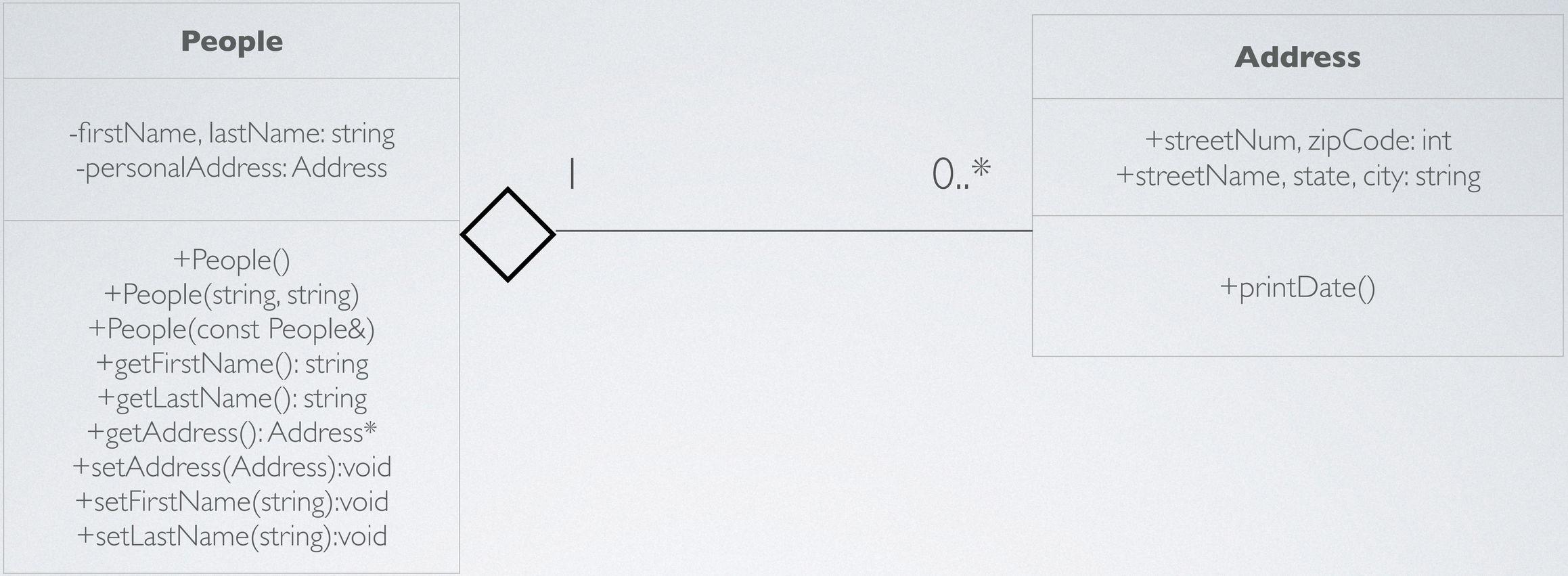
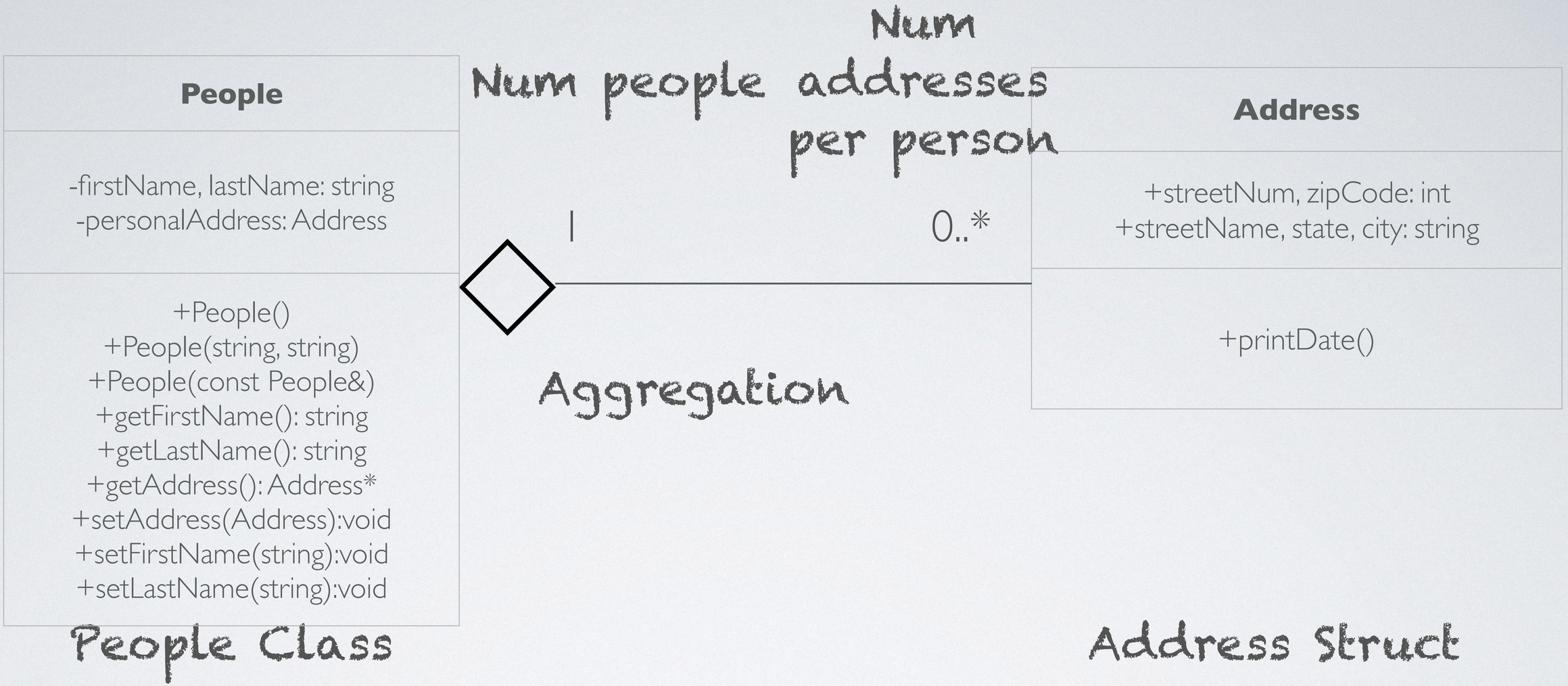
CLASSES AND INHERITANCE

