

Pointing to Nothing

- What's the default value for a pointer variable?
- Need a distinguished address meaning "not pointing to anything"

7/21/98 CSE 143 Summer 1998 214

The NULL Pointer

- ◆ NULL (defined in stdlib.h) can be used to point to nothing
 - ◆ Internally, NULL is just the memory location o
 - The compiler guarantees that nothing will get stored there

215

 \blacklozenge NULL is compatible with pointers of every type

7/21/98 CSE 143 Summer 1998

Using the NULL Pointer

As a default pointer value:

```
Fraction *pf = NULL;
...
if( pf == NULL ) {
    pf = new Fraction( 22, 7 );
}
```

◆ As a terminator (soon)

```
for( Node *n = front; n != NULL; n = n->next )
{
    // Do something to data at n
}
```

7/21/98 CSE 143 Summer 1998 216

The Receiver of a Method

```
bool Fraction::isEven() {
    if( NaN() ) {
        return false;
    }
    return (den == 1) && ((num % 2) == 0);
}
```

- ◆ Where do num, den and NaN come from?
 - They are associated with the (implicit) receiver of the method
- But how does this work?
 - How does the method get access to a particular receiver?

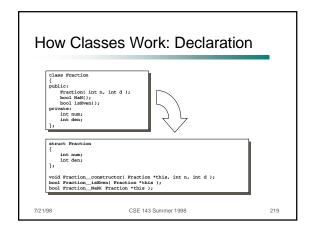
7/21/98 CSE 143 Summer 1998 21

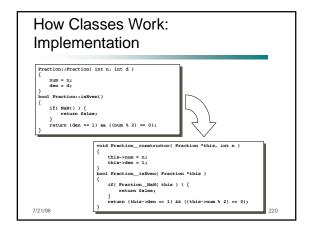
The this Pointer

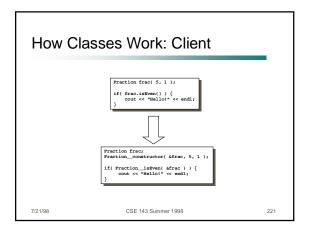
- Inside methods, there's a "hidden" extra parameter: this
 - this always points to the receiver of the method
 - ♦ For a method of class c, this has type pointer-to-c
- All implicit references to the receiver are actually referenced via the this pointer:

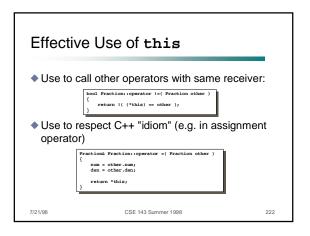
```
bool Fraction::isEven()
{
    if( this->NaM() ) {
        return false;
    }
    return (this->den == 1) && ((this->num % 2) == 0);
}
```

1/98 CSE 143 Summer 1998 218









Summary Null can be used as the value of a pointer that points to nothing this can be used as a way to point to the receiver of a method Classes are implemented by passing this as a hidden extra parameter in methods