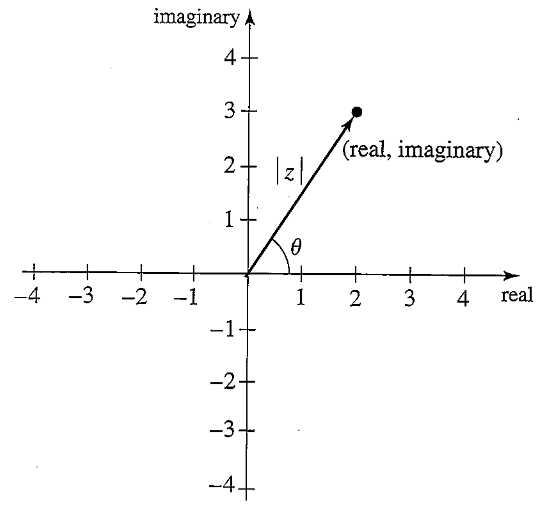

Figure 2.2 A complex number z can be defined by its real and imaginary parts, *real* and *imag*, respectively, or by its magnitude $|z|$ and phase angle θ.

```
      a.conjugate();              // complex conjugate of a
      System.out.println(a);
   }
}
```

Because the methods of class `Complex` are not static, we must first instantiate a `Complex` object with a statement such as

```
Complex a = new Complex(3.0, 2.0);
```

The variable `a` is an object of class `Complex`. As before, we can think of `new` as creating the instance variables and memory of the object. Compare the form of this statement to the declaration

```
double x = 3.0;
```

A variable of class type `Complex` is literally more complex than a primitive variable because its definition also involves associated methods and instance variables.

Note that we have first written a class that uses the `Complex` class before we have actually written the latter. Although programming is an iterative process, it is usually a good idea to first think about how the objects of a class are to be used. Exercise 2.21 encourages you to do so.

Exercise 2.21 Complex number test

What will be the output when `ComplexApp` is run? Make reasonable assumptions about how the methods of the `Complex` class will perform using your knowledge of Java and complex numbers. ■

We need to define methods that add, multiply, and take the conjugate of complex numbers and define a method that prints their values. We next list the code for the `Complex` class.

Listing 2.15 Listing of the `Complex` class.

```
package org.opensourcephysics.sip.ch02;
public class Complex {
   private double real = 0;
   private double imag = 0;

   public Complex() {
      this(0, 0); // invokes second constructor with 0 + i0
   }

   public Complex(double real, double imag) {
      this.real = real;
      this.imag = imag;
   }

   public void conjugate() {
      imag = -imag;
   }

   public Complex add(Complex c) {
      // result also is complex so need to introduce another variable
      // of type Complex
      Complex sum = new Complex();
      sum.real = real+c.real;
      sum.imag = imag+c.imag;
      return sum;
   }

   public Complex multiply(Complex c) {
      Complex product = new Complex();
      product.real = (real*c.real)-(imag*c.imag);
      product.imag = (real*c.imag)+(imag*c.real);
      return product;
   }

   public String toString() {
      // note example of method overriding
      if(imag>=0) {
         return real+" + i"+Math.abs(imag);
      } else {
         return real+" - i"+Math.abs(imag);
      }
   }
}
```

The `Complex` class defines two constructors that are distinguished by their parameter list. The constructor with two arguments allows us to initialize the values of the instance variables. Notice how the class *encapsulates* (hides) both the data and the methods that characterize a complex number. That is, we can use the `Complex` class without any knowledge of how its methods are implemented or how its data is stored.

The general features of this class definition are as before. The variables `real` and `imag` are the instance variables of class `Complex`. In contrast, the variable `sum` in method `add` is a *local* variable because it can be accessed only within the method in which it is defined.