CS202: Computer Science II
Sara Davis
UNR Fall 2022

TOPICS

- Class writing practice
- Inheritance


People
-firstName, lastName: string
-personalAddress: Address
+People()
+People(string, string)
+People(const People&)
+getFirstName(): string
+getLastName(): string
+getAddress(): Address*
+setAddress(Address):void
+setFirstName(string):void
+setLastName(string):void
1
0..*
Address
+streetNum, zipCode: int
+streetName, state, city: string
+printDate()


Num people addresses per person
People
-firstName, lastName: string
-personalAddress: Address
+People()
+People(string, string)
+People(const People&)
+getFirstName(): string
+getLastName(): string
+getAddress(): Address*
+setAddress(Address):void
+setFirstName(string):void
+setLastName(string):void
1
0..*
Aggregation
Num
Address
+streetNum, zipCode: int
+streetName, state, city: string
+printDate()
People Class
Address Struct

CLASS WRITING PRACTICE

- Write a Person class (person.cpp, person.h)
  - First Name
  - Last Name
  - Setters & Getters
  - Default, parameterized, copy constructors

CLASS WRITING PRACTICE

- Create an address struct (address.cpp)
  - Street number
  - Street
  - City
  - State
  - Zip

CLASS WRITING PRACTICE

- Write a driver file that
  - Sets first and last name to yours
  - Sets address to UNRs
  - Prints first and last name
- Write a makefile that combines the address.cpp, person.cpp, and person.h into the executable person

INHERITANCE

- Class relationship hierarchy.
  - Child inherits from parent class
    - Example: Cat class (child) Pet class (parent)
      - All pets have some number of legs, might have a tail, and have a species. -**What they have in common**
      - A cat is a pet -**IS a**
      - Cats usually have a breed associated with them (Tabby) while most other pets do not -**What is different**

INHERITANCE IS THE SECOND OOP PRINCIPLE

- Organizes classes into a hierarchy (abstraction!)
- Allows classes to inherit attributes and behavior through hierarchy
- Reuse common logic, extract unique logic for specific class
  - Less prone to error
  - **Extends** a generic, existing class with specific attributes and functionality

INHERITANCE PRACTICE

- Let's extend that person class we just wrote!
  - A customer is a type of person. Therefore, a Customer class can inherit from the Person class (so it has name, date, address) and add a username and password property only accessible through customer (not through person)

