Lecture 2 - Sep. 10

Review of OOP

Observe-Model-Execute Process
Java Data Types
NullPointerException
Parameters vs. Arguments
Scope of Variables

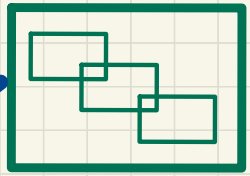
Lab 01 !

Lab 02 → reference-typed attributes

Lab Wed
↳ helper methods.

Separation of Concerns

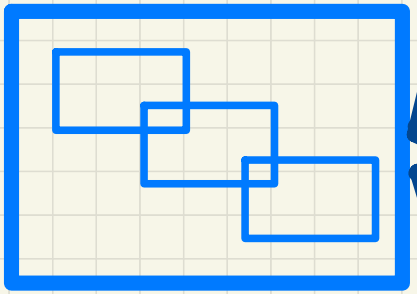
junit_tests



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

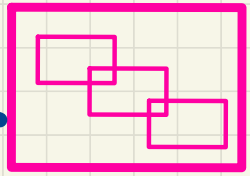
use

model



use

console_apps



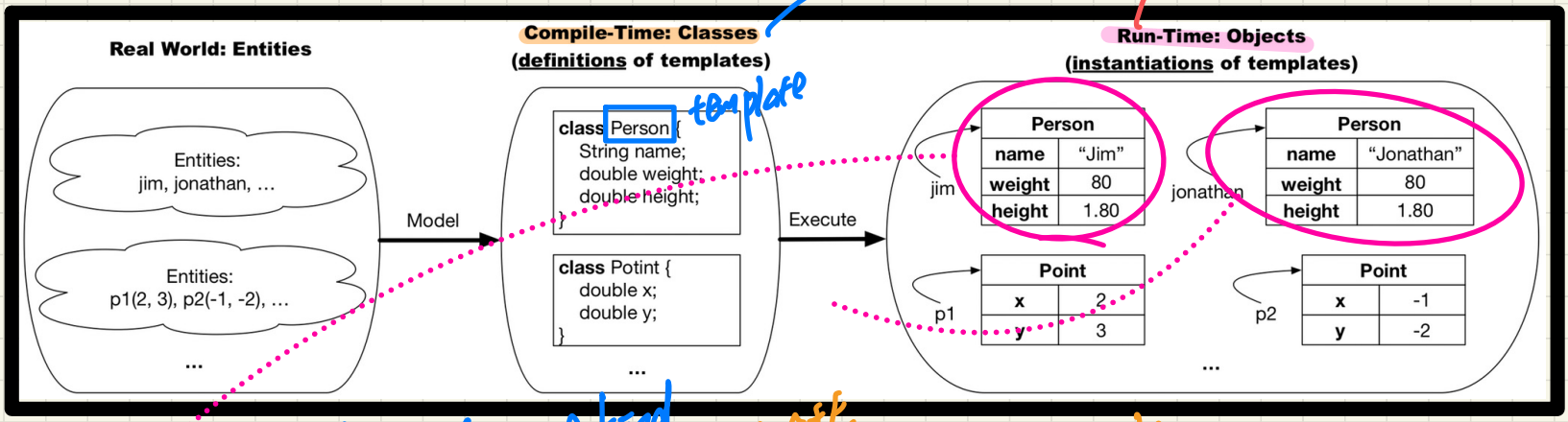
- Classes & Methods
- Methods
  * constructors
  * accessors: return statements
  * mutators: **no** return statements
  * containing **no** print statements

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

Observe-Model-Execute Process

development of code in editor

1. compiles
2. executes



template

h. 1.8

different entities of the same kind

h. 1.7

↳ common attr.
common behaviour ↳ change weight

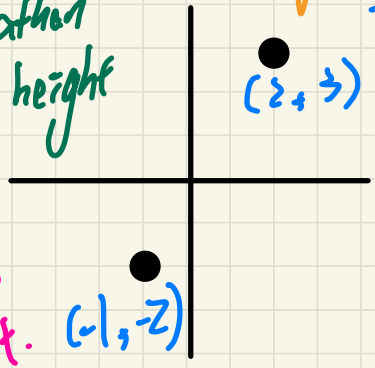
Entities: jim, jonathan

Attributes: weight, height

Changes:

Inquiries: instance-specific

Template: Person. values for att.



Exercise.

