Lecture 3 - May 13

Review of OOP

NullPointerException
Parameters vs. Arguments
Scope of Variables
Default vs. Customized Constructors
Variable Shadowing, Use of this

Announcements/Reminders

- Today's class: notes template posted
- LabOP1 due on Tue (exact deadline on instructions PDF)
- Priorities:
  - + LabOP2 (tutorial videos + PDFs)
  - + Due: Next Monday
- Lab1 to be released next Monday
- ProgTest0 next Friday (May 23)
- Lab this Friday (May 16):
  - + [≈ 40 min] Mock-up ProgTest0
  - + [≈ 40 min] Q&As (TAs, Jackie)

NullPointerException

→ runtime, logical error.

when context object is null

method call

obj.m(...)

- obj → context object
- m → method being called
- Go to the address stored in obj, invoke the def. of m on that memory block

when this is non-existing (null) ⇒ NPE.

e.g.

```
C0.m1().m2().m3()
```

(1) can be null
(2) can be null
(3) can be null

break point

Use "Expression" panel in debugger to find out.

(1) Compiles
(2) NPE at r.t.!

Q. How many possible ways can this NPE occur?

Slides 16 ~ 47

assigned readings!

Parameters vs. Arguments

def. of methods

use/call of methods

parameters.

```
class Point {
   Point(double x, double y) {...}

   double getDistanceFrom(Point other) {...}

   void move(char direction, double units) {...}
}
```

Template Definition

```
class PointTester {
   static void main(String[] args) {
      Point p1 = new Point(2.5, -3.6);
      Point p2 = new Point(-4.8, 5.9);
      double dist1 = p1.getDistanceFrom(p2);
      double dist2 = p2.getDistanceFrom(p1);
      p1.move('R', 7.6);
   }
}
```

Method Usages

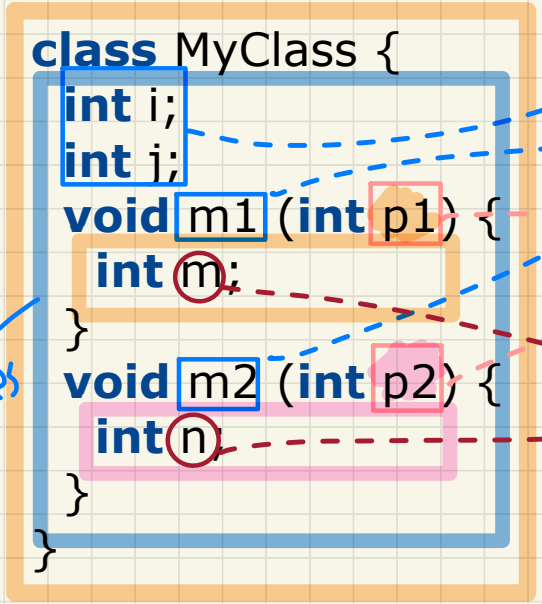
calls must be compatible with parameter types.

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.



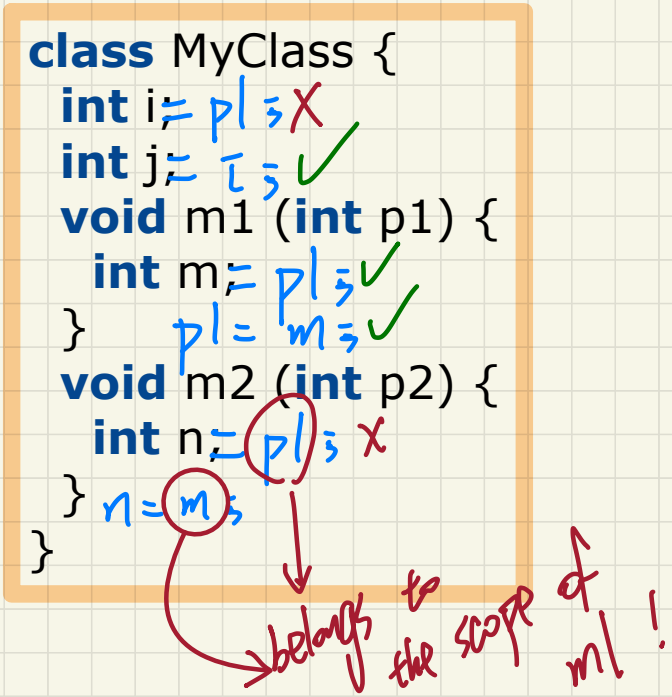
attributes (class-level variables)

methods (class-level)

method parameters (method level)

local variables (method-level)

scope of attributes


class MyClass {
int i;
int j;
void m1 (int p1) {
int m;
}
void m2 (int p2) {
int n;
}
}
p1 = i; X
i = j; ✓
p1 = m; ✓
n = m; ✓
p1; X
belongs to the scope of m1 !

Constructors not using this Keyword

jim == jonathan False

jim != jonathan True

```java
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;
    }
}
```

model

```java
@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);
}
```

JUnit

```java
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);
}
```

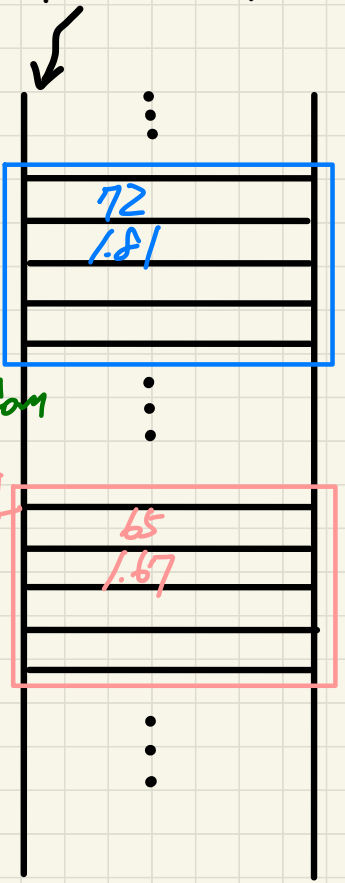
console

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

0x123 jim

0x789 jonathan

memory (sequence of bytes)



Is add. stored in jim diff. from add. stored in jonathan

jim != jonathan

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; weight
        height = newHeight; height
    }
}
```

weight height

this.weight

Question

- What if names of parameter & attribute are the same?
- implicit "this"

deliberately those param. names same as those attributes that we want to modify.