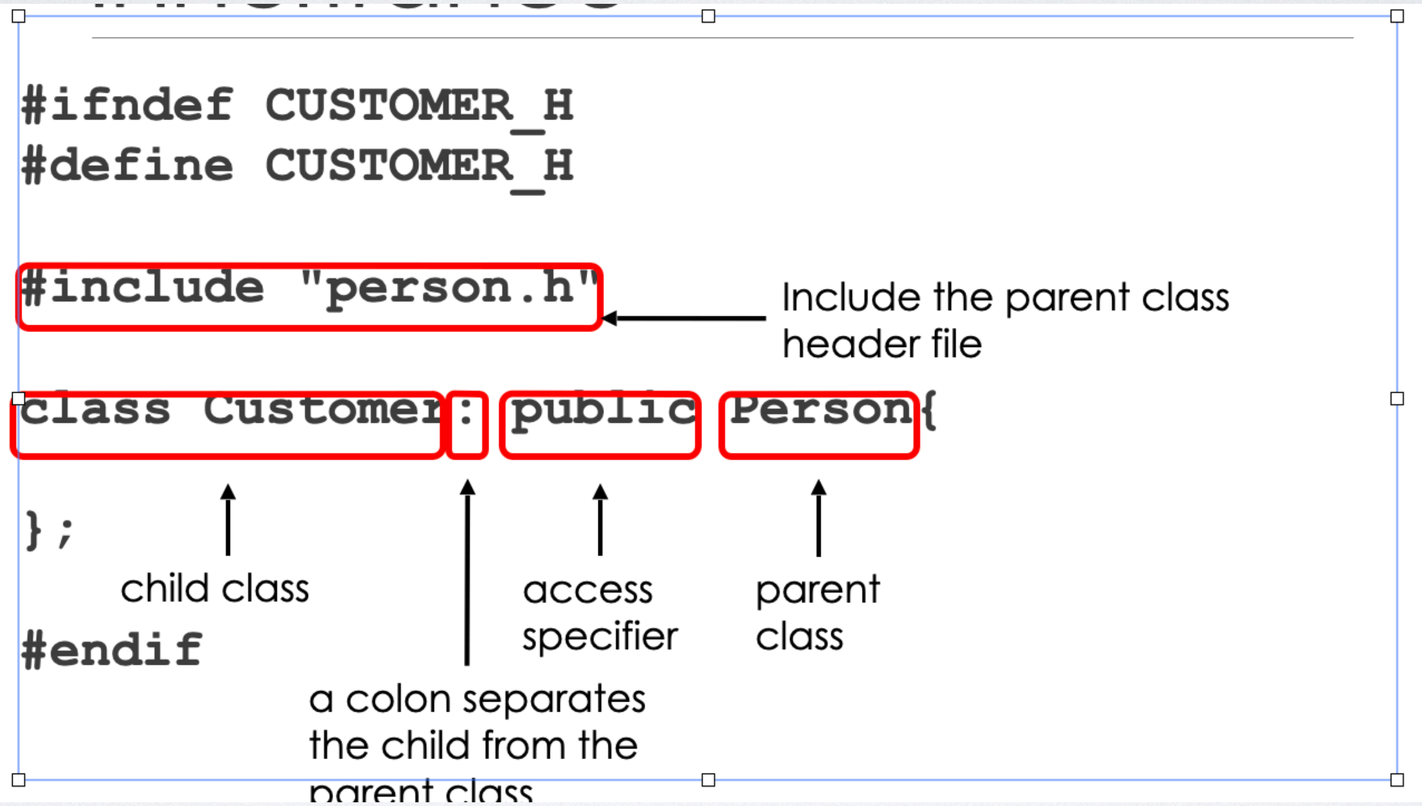
ACCESS

- Child classes (customer) can only access Parent (person) class public and protected methods and properties.
- Same Class: can use public, protected, private
- Derived Class (inherited): can use public, protected
- Outside class: public only

