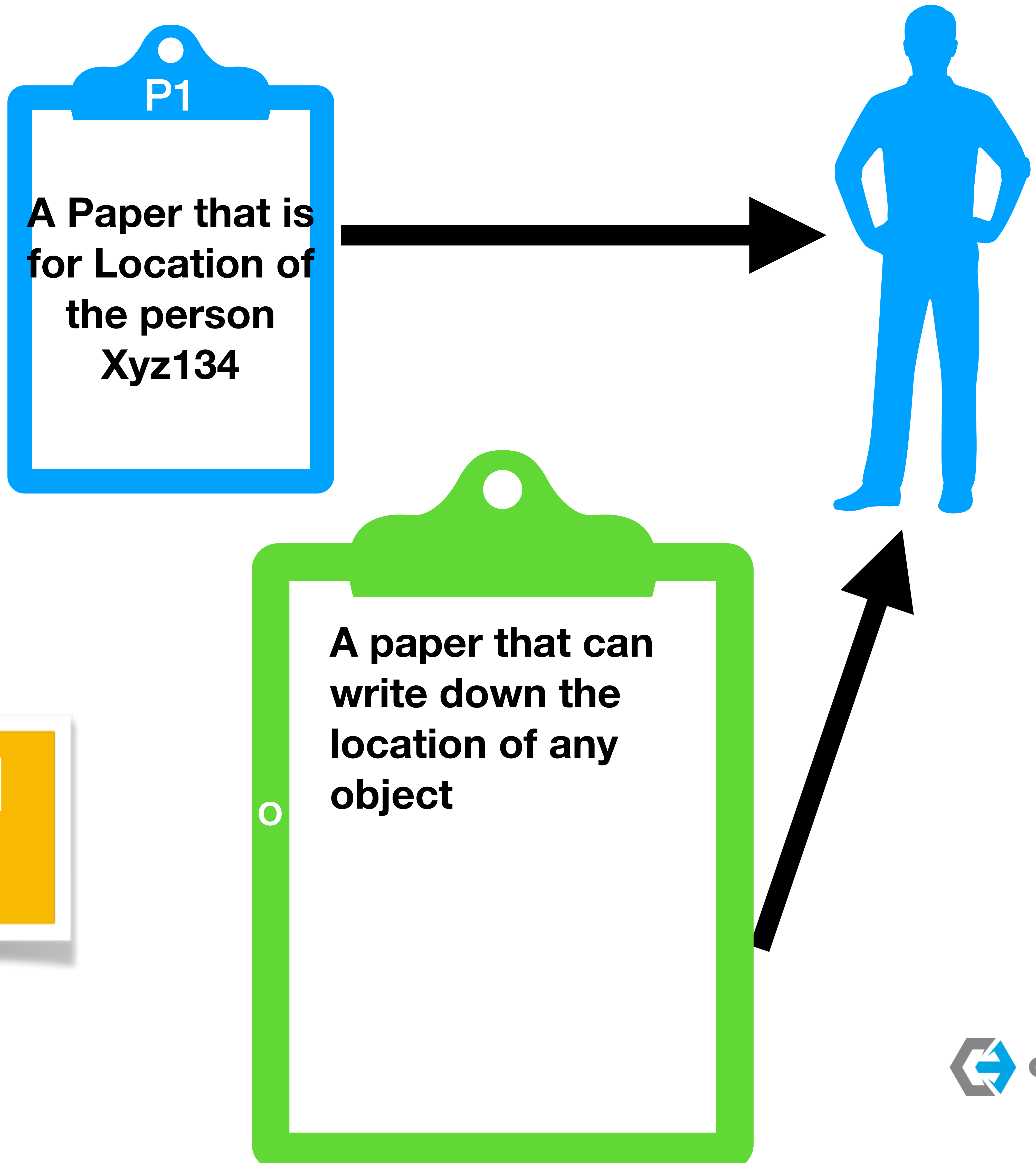

P1
A Paper that is for Location of the person Xyz134
o
A paper that can write down the location of any object

Reference Variable Downcasting

```
Object o = new Person() ;

Person p1 = (Person) o;
```

Parent Reference casted to Child reference has to be explicit



P1
A Paper that is for Location of the person Xyz134
A paper that can write down the location of any object
o

CYBERTEK

Casting Unrelated Types

```
Object o = new Person() ;

Cat p1 = (Cat) o;
```

instanceof Operator

use *instanceof* operator before downcasting to check if the object belongs to the specific type:

```
// Syntax
boolean b = ReferenceVariable instanceof Type ;
// Example
Object o = new Person() ;
boolean b1 = o instanceof Person ; // True
boolean b2 = o instanceof Object ; // True
boolean b1 = o instanceof Cat ; // FALSE
```

instanceof Operator

```
// Syntax
boolean b = ReferenceVariable instanceof Type ;
// Example
Object o = new Person() ;
if(o instanceof Person){
     Person p1 = (Person) o;
     p1.speak();
     // OR
     ( (Person) o ).speak();
    }
```

CYBERTEK

ClassCastException

If the real object doesn’t match the type we downcast to, then *ClassCastException* will be thrown at runtime

ClassCastException

```
Object o = new Object() ;

Person p1 = (Person) o;
```

At compile time :

It will check whether the objectType of reference variable and the casted Type have inheritance relationship or not.

SO THIS CODE WILL COMPILE FINE

ClassCastException

```
Object o = new Object() ;

Person p1 = (Person) o;
```

At compile time :

It will check whether the objectType of reference variable and the casted Type have inheritance relationship or not.

SO THIS CODE WILL COMPILE FINE

ClassCastException

```
Object o = new Object() ;

Person p1 = (Person) o;
```

At Run time :

It will check whether the Type of Object the reference variable pointing to IS-A the casted Type or not.

SO THIS CODE WILL THROW ClassCastException at runtime