Lesson 12: OOP #2, Accessor Methods (W03D4)

Balboa High School

Michael Ferraro

September 3, 2015

Do Now

- In your driver class from last class, create another new `Person` object with these characteristics:
  - `age = 22`
  - `weight = 170.2`
  - `location = "Buffalo, NY"`
  - `fullName = "Jim Kelly"`
  - `greeting = "Brrr...it's cold here."`
- Print to the console at least two of the instance's fields (similar to the printed output for `ralph` last class).
- Check your work against your neighbors' and make corrections as necessary.

Aim

Students will delve deeper into OOP, learning about two (of four) kinds of *methods*.

Recap of Objects Thus Far

What do you recall from the last lesson regarding objects?

Recap of Objects Thus Far

- Objects hold information
- Classes define what objects will contain, like a blueprint
- **Today:** Objects can *do* things!

Methods

- A *method* is a procedure that is part of a class
- *Methods* are the "verbs" of a class; they do things!
- There are general methods and a few special types

Special Method Type: Constructor

- *constructor* methods make objects.

Special Method Type: Constructor

- *constructor* methods make objects.
- **The name of constructor ALWAYS matches name of the class.** E.g., in the `Person` class, the constructor method is called `Person()`.

Special Method Type: Constructor

- *constructor* methods make objects.
- **The name of constructor ALWAYS matches name of the class.** E.g., in the `Person` class, the constructor method is called `Person()`.
- Our `Person()` constructor is dull — it doesn't *do* anything:

```
public class Person() {
        public Person() {
        }
}
```

Special Method Type: Constructor

- *constructor* methods make objects.
- **The name of constructor ALWAYS matches name of the class.** E.g., in the `Person` class, the constructor method is called `Person()`.
- Our `Person()` constructor is dull — it doesn't *do* anything:

```
public class Person() {

        public Person() {

        }

}
```

- In fact, we could have left `Person()` out entirely since it does nothing; **Java automatically provides a default, no-args constructor when you don't write one**

An Example of a General Method

Recall the `greeting` string in the `Person` class:

```
public class Person {

    int age;
    int weight;
    String location;
    String fullName;
    String greeting;

    public Person() {
    }
}
```

An Example of a General Method

Omitting the variables we're not currently using:

```
public class Person {

    //other vars omitted...
    String greeting;

    public Person() {
    }
}
```

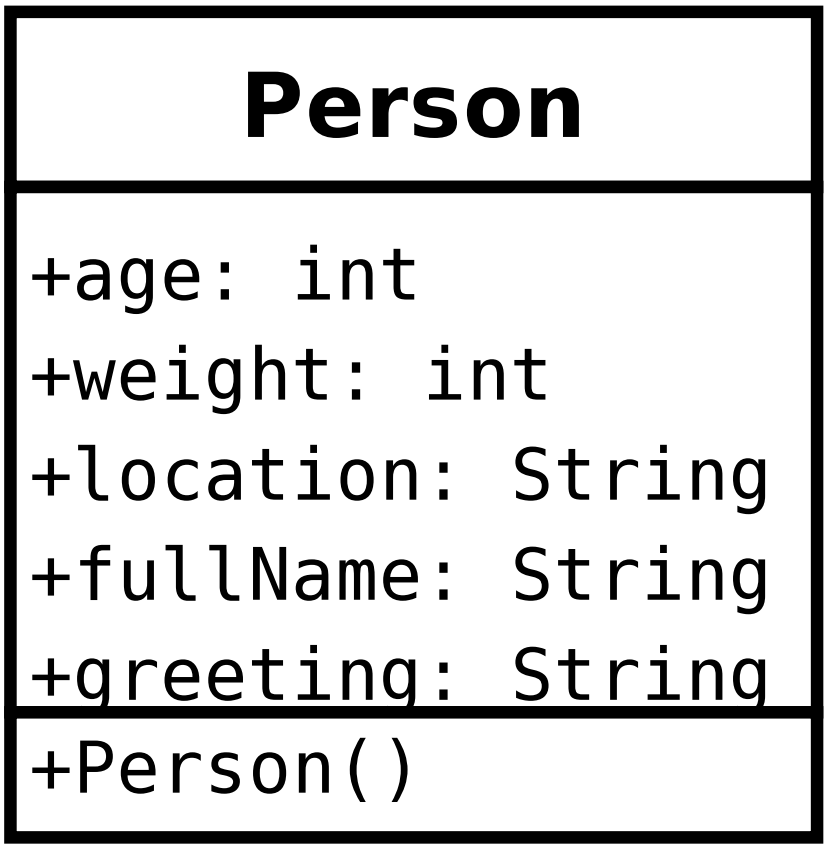
An Example of a General Method

Adding a method called `greet()`:

```
public class Person {

    //other vars omitted...
    String greeting;

    public Person() {
    }

    public void greet() {
        System.out.println(greeting);
    }
}
```

