Lesson 14: OOP #4, Inheritance (W04D2)

Balboa High School

Michael Ferraro

September 9, 2015

Do Now

- **Before** starting Eclipse, back up your existing `Person*.java` files. Start terminal shell and...

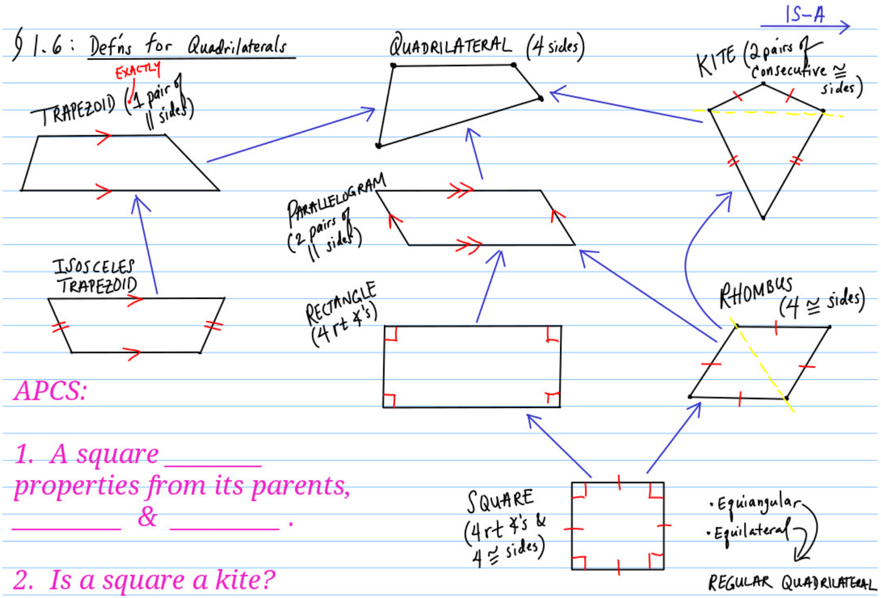
```
cd ~/MOUNTED/apcs-locker/workspace0/PeopleAsObjects/src
mv Person.java Person.java_20150909
mv PersonDriver.java PersonDriver.java_20150909
```

- Start Eclipse. Notice any errors that occur due to "missing" source files?
- Download fresh copies of those sources from here, saving them to the same `src` folder as you worked with above.
- Within Eclipse, right-click the project folder and click *refresh*. **This is how to force Eclipse to re-examine a project's directories in the file system so that it can detect changes that have occurred.**
- Read over the set of accessor & mutator methods — make sure you understand how they work!

Aim

Students will learn about *class inheritance*, extending a class into more specific subclasses.

Inheritance Example from Geometry



§1.6: Defn's for Quadrilaterals
IS-A
QUADRILATERAL (4 sides)
TRAPEZOID (EXACTLY 1 pair of ∥ sides)
KITE (2 pairs of consecutive ≅ sides)
PARALLELOGRAM (2 pairs of ∥ sides)
ISOSCELES TRAPEZOID
RECTANGLE (4 rt ∡'s)
RHOMBUS (4 ≅ sides)
SQUARE (4 rt ∡'s & 4 ≅ sides)
· Equiangular
· Equilateral
REGULAR QUADRILATERAL

APCS:

1. A square ________ properties from its parents, ________ & ________ .

2. Is a square a kite?

Return of the Martians

- Download `MartianChildren.pde` and save to your desktop[1]
- Open Processing, load `MartianChildren.pde`

[1] Running this at home? Use OpenProcessing.

Return of the Martians

- Download `MartianChildren.pde` and save to your desktop[1]
- Open Processing, load `MartianChildren.pde`
- Let's go over the chances since last time:
  - find the new classes at the end of the sketch: `EvolvedMartian` and `PrettyEyedMartian`
  - notice the new objects declared at the top of the sketch: `joe` and `tammy`
  - inside the `setup()` method, see how the new objects are initialized

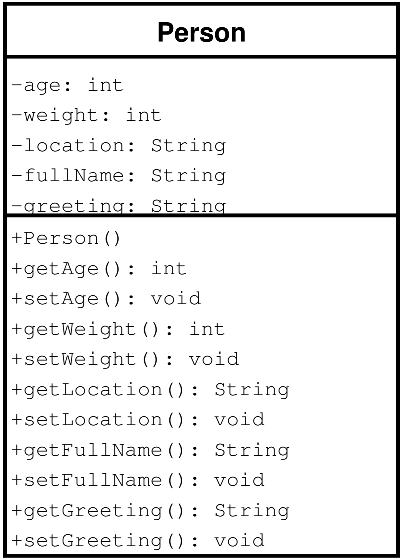
[1]Running this at home? Use OpenProcessing.