ACCESS

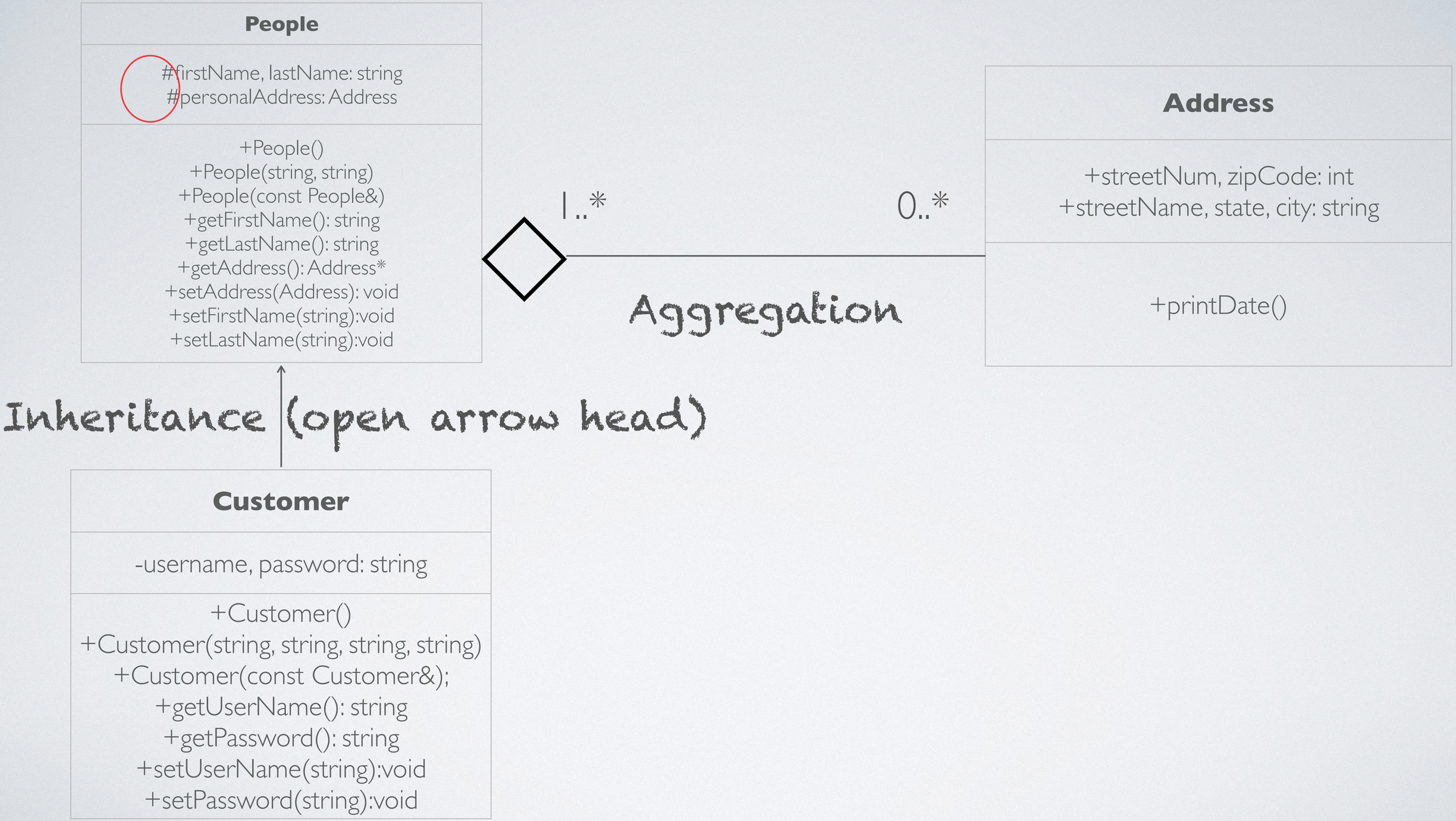
- Our original person class uses **private** properties
  - For customer to inherit, they must be changed to **protected**
  - Child can access any public member
  - If child (customer) needs access to parent (person) private data, then the parent's getters and setters must be used.

INHERITANCE



INHERITANCE

- Inherit parent class public/protected methods and properties
- Must use parent getters/setters BUT create separate constructors for child
    - If not implemented, copy constructor and assignment operator is created by invoking base class


People
#firstName, lastName: string
#personalAddress: Address
+People()
+People(string, string)
+People(const People&)
+getFirstName(): string
+getLastName(): string
+getAddress(): Address*
+setAddress(Address): void
+setFirstName(string):void
+setLastName(string):void
1..*
0..*
Aggregation
Address
+streetNum, zipCode: int
+streetName, state, city: string
+printDate()
Inheritance (open arrow head)
Customer
-username, password: string
+Customer()
+Customer(string, string, string, string)
+Customer(const Customer&);
+getUserName(): string
+getPassword(): string
+setUserName(string):void
+setPassword(string):void

INHERITANCE DEMO (CUSTOMER)

INHERITANCE DEMO: HOLIDAY