Lecture 2 - Sep. 12

Review on OOP

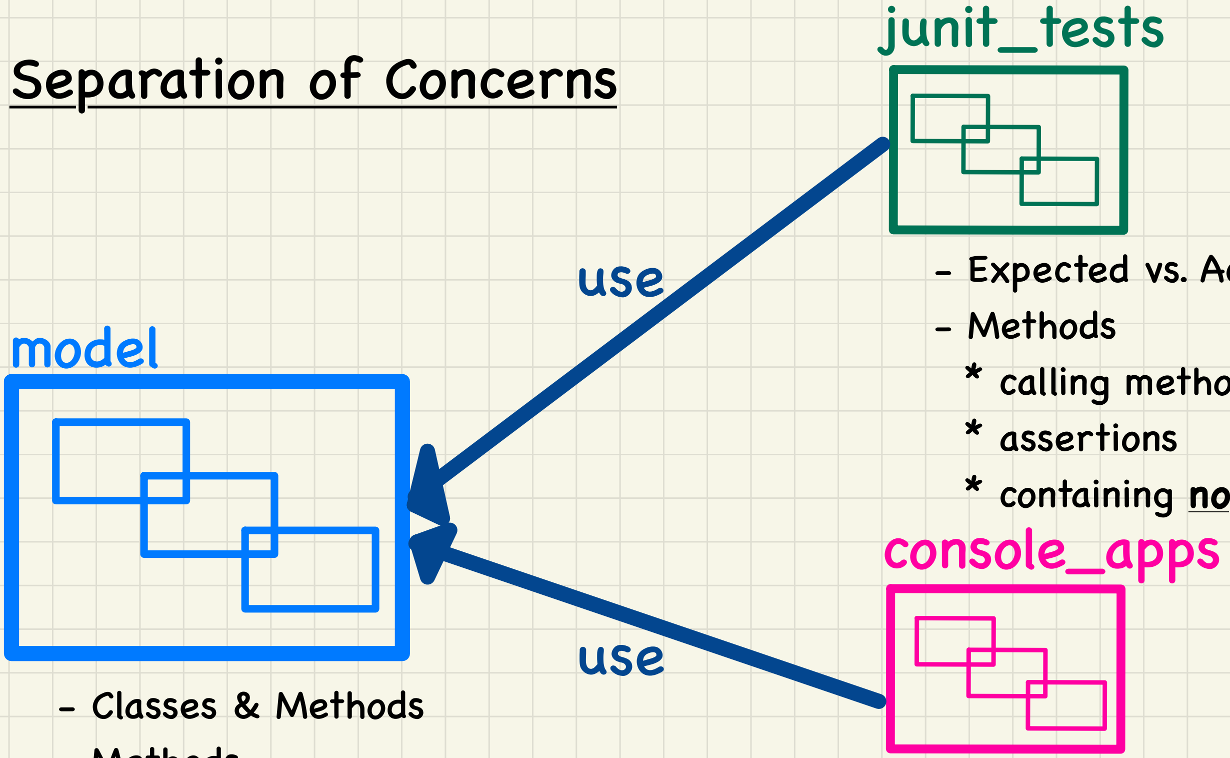
Object Orientation
Classes, Objects, Methods

- Lab0 Part 1
  + Eclipse: Your Machine vs. RemoteLabs

    latest. try. eecs account.
  + Tutorial Videos ✓
  + PDF guides:
    * Inferring Java Classes from JUnit Tests
    * Programming Pattern: Array Attributes
- Scheduled Lab this Week: Optional Q&A
- Office Hours

Reading.. up to slide 49.

Separation of Concerns



- Expected vs. Actual Values
- Methods
  * calling methods from model
  * assertions
  * containing **no** print statements

