Important For This Lecture

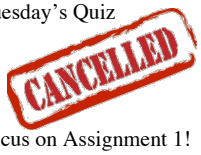
Readings

- Sections 2.1 – 2.4

We may not get to everything on the slides today. You are still responsible for reading them in the text for the next lab.

Announcements

- Tuesday's Quiz

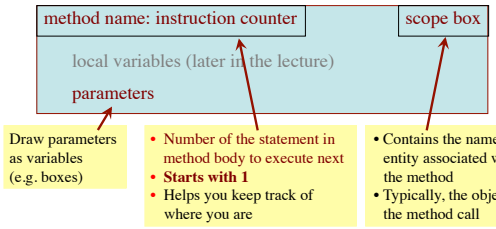


- Focus on Assignment 1!
  - Do not wait until Monday
- 1-on-1s for next 2 weeks
  - Slots still available

How Do Methods Work?

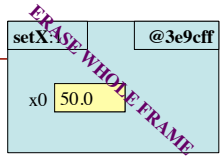
Draw template on a piece of paper

- **Method Frame**: Formal representation of a method call
- *Remember* that methods are inside objects (folders)

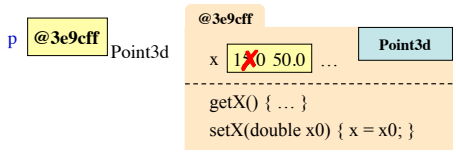


Example: p.setX(50.0);

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the method body
   - Look for variables in the frame
   - If not there, look in folder given by the scope box
4. Erase the frame for the call




```
public void setX(double x0) {
    x = x0;
}
```

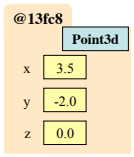


Static Methods

- **Static** methods are tied to a class (e.g. file drawer)
- They must not access the fields!
  - Fields are in the folders
  - Folders have different field values
- Their method calls are different:
  - <Class-Name>.<Method-Call>
- **Example**: Math methods in lab
  - Math.ceil(5.6);
  - Math.min(1,2);
  - Math.sqrt(5);

Class

Object

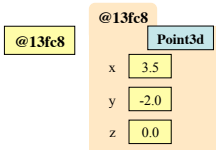


Defining Static Methods

Regular Version

```
/** Yields: "at least one of the
 * coordinates of this point is 0" */
public boolean hasAZero() {
    return x == 0 || y == 0 || z == 0;
}
```

Call: `q.hasAZero();`



Static Version

```
/** Yields: "at least one of the
 * coordinates of the point q is 0" */
public static boolean
        hasAZero(Point3d q) {
    return q.x == 0 || q.y == 0
                    || q.z == 0;
}
```

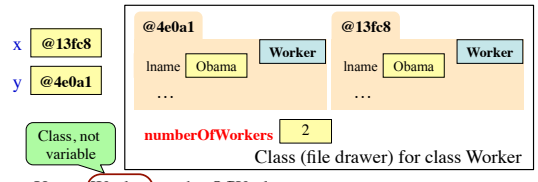
Call: `Point3d.hasAZero(q);`

Goes in the **scope box**

Static Variables

- Static variable is a *single entity in the class*
  - Used to hold information about all objects
- Declare it just like a field declaration

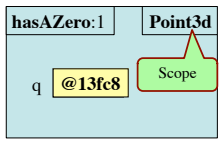
```
public static int numberOfWorkers;  // no. of Worker objects created
```



- Usage: `Worker.numberOfWorkers`

Method Model for Static Methods

1. Draw a frame for the call
   - Scope box contains class!
2. Assign the argument value to the parameter (in frame)
3. Execute the method body
   - Look for variables in the frame
   - If not there, look in **static variables** in **class** in scope box
4. Erase the frame for the call




```
public static boolean
        hasAZero(Point3d q) {
    return q.x == 0 || q.y == 0
                    || q.z == 0
}
```

Conditionals: If-Statements

Format

```
if (<boolean-expression>) {
    <statement>;
    …
    <statement>;
}
```

Example

```
/* Put x in z if it is positive */
if (x > 0) {
   z = x;
}
```

Execution:
if the <boolean-expression> is true, then execute all of the statements inside of the braces ({ })

Conditionals: If-Else-Statements

Format

```
if (<boolean-expression>) {
    <statement>;
    …
} else {
    <statement>;
    …
}
```

Example

```
/* Put max of x, y in z */
if (x > y) {
   z = x;
} else {
   z = y;
}
```

Execution:
if the <boolean-expression> is true, then execute all statements in braces after if; otherwise execute statements in braces after else

Application: Invariants

```
public class Worker {
  private String lname;  // Last name
                         // never null

  /** Set worker's last name to n
   * Precondition: Cannot be null
   */
  public void setName(String n) {
    lname= n;
  }

}
```

```
public class Worker {
  private String lname;  // Last name
                         // never null
  /** Set worker's last name to n
   * OR to "" if n is null
   */
  public void setName(String n) {
    if (n == null) {
      lname = ""
    } else {
      lname = n;
    }
  }
}
```

Local Variables

- **Local variable**: declared inside a *method body*
- Four types of variables:
  - Fields (in folders)
  - Parameters (method header)
  - Static (in file drawer)
  - Local (method body)
- Local variables are very useful with if-statements
  - Hold temporary values
  - "Scratch computation"

```
// swap x, y
// Put the larger in y
if (x > y) {
  int temp;
  temp = x;
  x = y;
  y = temp;
}
```

x 0 y 3

temp 3

Local Variable Scope

```
/** Yields: the max of x and y */
public static int max(int x, int y) {
   // Swap x and y
   // Put the max in x
   if (x < y) {
      int temp;
      temp= x;
      x= y;
      y= temp;
   }

   return x;
}
```

scope of temp

Cannot use temp down here. You will get an error!

- **Scope of local variable**: the places it can be used
- Only inside a "block"
  - Following the declaration
  - Inside of the braces {}