

Classes

CS 401
(an excerpt)

Object-Oriented Programming

- Object-oriented programming
 - Real-world objects represented inside “digital” objects
 - Methods interact with and modify objects
- Classes
 - Blueprints for data
 - Describe how to interact with data
- Encapsulation
 - All data and operations for an *object* are together
 - Everything related to that *object* is encapsulated inside the class
- Data Abstraction
 - Hide details of how data is stored/represented

Example Class: Person

- What data should be stored in the Person class?

Example Class: Person

- What data should be stored in the Person class?
 - Name
 - Age
 - Address

Example Class: Person

- What data should be stored in the Person class?
 - Name
 - Age
 - Address
- What actions should the Person class allow?

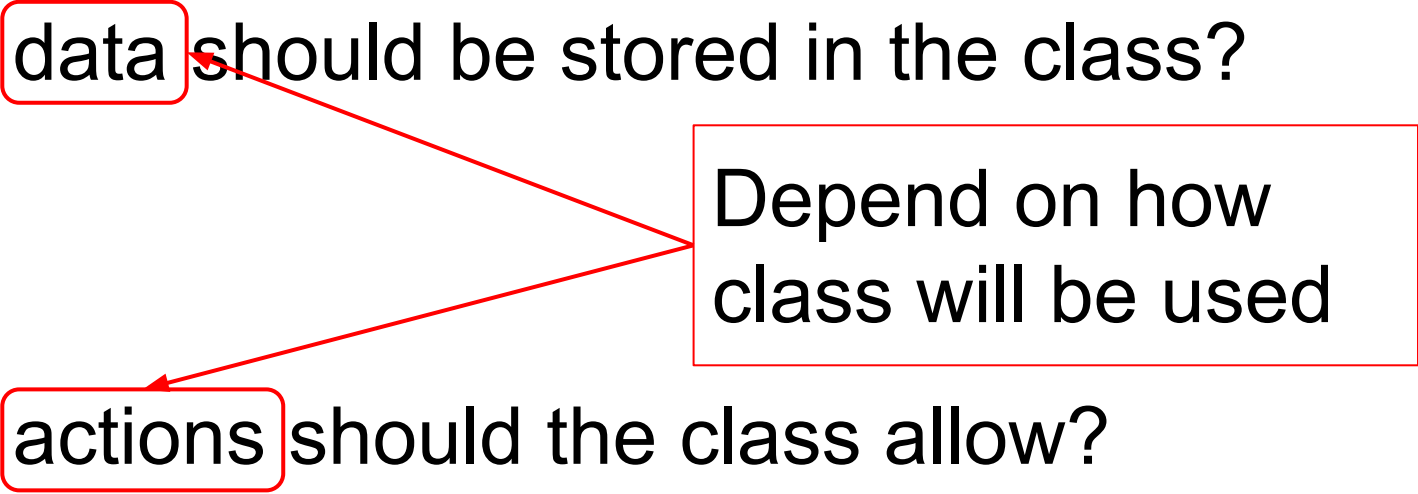
Example Class: Person

- What data should be stored in the Person class?
 - Name
 - Age
 - Address
- What actions should the Person class allow?
 - Get the name, set the name
 - Get the age, increment the age
 - Get the address, update the address

Designing a Class

- What **data** should be stored in the class?

Depend on how
class will be used



- What **actions** should the class allow?

Classes and Objects

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class

Classes and Objects

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class

Person Class

Name:	<input type="text"/>
Age:	<input type="text"/>
Address:	<input type="text"/>

Classes and Objects

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class
- **Objects: Instances of a class**
 - Holds values for the data in the blueprint

Person Class

Name:	<input type="text"/>
Age:	<input type="text"/>
Address:	<input type="text"/>

Person

Name:	<input type="text" value="Zeus"/>
Age:	<input type="text" value="4000"/>
Address:	<input type="text" value="Mt. Olympus"/>

Classes and Objects

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class
- **Objects: Instances of a class**
 - Holds values for the data in the blueprint
- **Reference Variables: Refer to objects**

Person Class

Name:	<input type="text"/>
Age:	<input type="text"/>
Address:	<input type="text"/>

Person

Name:	<input type="text" value="Zeus"/>
Age:	<input type="text" value="4000"/>
Address:	<input type="text" value="Mt. Olympus"/>

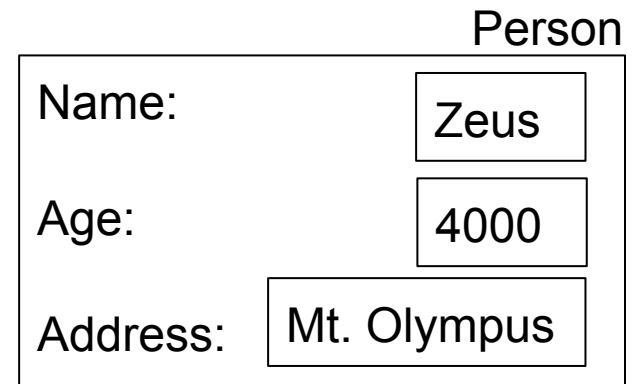
Classes and Objects

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class
- **Objects: Instances of a class**
 - Holds values for the data in the blueprint
- **Reference Variables: Refer to objects**
`p = Person("Zeus", 4000, "Mt. Olympus")`

Classes and Objects

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class
- **Objects: Instances of a class**
 - Holds values for the data in the blueprint
- **Reference Variables: Refer to objects**

