Lesson 11: OOP #1, Intro to OOP (W03D3)

Balboa High School

Michael Ferraro

September 2, 2015

Do Now

In your `InteractiveInts` project, *import* this source file: `MoreFun.java`. You will need to correct 5 errors in order for it to compile and run properly!

Recapping the steps for *importing* a source file into a project:

1. Download file to your computer (let's say to the *Desktop*)
2. Drag-and-drop the file to your project's `src` folder in Eclipse
3. To compile/run, click inside editor pane for `MoreFun.java` and press CTRL-F11...

Aim

Students will be introduced to OOP [Object-Oriented Programming], a core concept of Java programming.

PS #1 & #2

- PS #1 is due today! Extension needed? Ask via email. Note that there will be a credit reduction based on the number of additional days taken, starting at 1 day (-20%).
- PS #2 will be available starting next week.

What is OOP?

- OOP, or Object-Oriented Programming, refers to writing programs in a language that supports *objects*.
- Objects are abstract entities that represent some (possibly) real thing.
- Objects maintain information about the things they represent.
- It's a convenient way to manage information that programs need to work with.
- Most popular languages today OOP support included: C++, Java, Ruby, Perl, etc.

A Playful Example: Martians

- Download `MartianObjects.pde` and save to your desktop
- Start Processing in Linux:[1]
  - use the Processing launcher in the dock OR
  - from a terminal shell, type `processing &`
- Open the file: File → Open...
- Press CTRL-R to run
- See if you can figure out how to change the size of the Martians
- Can you make a new Martian?

[1]At home, you might use a web-based Processing environment like OpenProcessing.org. Copy and paste the contents of the `pde` file into the code area and click `Run`.

Programming with Class

- A programmer designs and writes *classes* that define kinds of objects.
- Think of a class as a **blueprint** describing an object's contents and behaviors

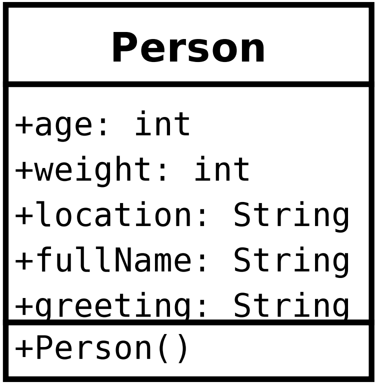
Our First Class: `Person`

Discuss: What are attributes/features that all people have?

For example, `hairColor` is one attribute.

Our First Class: `Person`

Our simple `Person` class, represented using a UML[2] diagram:



[2]Unified Modeling Language

Person.java Defines the Person Class

Create project PeopleAsObjects in current workspace and add Person.java:

```
public class Person {

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

    public Person() {
    }

}
```

Save the file, but don't compile/run — with no main(), there's nothing to

Objects are Instances of a Class

- When create an *object* of a particular class, you have created an *instance* of that class

Objects are Instances of a Class

- When create an *object* of a particular class, you have created an *instance* of that class
- How we may speak of objects:

Objects are Instances of a Class

- When create an *object* of a particular class, you have created an *instance* of that class
- How we may speak of objects:
    - “create an instance of a class”

Objects are Instances of a Class

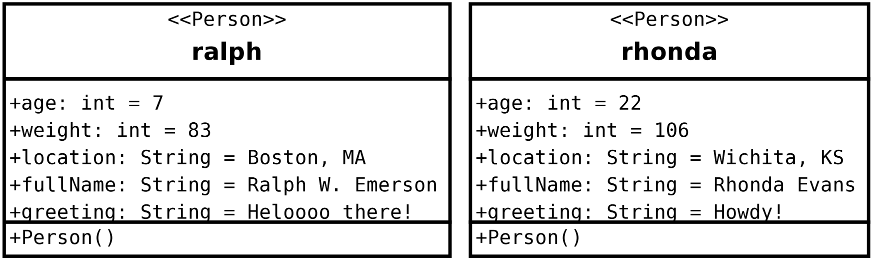
- When create an *object* of a particular class, you have created an *instance* of that class
- How we may speak of objects:
    - “create an instance of a class”
    - “instantiante a class into an object”

Objects are Instances of a Class

- When create an *object* of a particular class, you have created an *instance* of that class
- How we may speak of objects:
    - “create an instance of a class”
    - “instantiante a class into an object”
    - “an object is an instance of a class”

Two Instances of `Person`

We will create two objects, or instances, of class `Person`.



```
<<Person>>
ralph
+age: int = 7
+weight: int = 83
+location: String = Boston, MA
+fullName: String = Ralph W. Emerson
+greeting: String = Heloooo there!
+Person()
```

```
<<Person>>
rhonda
+age: int = 22
+weight: int = 106
+location: String = Wichita, KS
+fullName: String = Rhonda Evans
+greeting: String = Howdy!
+Person()
```

Where to Make Objects

- Add new file `PersonDriver.java` to project `src` folder
- Give this file a class declaration and a `main()` method

PersonDriver.java

```
public class PersonDriver {

    public static void main(String[] args) {

    }

}
```

PersonDriver.java

```
public class PersonDriver {

    public static void main(String[] args) {

              ← when program runs, create objects!

    }

}
```

PersonDriver.java

```
public class PersonDriver {

    public static void main(String[] args) {

        //create the "ralph" instance of Person
        Person ralph = new Person();
        ralph.age = 7;
        ralph.weight = 83;
        ralph.location = "Boston, MA";
        ralph.fullName = "Ralph W. Emerson";
        ralph.greeting = "Heloooo there!";

    }
}
```

Notable Features in `PersonDriver`

- Declaration of an object, `ralph`, of type `Person`
- Java keyword `new`
- `Person()` *constructor*[3]
- *Dot notation* for setting *field/instance variables*

[3]Note that the constructor `Person()` in `Person.java` isn't *necessary* — Java automatically provides a *no-args constructor* for any class if we don't write one. We include it for clarity.

Display Object Data

```
public class PersonDriver {

    public static void main(String[] args) {

        //create the "ralph" instance of Person
        //(code omitted)

        System.out.println("Ralph is " + ralph.age + " years old.");
        System.out.println("His weight is " + ralph.weight + "lbs");
        System.out.println("Unless you know him already, you should " +
            "call him " + ralph.fullName);
    }
}
```

Once the above code runs successfully, create an the `rhonda` instance of `Person` according to the "Two Instances of Person" slide, and output the field variables' values.

Next Class

Next class, you'll learn how to get and set field variables (also called *instance variables*) *safely* using methods.

PS #1 Sign-Offs

- Once you have your PS #1, §5.1 program signed off, turn in your paper form.
- If you're still working on PS#1, request an extension and keep working!

HW

If you're not done with any parts of PS #1, continue working on it.