# Visibility

→ A declaration introduces an entity (variable, method, class) into a program and includes an identifier (name) to represent that identity

[SUMMARY TABLE]

| | modifier keyword | UML | Top-level | Member-level | Local-level |
|---|---|---|---|---|---|
| Private | private | − | X | ✓ | X |
| Package-private | | ~ | ✓ | ✓ | X |
| Protected | protected | # | X | ✓ | X |
| Public | public | + | ✓ | ✓ | X |

- Top-level Declaration – outermost declaration, includes classes and interfaces
  → not nested

  ex) public class MyClass {...}
  public interface MyInterface {...}

- Member-level Declaration – any declaration of a class or interface member. Members can include the constructors, methods, variables and constants (both static or nonstatic/instance) of a class/interface

  * Never include local-level declarations *
  (declared within methods/blocks)

- Local-level Declaration – any variable declaration that is local or in scope to a particular method. Local variables include its parameters & any variables declared in body of method

ex]
```
public class IceCream { // top-level
    private String flavor; // 'flavor' is member level
    public IceCream(String flavor) { // 'IceCream' is member-level here, & flavor is local-level
        int price = 10; // local-level
        this.flavor = flavor;
    }
}
```
this → instance one

# Package-Private Visibility

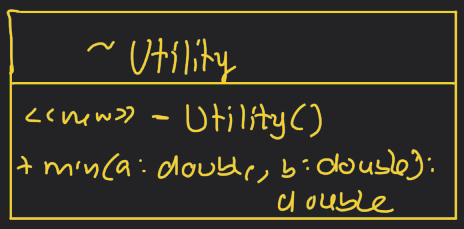
* only top-level & member-level declarations are allowed to be package-private
  - same class ✓
  - same package ✓
  - child class X
  - elsewhere X

* no modifier keyword
  - for top-level & member-level, omitting a visibility modifier will cause compiler to treat declared thing as package private.
  - different for interface: for member-level declarations, it will default to public *

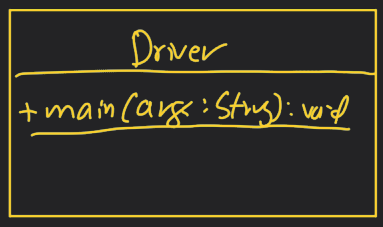
* In UML, if you don't include "~" then it defaults to public

→ Javadoc does not include package private declarations but can be included by adding the -package command-line arg

CS1302.models




CS1302.store




Uses

```
package cs1302.models;
class Utility { ... }
```

↳ defaulted to package private visibility

```
package cs1302.store;
import cs1302.models.Utility;
```

↳ different package & since class is package private, it's not visible from this line so error while compiling

→ 2 files are in different packages whenever their package statements are not identical. cs1302.foo & cs1302.foo.bar are different

→ Private Constructor Use Cases:

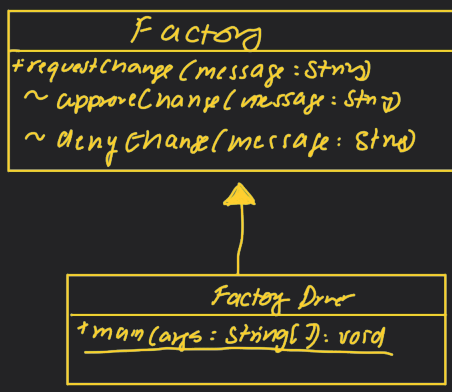
1) Prevent Object Creation: If only static methods & constants in class
2) Restrict object Creation: restrict the total number of objects of a particular class that can be created

# Package-Private Continued

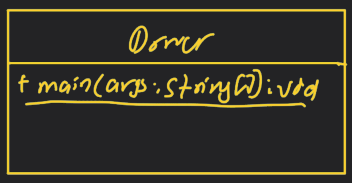
→ For member-level declarations:

* think: suppose a factory has a contract with a store to produce some product. The driver program on the factory's side should be able to access methods to request, approve, and deny changes but overall class design should not allow for this.

