OOP (Object-Oriented Programming)

- Modularization: determing the classes you need to solve a problem
- Encapsulation: making sure all related components (variables, methods) are in a single class so that the class is self-contained
- Abstraction: once a class is built, we can treat it as a new data type and use objects of that class w/o having to know how it works inside
- Accessibility (public / private)
  - Data Hiding
    - Security reasons
    - Doesn't need to know
    - Shouldn't access it
  - Ideal: make everything private unless access is absolutely needed
- Polymorphism: "many forms"
  - ↳ allows for using the same term to mean many things, unambiguously
  - Method Overloading:
    - ↳ same name, different param. lists


```
public void Draw(Rectangle r)
{
  //Draw the rect.
}
public void Draw(Circle c)
{
  //_____ circle
}
public int Add(int a, int b)
{
}
public double Add(double a, double b)
{
}
// see code
```

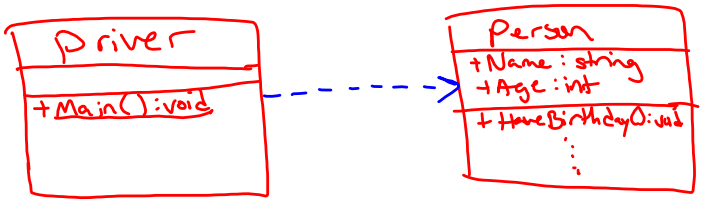
Length
- ↳ Rectangle (length of side)
  Sentence (# of words)
  Lecture (# mins/hrs)

- Inheritance (discussed later)

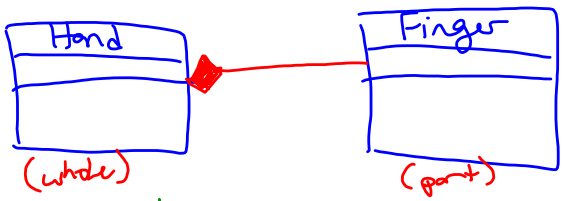
- Classes + Objects
  - Class: blueprint / pattern / design
    describes the attributes + functionality of every instance of the class
  - Object: one unique instance created from blueprint (class)
    - ↳ many obj. from the same class
    - ↳ each obj. has the same attr. + func.
    - ↳ values of the attr. allow objs to differ
  - Message passing: asking an obj. of a class to perform some action (calling a method)
  - State of an object: the set of values for all attr. of an obj. at a particular point in time
  - Components of object:
    attributes / properties
    methods
    constructors

UML: Unified Modeling Language

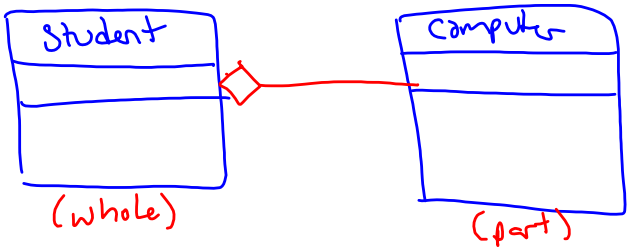
- tool to make diagrams to represent our code
- Relationships:
  - Dependency / Instantiation
    - ↳ a method of an object may instantiate and use an object of another class to do a specific task but the object being used is NOT a fundamental part of the object using it
      - ↳ makes a obj. of a class it uses it but not an attribute



  - Has-A / Part-Whole Relationships
    - ↳ obj. of one class has an attribute of an object of another class
      - Composition
        - ↳ relationship is essential
        - ↳ constructor instantiates an object of the other class
        - ↳ the 2 come into existence together



      - Aggregation
        - ↳ relationship is NOT essential
        - ↳ not instantiated in the constructor
          - ↳ methods, may or may be critical



Enumerated Data Types

→ enums

- data type w/ small # of specific values