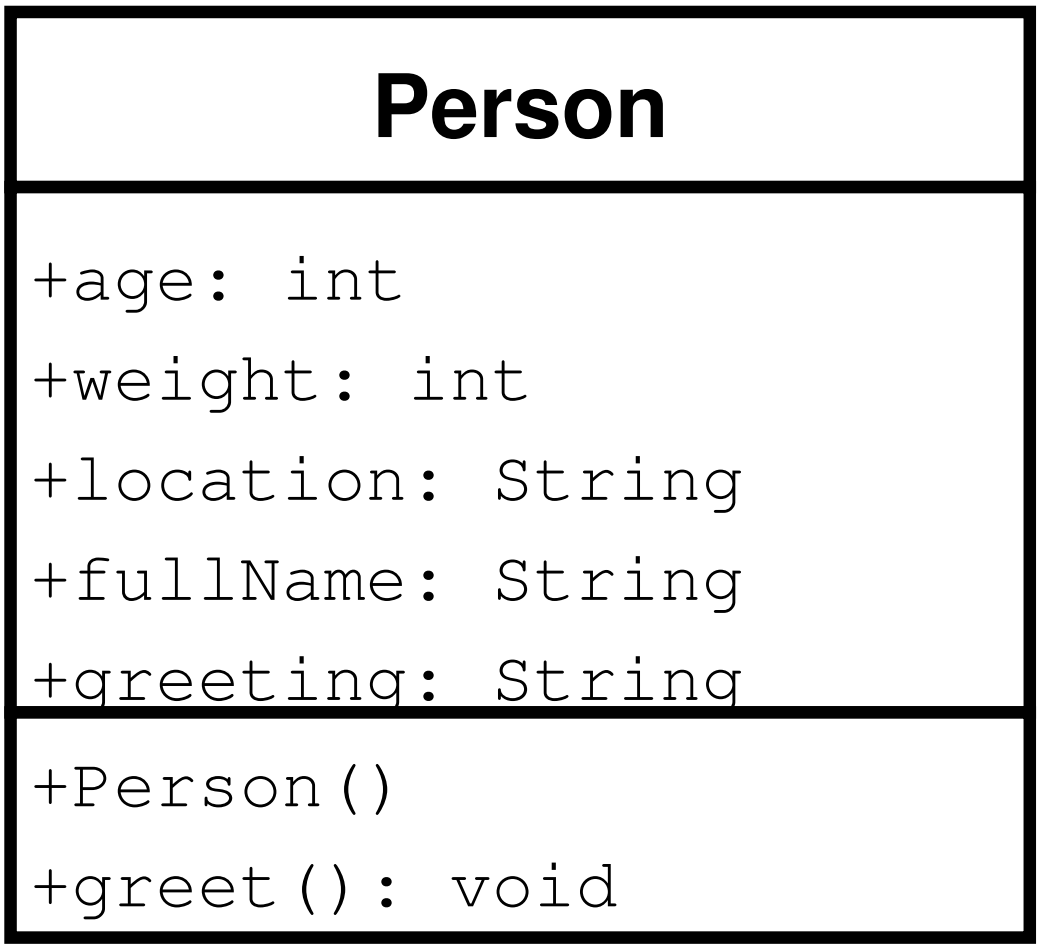
Update the UML Class Diagram

Last class, we had only one method, the constructor `Person()`.



Update the UML Class Diagram

Last class, we had only one method, the constructor `Person()`. Now we add `greet()`.



Update Driver Class to Use `Greet()`

```
public class PersonDriver {

    public static void main(String[] args) {

        Person ralph = new Person();

        // remember, we set age, location,
        // etc. last time...

        ralph.greeting = "Heloooo there!";

        ralph.greet(); // using dot notation
                       // to call method
    }
}
```

Functions: Input and Output

- What we call **methods** in Java are similar to what would be called **functions** or **procedures** in other languages.

Functions: Input and Output

- What we call **methods** in Java are similar to what would be called **functions** or **procedures** in other languages.
- Functions take in some value(s)

Functions: Input and Output

- What we call **methods** in Java are similar to what would be called **functions** or **procedures** in other languages.
- Functions take in some value(s)

  then they "think"...

Functions: Input and Output

- What we call **methods** in Java are similar to what would be called **functions** or **procedures** in other languages.
- Functions take in some value(s)

  then they "think"...

  and finally return something.