Entities:

Attributes:

Changes:

Inquiries:

Template:

Object Oriented Programming (OOP)

- Templates (compile-time Java classes)
  + attributes (common around instances)
  + methods
    * constructors
    * accessors/getters
    * mutators/setters
  + Eclipse: Refactoring

change
↳ - rename variables
- rename methods

- Instances/Entities (runtime objects)
  + instance-specific attribute values
  + calling constructor to create objects
  + using the "dot notation", with the right contexts, to:
    * get attribute values
    * call accessors or mutators

obj.m()
obj.m1().m2()....

Modelling: from Entities to Classes

Identify Critical Nouns & Verbs

Example 1

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.

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.

OO Thinking: Templates vs. Instances

(Exercise)!

Templates

Person

Common
Attribute Definitions
(Types)

double weight
double height

Common
Behaviour Definitions
(Headers/API)

double getWeight()
void gainWeight(double w)
double getBMI()

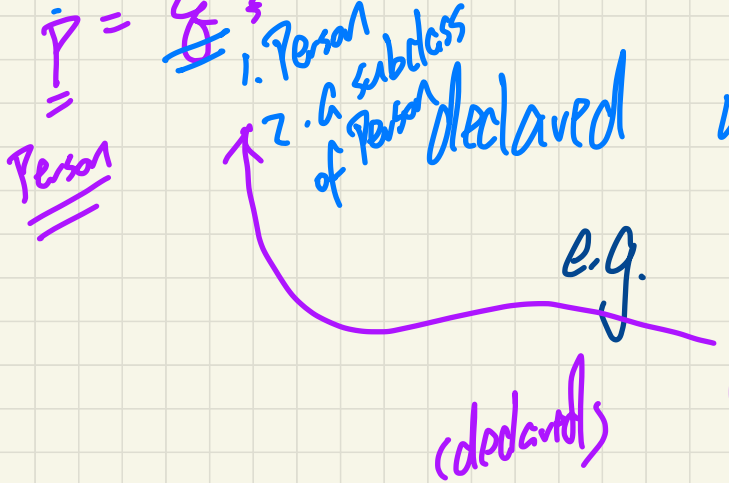
Instance-Specific
Attribute Values

object 1: W: 46.3
object 2: W: 70.7

Instance-Specific
Behaviour Occurrence

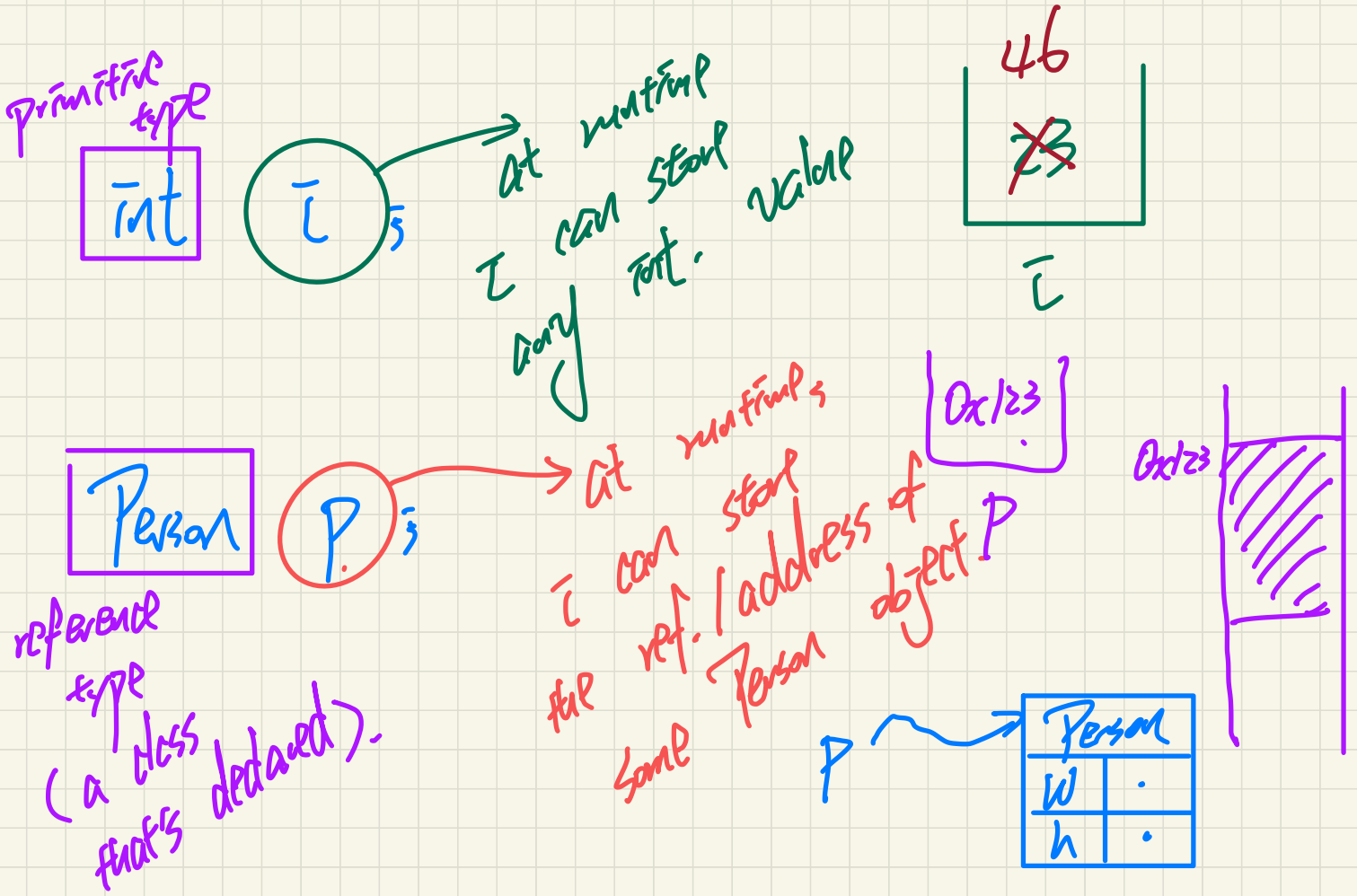
objects.
diff. content
obj1
obj2
gainW(2)
gainW(2)
same method with same input invoked