Return of the Martians

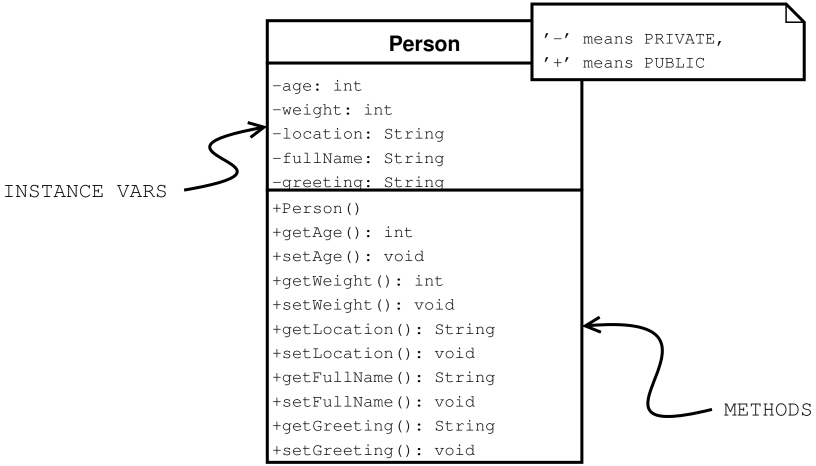
- Download `MartianChildren.pde` and save to your desktop[1]
- Open Processing, load `MartianChildren.pde`
- Let's go over the chances since last time:
  - find the new classes at the end of the sketch: `EvolvedMartian` and `PrettyEyedMartian`
  - notice the new objects declared at the top of the sketch: `joe` and `tammy`
  - inside the `setup()` method, see how the new objects are initialized
- What does `extends` appear to do?

[1]Running this at home? Use OpenProcessing.

Updated State of Person Class



Updated State of Person Class



So Many Kinds of People

- Let's say that we want to have more specific kinds of `Persons`:

So Many Kinds of People

- Let's say that we want to have more specific kinds of `Persons`:
  - `Teacher extends Person`

So Many Kinds of People

- Let's say that we want to have more specific kinds of `Persons`:
  - `Teacher extends Person`
  - `Student extends Person`

So Many Kinds of People

- Let's say that we want to have more specific kinds of `Persons`:
  - `Teacher extends Person`
  - `Student extends Person`
- `Teacher` and `Student` are subclasses of `Person`: Each has all the qualities of a `Person` — like `age`, `location`, and `greeting` — yet each can have additional instance variables.

So Many Kinds of People

Discuss: What information shall we store that is specific to `Teachers`? How about `Students`?

So Many Kinds of People

Discuss: What information shall we store that is specific to Teachers? How about Students?

- Teacher subclass
    - int yearsTeaching
    - String primarySubject

So Many Kinds of People

Discuss: What information shall we store that is specific to `Teachers`? How about `Students`?

- `Teacher` subclass
  - `int yearsTeaching`
  - `String primarySubject`
- `Student` subclass
  - `int gradeLevel`
  - `String intendedMajor`

Code for Teacher class

```
public class Teacher extends Person {

    int yearsTeaching;
    String primarySubject;

    public Teacher() {
        //no-args constructor
    }
}
```

Code for Teacher class

```
public class Teacher extends Person {

    int yearsTeaching;
    String primarySubject;

    public Teacher() {
        //no-args constructor
    }
}
```

- A `Teacher` object IS-A `Person`

Code for Teacher class

```
public class Teacher extends Person {

    int yearsTeaching;
    String primarySubject;

    public Teacher() {
        //no-args constructor
    }
}
```

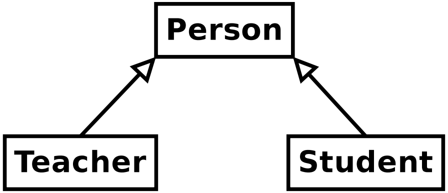
- A `Teacher` object IS-A `Person`
- `Person` is the parent class of class `Teacher`

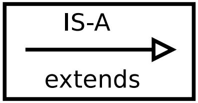
Code for Teacher class

```
public class Teacher extends Person {

    int yearsTeaching;
    String primarySubject;

    public Teacher() {
        //no-args constructor
    }
}
```

- A `Teacher` object IS-A `Person`
- `Person` is the parent class of class `Teacher`
- Since the `Teacher` class `extends` `Person`, it has `Person`'s field variables (e.g., age and `weight`), as well as its accessor and mutator methods

Class Inheritance Hierarchy





Create a Teacher

- In the driver class, create a new `Teacher` instance:
  `Teacher teacher1 = new Teacher();`
- Set `teacher1`'s state as follows:

var	value
`age`	`31`
`weight`	`168`
`fullName`	`Mr. Sparkle`
`yearsTeaching`	`14`
`primarySubject`	`Math`

Create a Teacher

- In the driver class, create a new `Teacher` instance: `Teacher teacher1 = new Teacher();`
- Set `teacher1`'s state as follows:

var	**value**
`age`	`31`
`weight`	`168`
`fullName`	`Mr. Sparkle`
`yearsTeaching`	`14`
`primarySubject`	`Math`

- Wait. . . we don't have accessor/mutator methods for `Teacher.yearsTeaching` and `Teacher.primarySubject` yet! Create these — and mark the corresponding instance variables `private`.

Code Snippet for PersonDriver class

```
...
 Teacher teacher1 = new Teacher();
 teacher1.setAge(31);
 teacher1.setWeight(168);
 teacher1.setFullName("Mr. Sparkle");
 teacher1.setYearsTeaching(14);
 teacher1.setPrimarySubject("Math");

 String teacher1Stats;
 teacher1Stats = teacher1.getFullName() + " has been teaching ";
 teacher1Stats += teacher1.getPrimarySubject() + " for ";
 teacher1Stats += teacher1.getYearsTeaching() + " years!";
 System.out.println(teacher1Stats);
...
```

Rest of Class. . .

- Didn't address PS02 §1.3 yet? Do it now: Create new project `ps02` add file, push (read details in problem set).
- Make sure you understand what we've done thus far! If you're having any issues understanding — or you're having a hard time with the hands-on exercises — please ask for a classmate's help and/or mine!
- Proceed on to the HW slide →

HW

Continue working on PS #2.

- You should be nearly finished with §§1-5 by now.
- Start looking over §6.