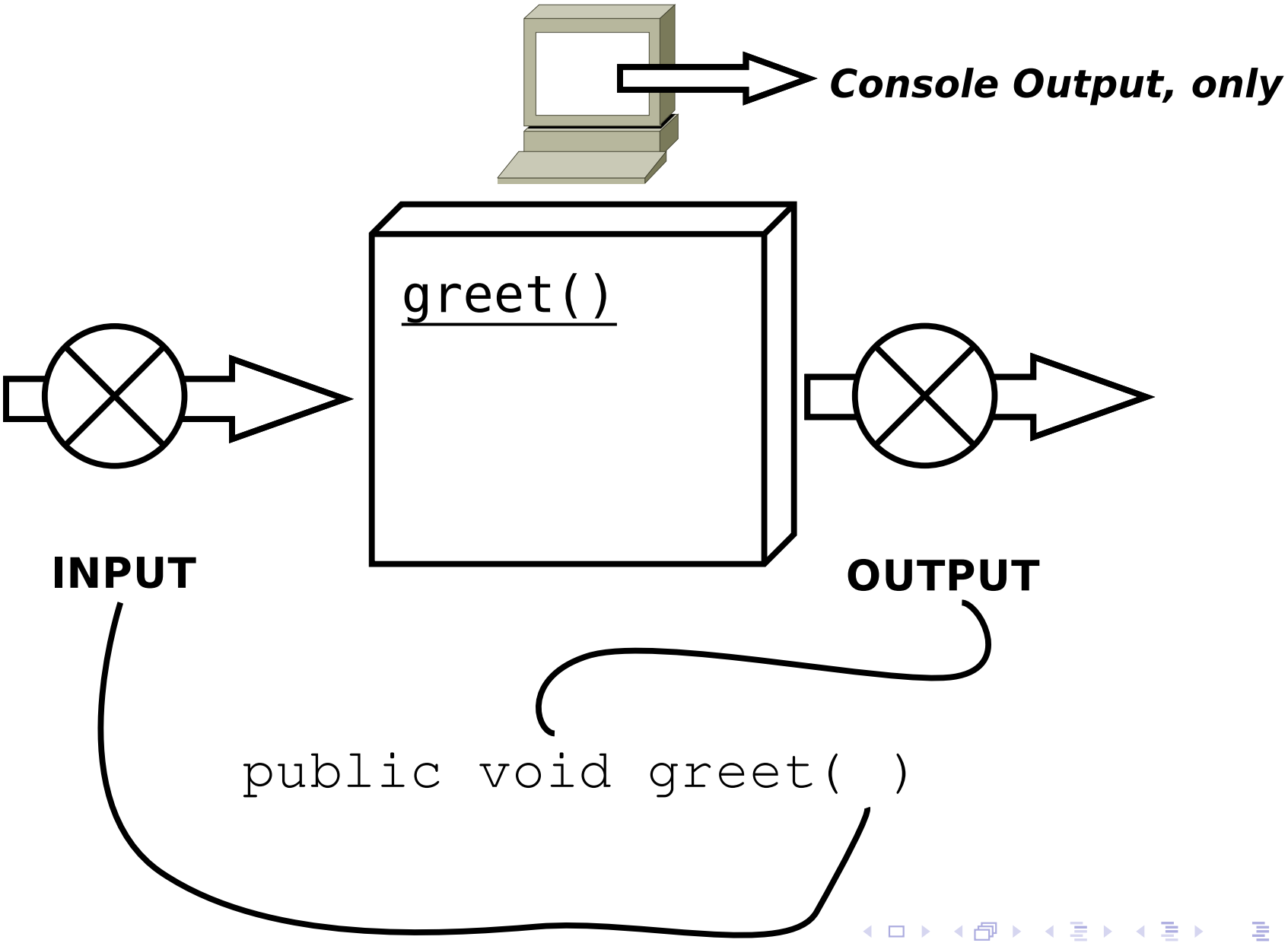
Functions: Input and Output

- What we call **methods** in Java are similar to what would be called **functions** or **procedures** in other languages.
- Functions take in some value(s)