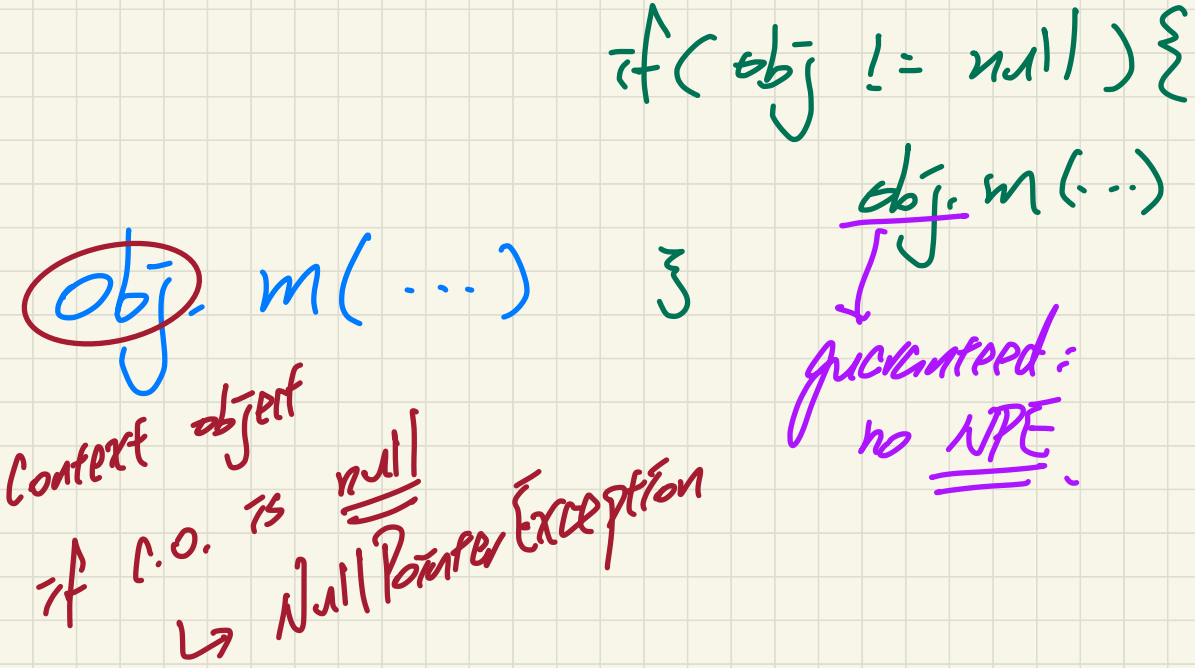
- In Java, each variable must be declared with a type.



i = 23 ;
i = 46 ;
i = 3b. 7 ; X
i = p ; X

- A variable's type denotes the set of values that is allowed to be stored in that variable.

