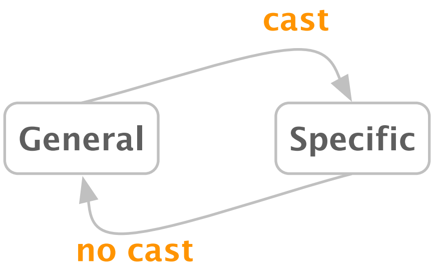
# Casting in Java

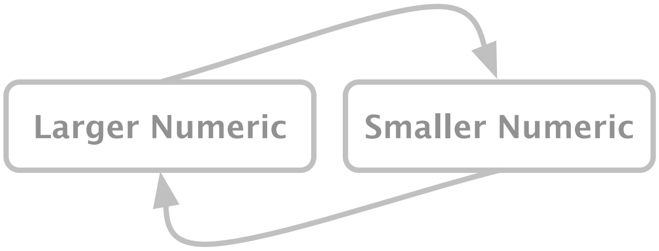
In a broad sense the Java compiler requires programmers to provide an *explicit cast* whenever assigning a "general" type to a more "specific" one. Three examples of this principle are shown in the diagrams below. Note that the principle applies to primitive and reference types alike.



When assigning a `double` to a `float`, information can be lost. (The range and precision of type `double` is greater than that of type `float`.) Therefore, in the absence of a cast, the compiler will signal an error when the programmer assigns a `double` to a `float`. Providing an explicit cast is the programmer's way of saying to the compiler "I'm aware that there could be a problem here. Relax."

On the other hand, assigning a `float` to a `double` causes no error, even without a cast.

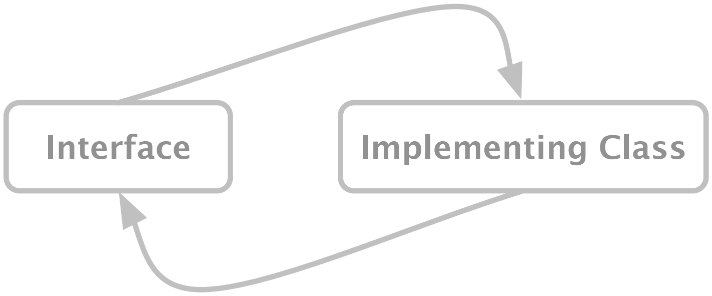
```
float f = (float) d;
```



```
double d = f;
```

**Figure 1**. Converting between `double` and `float`. Can you think of analogous examples using primitive types other than `double` or `float`?

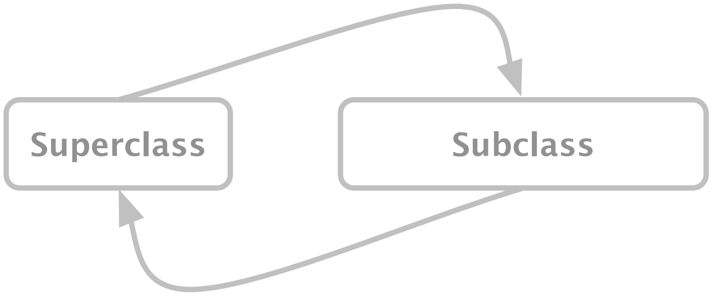
```
IPod pod = (IPod) p;
```



```
Playable p = pod;
```

**Figure 2**. Converting between interface and implementing class types.

```
IPod pod = (IPod) mp;
```



```
MP3Player mp = pod;
```

**Figure 3**. Converting between superclass and subclass types.

When assigning an implementing class reference to an interface reference, no cast is required. Likewise when assigning a subclass reference to a superclass reference. (As shown to the left, in both of the reverse cases the compiler does require a cast.)

Some casts can be checked at compile time:

```
String s = "Bucky";
IPod pod = (IPod) s; //compiler error
//(since IPod is neither a subclass
//nor superclass of String)
```

Other casts are checked during runtime. See below for an example.

## ClassCastException

If an invalid cast is made at runtime a `ClassCastException` is thrown.

```
MP3Player mp3 = new IRiver();
//next line causes an exception since
//IPod is not a subclass of IRiver
IPod pod = (IPod) mp3; //exception
//Why didn't this cause a
//compile-time error?
```

## instanceof operator

`instanceof` is a binary operator which can be used to avert a `ClassCastException`.

```
if(mp3 instanceof IPod){
    //this cast is always safe
    IPod pod = (IPod) mp3;
```

Handout for CS 302 by Aneesh Karve, karve@cs.wisc.edu