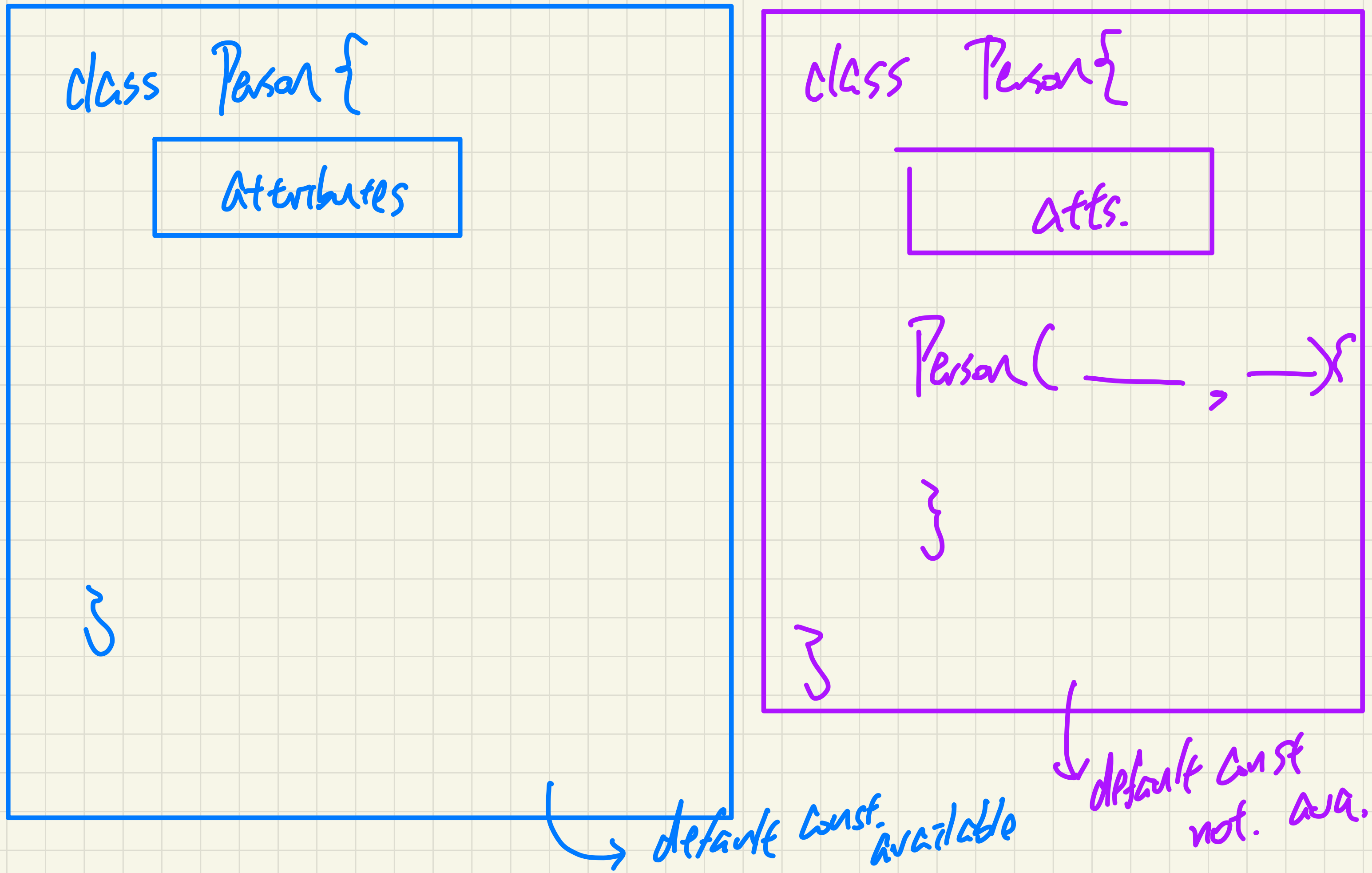
- Classes & Methods
- Methods
  * constructors
  * accessors: return statements
  * mutators: **no** return statements
  * containing **no** print statements

- main method (entry point of execution)
  * reading inputs from keyboard
  * calling methods from model
  * producing outputs to console (print)
  * containing **no** return statements

attributes : should be private

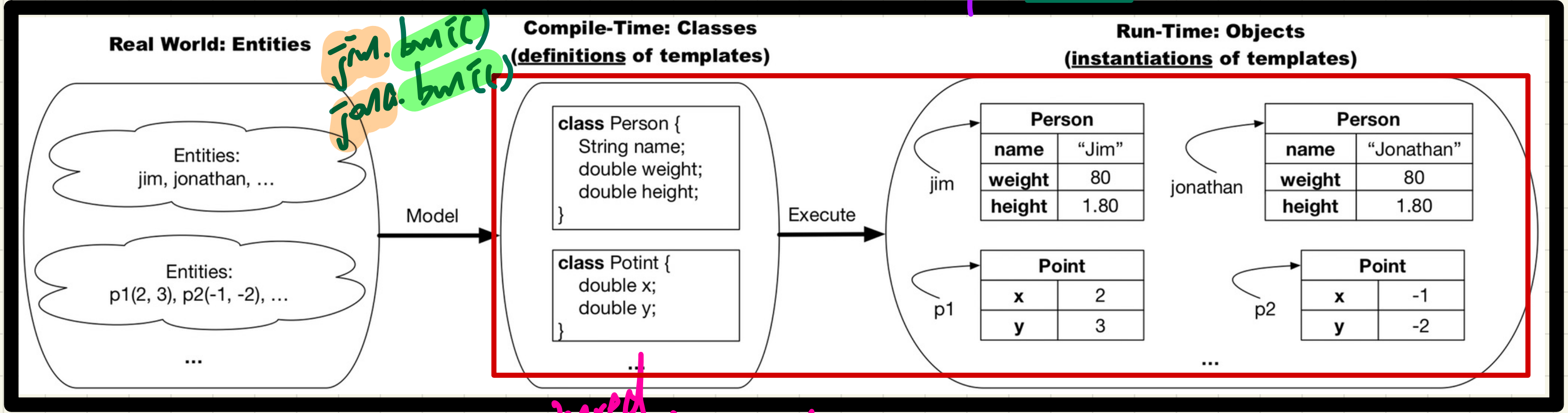
methods : 1. helper methods : private

2. to be called by other classes:
public


class Person {
Attributes
}
default const. available
class Person {
atts.
Person(____ , ____){
}
}
default const not. ava.

