

**Last Day**

- objects
  - constructors
  - static

**Today**

- more objects
  - comparing
  - assignment
  - strings
  - public, private

**Objects**

**Important words from last day...**

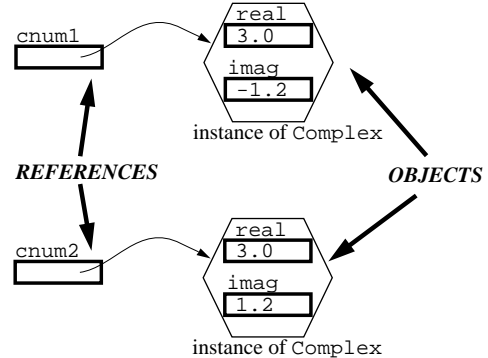
- **instance variable** — a variable *inside a class* that isn't static
- **class variable** — a static variable *inside a class*
- **instance method** — a method that isn't static
- **class method** — a static method
- **class methods** cannot access *instance variables*
  - unless they have a reference to an object
  - why?
    - because the instance variables *don't exist* unless an object is created
    - they exist only for that object (objects do not share instance variables)

**Comparing Objects (references)**

```

• Not what you think!!!
class ComplexTest
{
    public static void main(String[] args)
    {
        Complex cnum1=new Complex(3.0,-1.2);
        Complex cnum2=new Complex(3.0,-1.2);
        if( cnum1 == cnum2 ) {
            // compares references,
            // never gets here!!!
        }
    }
}
    
```

- After running:



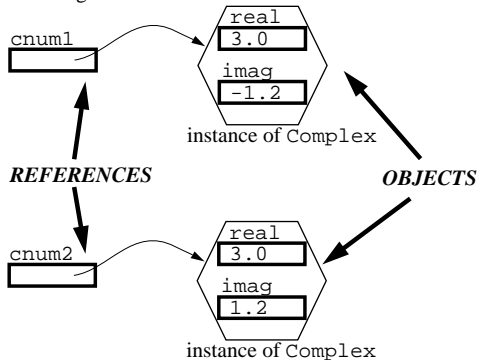
- cnum1 and cnum2 point to *different objects*

**Comparing Objects (contents, Text p.340)**

```

• This is probably what you want
class ComplexTest
{
    public static void main(String[] args)
    {
        Complex cnum1=new Complex(3.0,-1.2);
        Complex cnum2=new Complex(3.0,-1.2);
        if( cnum1.equals(cnum2) ) {
            // you must write equals() method
            // so this will work properly!!!
        }
    }
}
    
```

- After running:



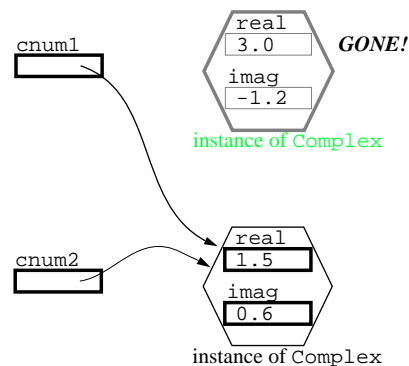
- cnum1 and cnum2 point to *different objects*, same contents

**Assignment of Objects (references)**

```

• Not what you think!!!
class ComplexTest
{
    public static void main(String[] args)
    {
        Complex cnum1, cnum2;
        cnum1 = new Complex(3.0,-1.2);
        cnum2 = new Complex(1.5, 0.6);
        cnum1 = cnum2;
    }
}
    
```

- After running:



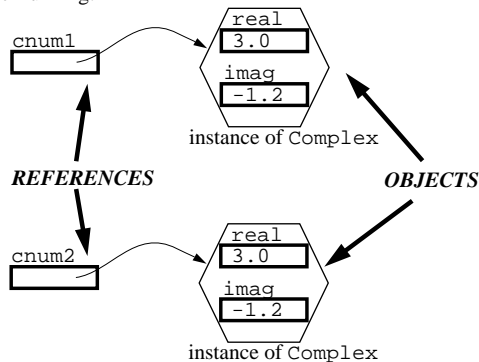
- cnum1 and cnum2 point to the *same object*
- previous cnum1 object is gone

## Assignment of Objects (copying, Text 3.8)

- Two ways to do it properly (we won't use clone() method):
 

```
class ComplexTest
{
    public static void main(String[] args)
    {
        Complex cnum1, cnum2, cnum3;
        cnum1 = new Complex(3.0, -1.2);
        //cnum2 = (Complex)cnum1.clone();
        cnum2 = new Complex(cnum1);
    }
}
```

- After running:



- cnum2 is a **new object**, contents are same as contents of the other object cnum1

## Hiding Information

can declare *methods* or *variables* in the class

- public
- private
  - these are sometimes called **visibility modifiers**

public

- any method (even in other class) can access public variables
- any method (even in other class) can invoke public methods

private

- restricts access of a variable or method to *this class only*
- only** methods inside the same class can use

```
class Complex
{
    private double real, imag;
    ...
}
```

- real can **only** be accessed by methods inside Complex

- real **cannot** be accessed by ComplexTest

```
class ComplexTest
{
    public static void main(String[] args)
    {
        Complex cnum = new Complex();
        cnum.real = 3; // error invisible!
    }
}
```

- we can provide public **mutator** and **accessor methods** for
  - setting or getting some value, respectively
  - we are **HIDING** the **INSTANCE VARIABLES**
  - all access to instance variables is controlled by methods

## Returning Objects

- until now, can only return one value
  - factorial(int n) returns int
  - power(double x, int n) returns double
  - isPrime(int n) returns boolean
- return **more than one** value?

- example... **incorrect way...doesn't work**

```
public (double,double)
complexAdd( Complex x )
{
    double sumReal = real + x.real;
    double sumImag = imag + x.imag;
    return (sumReal,sumImag);
}
```

...using it...

```
(valReal,valImag) = complexAdd(...);
```

- correct way**

```
public Complex addNew( Complex x )
{
    return new Complex( real+x.real,
                        imag+x.imag );
}
```

...using it...

```
Complex cnum1 = new Complex(1,2);
Complex cnum2 = new Complex(1,3);
Complex cnum3 = cnum1.addNew(cnum2);
```

```
class Complex
{
    private double real, imag;

    public void setReal( double r )
    { real = r; }

    public void setImag( double i )
    { imag = i; }

    public double getReal()
    { return real; }
}

class ComplexTest
{
    public static void main(String[] args)
    {
        Complex cnum = new Complex();
        cnum.setReal(3);
        cnum.setImag(2);
        System.out.println(cnum.getReal());
    }
}
```

what if **neither** private nor public is given?

- "package" access
- in between private and public
- visible to classes belonging to the same package
- programs say which package they belong to by placing at the top of their program:
 

```
package MyPackageName;
```