Useless Fact of the Day

A penguin named Nils Olav II currently holds the rank of Colonel-in-Chief in the Norwegian Army.

Variable Categories

Three different species of variables:

- instance variables
- static variables
- local variables
**Instance Variables**

- Also known as a class’s **data members**
- Declared at the top of a class definition’s body, before the class’s methods are defined
- There is a different copy of the variable for each instance (object) created from a class
- The variable is accessible throughout the class (and from other classes, if public, using the dot operator)
- The variable “lives” as long as the instance lives

```java
public class Point2D.Double
{
    public double x;
    public double y;
    public double distance(Point2D arg)
    {
        ...
    }
    ...
}
```

**Static Variables**

- Also declared at the top of a class definition’s body, before the class’s methods are defined
- There is only one copy of the variable, shared by all instances (objects) created from the class
- Constants can even be accessed if no instances of the class have been created!
- All **constants** are static
- Use the keyword **final** to specify that a variable is constant (it can never be changed after first initialized)
- These variables are accessible throughout the class (and from other classes, if public, using the dot operator)

```java
public class LanderTracker implements Tracker
{
    public static final double GRAVITY_ACCEL = 9.8;
    public static double windSpeed;
    public Point2D.Double velocity;
    public Point2D.Double translation;
    public Point2D.Double getTranslation()
    {
        ...
    }
    ...
}
```

We can access the constant even without an instance of the LanderTracker class:

```java
myVariable = LanderTracker.GRAVITY_ACCEL;
```

If we were to create two LanderTrackers (called lt1 and lt2, for example), their value for windSpeed would always be the same:

```java
lt1.windSpeed == lt2.windSpeed  ->  true
```
Local Variables

- Declared at the top of a method body
- ...or sometimes at the top of an if statement body
- Also, method parameters are local variables in the method
- The variable “lives” only in the code block in which it was created
- (between the squiggly braces, { })

Scope

- Where a variable “lives” is called the scope of the variable
- WARNING:
  - Different variables can have the same name if they have different (and non-overlapping) scope!

Public vs. Private

- Anything declared at the “top level” of a class body (variables, constants, and methods) can be either public or private
- public things can be “seen,” or used, from code outside of the class containing the things (using the dot operator)
- private things can not be seen outside of the class containing the things
**Public vs. Private: Example**

```java
class Point2D{
    public double x;
    private double y;

    public double distance(Point2D arg) {
        ...  // Implementation
    }
}
class Rectangle {
    public Point2D location;

    public double distance(Point2D arg) {
        location.x = 5;
        location.y = 5;
    }
}
```

Since "x" is public and "y" is private, this line is OK...
...but this line will be an error.

Why would you ever want to use `private`?

- Controlled access and modification of information in the class
- Make the class a “black box” machine
- Let the user of the class only see what he needs to see -- hide the internal workings
- Allow someone to only see data without being able to change it (using accessor methods)

**Example**

```java
class Car {
    private double speed;
    ...

    public double getSpeed() {
        return speed;
    }
    public void pressGas(double howHard) {
        ...
    }
    public void pressBreaks(double howHard) {
        ...
    }
    private void doEngineStuff(...) {
        ...
    }
}
```

Let's say you make a Car object and call it “myCamaro.”

Since “speed” is private, you can't modify it directly (you couldn't say “myCamaro.speed = 1000”).

You can see it, though, by saying “myCamaro.getSpeed()” (since “getSpeed” is a public method).

You can also modify it by using the public methods “pressGas” and “pressBreaks.”