Observe-Model-Execute Process

Context objects p1 dist( ) p2 dist( )



shared by all Person objects

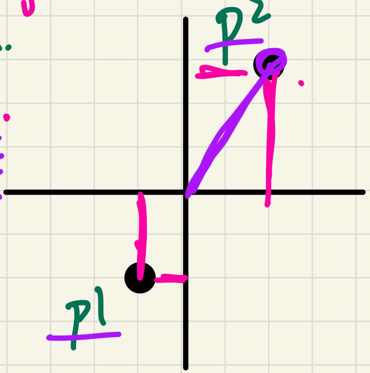
Entities: jim, jona.

Attributes: w., h.

Changes: gainWeight

Inquiries: getBMI

Template: Person



Entities: p1, p2

Attributes: x, y.

Changes: ↑x ↓y

Inquiries: dist

Template: Point

Modelling: from Entities to Classes

Identify Critical Nouns & Verbs

mutator/ setter

classes, attributes, accessor/ getter

Example 1

Point (class)

x, y attributes

Points on a two-dimensional plane are identified by their signed distances from the X- and Y-axises. A point may move arbitrarily towards any direction on the plane. Given two points, we are often interested in knowing the distance between them.

moveUp Down East West.

Example 2

A person is a being, such as a human, that has certain attributes and behaviour constituting personhood: a person ages and grows on their heights and weights.

Object Oriented Programming (OOP)

- Templates (compile-time Java classes)
  - attributes (common around instances)
  - methods
    - constructors
    - accessors/getters
    - mutators/setters
  - Eclipse: Refactoring
- Instances/Entities (runtime objects)
  - instance-specific attribute values
  - calling constructor to create objects
  - using the "dot notation", with the <u>right</u> contexts, to:
    - get attribute values
    - call accessors or mutators

Constructors not using this Keyword

model

```
public class Person {
    /*
     * Attributes.
     * Person instances have the same attribute names.
     * Person instances have specific attribute values.
     */
    double weight;
    double height;

    /*
     * Constructors
     */
    public Person() {

    }

    public Person(double ~~newWeight~~, double ~~newHeight~~) {
        weight = ~~newWeight~~;
        height = ~~newHeight~~;
    }
}
```

72 65 1.81 1.67

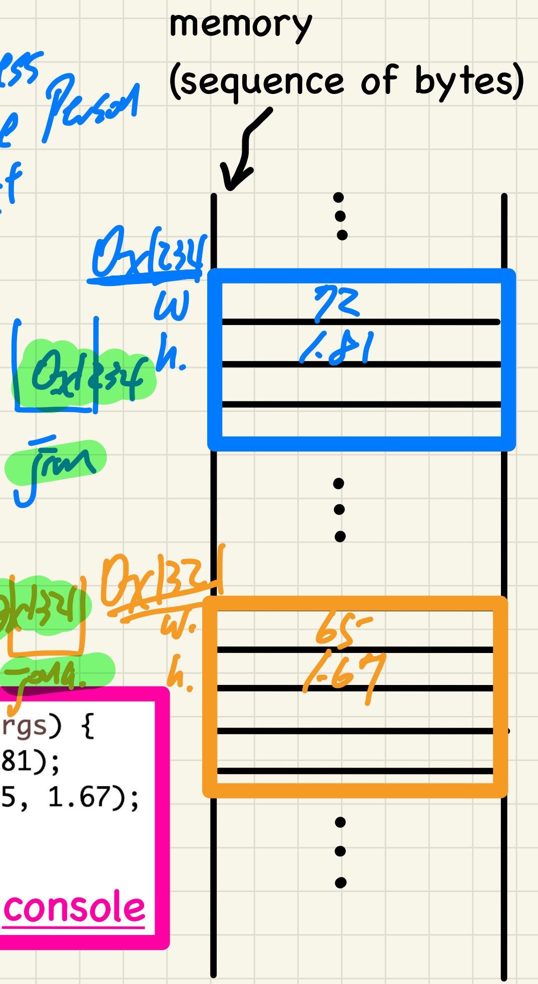
JUnit

```
@Test
public void test_1() {
    Person jim = new Person(72, 1.81);
    Person jonathan = new Person(65, 1.67);
    assertTrue(jim != jonathan);
    assertFalse(jim == jonathan);
    assertNotSame(jim, jonathan);
    assertNotEquals(jim, jonathan);
}
```

store address of some Person object

console

```
public static void main(String[] args) {
    Person jim = new Person(72, 1.81);
    Person jonathan = new Person(65, 1.67);
    System.out.println(jim);
    System.out.println(jonathan);
}
```



- Default Constructor?
- Parameters vs. Arguments
- Reference Variables