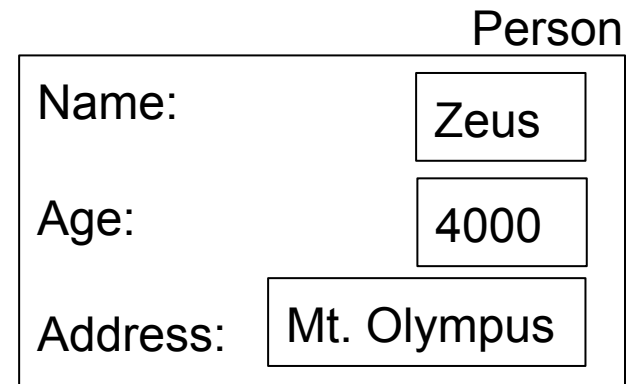
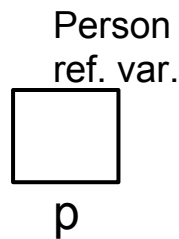
```
p = Person("Zeus", 4000, "Mt. Olympus")
```



Classes and Objects

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class
- **Objects: Instances of a class**
 - Holds values for the data in the blueprint
- **Reference Variables: Refer to objects**

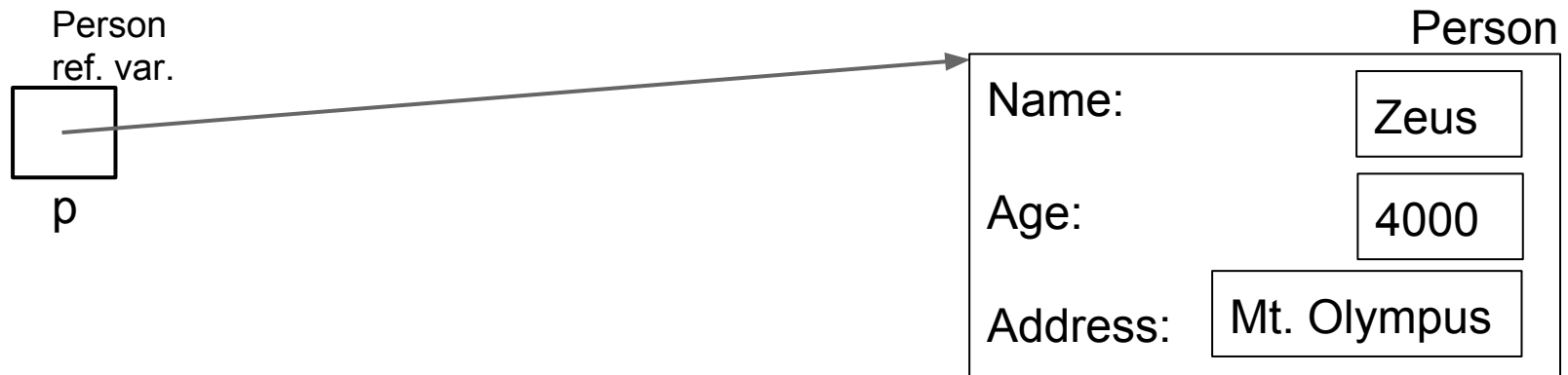
```
p = Person("Zeus", 4000, "Mt. Olympus")
```



Classes and Objects

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class
- **Objects: Instances of a class**
 - Holds values for the data in the blueprint
- **Reference Variables: Refer to objects**

```
p = Person("Zeus", 4000, "Mt. Olympus")
```

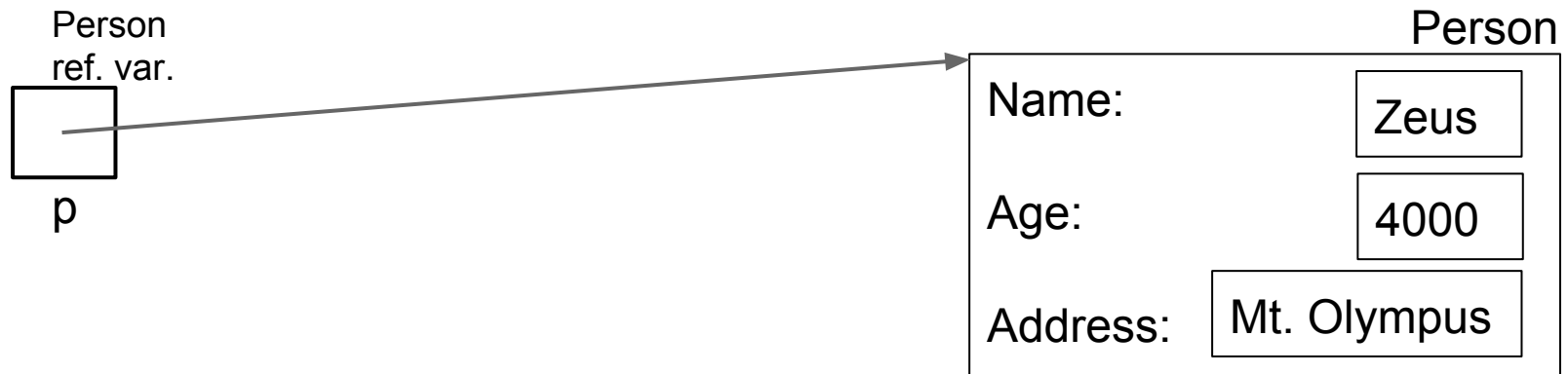


Classes and Objects

We'll need to create the Person constructor ... more on this later.

- **Classes: Blueprints for data**
 - Define/declare a class
 - List data and actions for the class
- **Objects: Instances of a class**
 - Holds values for the data in the blueprint
- **Reference Variables: Refer to objects**

```
p = Person("Zeus", 4000, "Mt. Olympus")
```



Object Independence

- Objects are independent of each other

```
zeus = Person("Zeus", 4000, "Mt. Olympus")  
hades = Person("Hades", 3900, "Underworld")
```