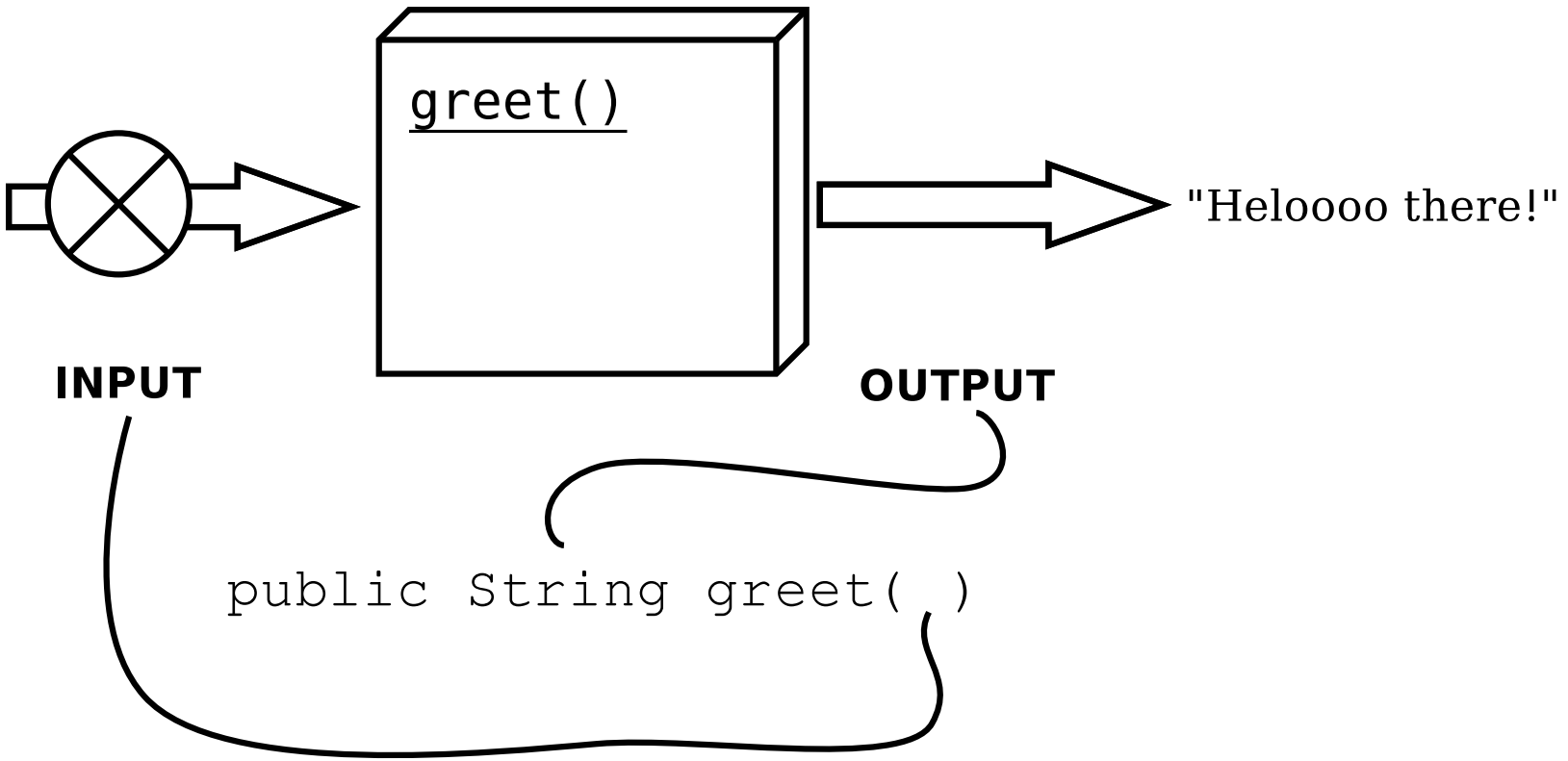
  then they "think"...

  and finally return something.
- Sometimes, a method need take no values from the caller. And sometimes nothing is returned to the caller. So complicated...

Functions: Input and Output



Modify `greet()`: return String



After changing the definition of `greet()`, it returns a `String` object to the driver class

Changes Needed for New greet()

- In Person.java, change greet() method:

```
public String greet() {
    return greeting;
}
```

- In PersonDriver.java, change call to ralph.greet():

```
String ralphGreeting = ralph.greet();
System.out.println("Ralph says:  " +
                           ralphGreeting);
```

Accessor Methods

- The method `Person()` in the `Person` class is a constructor method, which makes *instances* of the `Person` class (can also be said to create *objects* of type `Person`)

Accessor Methods

- The method `Person()` in the `Person` class is a constructor method, which makes *instances* of the `Person` class (can also be said to create *objects* of type `Person`)
- The method `greet()` *gets* information out of an object for us; we call this an accessor method

Accessor Methods

- The method `Person()` in the `Person` class is a constructor method, which makes *instances* of the `Person` class (can also be said to create *objects* of type `Person`)
- The method `greet()` *gets* information out of an object for us; we call this an accessor method
- Slang: accessor methods are *getters*

Accessor Methods

Your task (5-7min):

- Write another accessor method — `getWeight()` — that returns a `Person`'s weight to the driver as an `int`.
- Then have the driver display `ralph`'s weight using `getWeight()`.

Rest of the Period

- If you still need to show me your §5.1 program from PS #1, get it loaded up and ready to demonstrate.
- Done with PS #1?
  - download `BalClass.java` to your `Desktop` folder
  - compile and run using the terminal shell to prove it works
  - using gedit, fix the formatting of the code to have proper spacing and indentation