Lab 1: Complex numbers.

A complex number is an expression of the form $a + bi$ where $a, b \in \mathbb{R}$ and $i^2 = -1$.

Add $z = a + bi, w = c + di$

$z + w = (a + c) + (b + d)i$

\[ \begin{array}{c}
\text{b} \\
\text{a}
\end{array} \quad a + bi \]
\[ z - w = (a + bi) - (c + di) \]
\[ = (a - c) + (b - d)i \]

\[ z \cdot w = (a + bi)(c + di) \]
\[ = ac + bc i + ad i + bd i^2 \]
\[ = (ac - bd) + (bc + ad)i \]

\[ \overline{z} = a - bi \]
Note:
\[ z \bar{z} = (a+bi)(a-bi) = a^2 + b^2 \]

Modulus or Absolute Value or Length:
\[ |z| = |a+bi| = \sqrt{a^2 + b^2} = (z \bar{z})^{\frac{1}{2}} \]

Argument:
\[ \arg(z) = \Theta \]
\[ \arg(\bar{z}) = -\Theta = -\arg(z) \]

Real Part:
\[ \text{Re}(z) = a \]

Imaginary Part:
\[ \text{Im}(z) = b \]
\[ |z \cdot w| = |z| \cdot |w| \]

i.e. \( \sqrt{(ac - bd)^2 + (bc + ad)^2} = \sqrt{a^2 + b^2} \cdot \sqrt{c^2 + d^2} \)

Reciprocal: say \( z = a + b i \)

\[
\frac{1}{z} = \frac{\overline{z}}{z \overline{z}} = \frac{a - b i}{a^2 + b^2} = \left( \frac{a}{a^2 + b^2} \right) + \left( \frac{-b}{a^2 + b^2} \right)i
\]

Check: \( \frac{1}{z} \cdot z = 1 \)

Division: \( z = a + b i, \ w = c + d i \)

\[
\frac{w}{z} = \frac{w \overline{z}}{z \overline{z}} = \frac{(c + di)(a - bi)}{a^2 + b^2}
\]
\[
\frac{(ac+bd) + (ad-bc)i}{a^2 + b^2} = \left(\frac{ac+bd}{a^2 + b^2}\right) + \left(\frac{ad-bc}{a^2 + b^2}\right)i
\]

\textbf{Result:} \quad a(b,c) = (ab)c
chapter 7: Inheritance

Recall: all classes are subclasses of Object

```
Object
  ├── Person
  │    └── PersonRational
  │        ├── String
  │        └── Math
  └── Scanner
```

To create a subclass of an existing class, use `extends`.
Ex.

// Foo.java
class Foo {

// Bar.java
class Bar extends Foo {

Object o = new Foo();
Foo f = (Foo) o;
Bar b = (Bar) o;

in some method, in some other class
Foo f;
Bar b;

To allocate memory:

```java
f = new Foo();
b = new Bar();
```

Any member variable or method of `f` also belongs to `b`, but not conversely.

"A Bar is a Foo"
Subtype Principle

A subclass object can always be used where an object of its superclass is expected.

In particular,

we can assign a variable of superclass type to an object of the subclass type. (reverse is false.)

So reference variables can point down lines of ancestry, but not up.
Ex. FooSubclassBar

`f`  

`b`  

\[ f = b \]  // subtype principle.

However

\[ b = f \]  // syntax error.

Ex. Student

StudentText
why use a subclass?

+ code re-use
  Student uses existing tested code from Person.

+ class hierarchy reflects actual relationship in problem domain, i.e. a student "really is" a person.

Ex. Employee

Dynamic Method Dispatch: each runtime object "knows" what it is and how it should be acted upon.