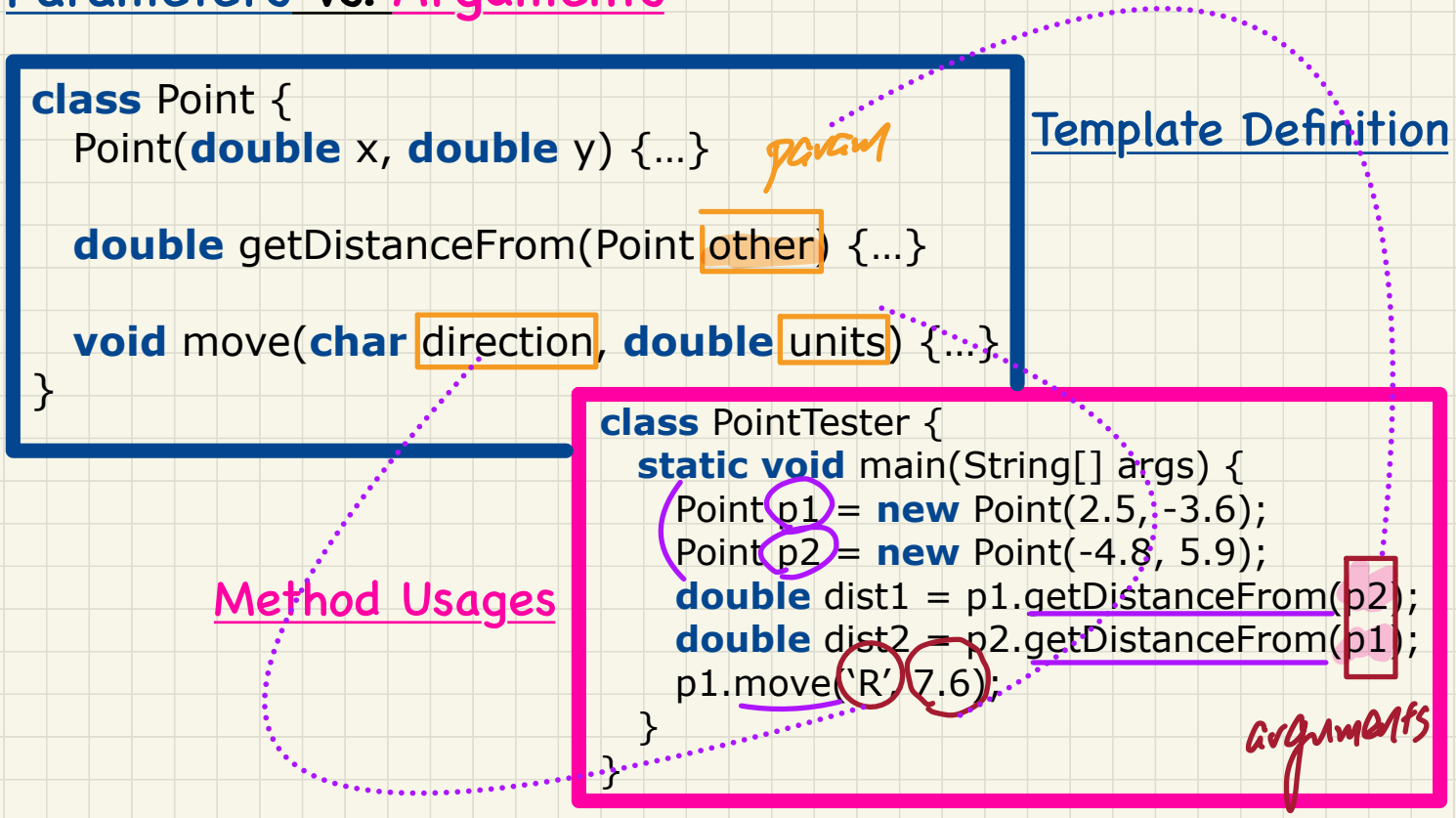

primitive type
int
i
5
At runtime i can store any int. value
46
23
i
Person
p
5
reference type (a class that's declared).
At runtime i can store the ref. / address of some Person object P
0x123
0x123
P
Person
W
h


if(obj != null){
obj.m(...)
}
obj.m(...)
Context object
if c.o. is null
↳ NullPointerException
guaranteed:
no NPE.

slides 16 – 47

↳ self study.

Parameters vs. Arguments



Scope of Variables in a Class

- Class-level variables may be accessed/modified between methods.

[Assume for now: non-static]

- Method-level parameters may be accessed within the declared method only.
- Method-level (local) variables may be accessed/modified within the declared method.

```
class MyClass {
  int i;
  int j;
  void m1 (int p1) {
    int m;
  }
  void m2 (int p2) {
    int n;
  }
}
```

class-level

i = 23; m = p1; m = p2; X

i = 46; n = i; n = m; X