CS1302.Factory




CS1302.Store




```
// inside FactoryDriver.java
public static void main ... {
    Factory factory = new Factory();
(1) factory.requestChange("dec pico");
(2) factory.approveChange("dec pico");
(3) factory.denyChange("dec pico");
```

→ Lines 1, 2, 3 attempt to access a diff member of Factory class from SAME PACKAGE

→ All three methods in Factory are visible by FactoryDriver because one is public & other 2 are package-private

```
// inside Driver.java
public static void main ... {
    Factory factory = new Factory();
(4) factory.requestChange("inc pico");
(5) factory.approveChange("inc pico");
(6) factory.denyChange("inc pico");
```

→ Lines 4, 5, 6 try to do the same but from a DIFFERENT PACKAGE

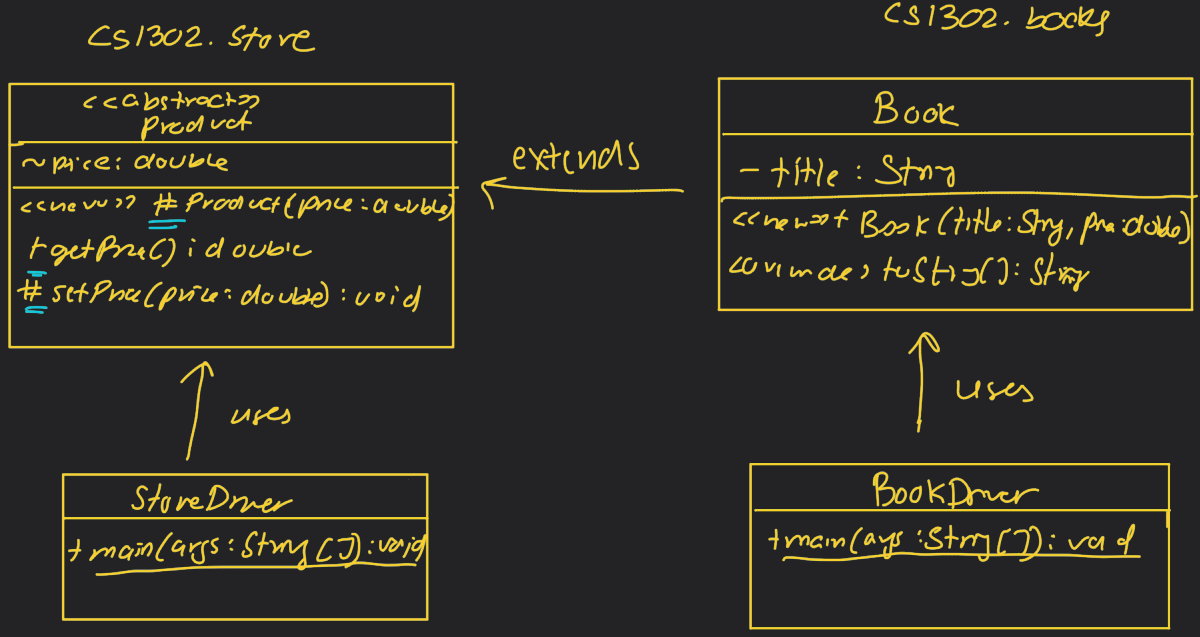
→ They can only access requestChange method because it's public. Other 2 are package private & here the package where the methods are located and the package from where they are attempted to be accessed are different

# Protected

→ Only member-level declarations are allowed to be protected
→ visible only from lines of code in the same package or a child class regardless of its package

* Same class ✓
* Same Package ✓
* Child class ✓
* Elsewhere ✗

- Protected
- #
- javadoc includes protected declar. in website by default




APPROACH 1

```
// inside Book.java (cs1302.books)
Public Book(String title, double price){
    super(price);
    this.title = title;
}
```

APPROACH 2

```
// inside Book.java (cs1302.books)
public Book(String title, double price){
    setPrice(price);
    this.title = title;
}
```

```
// inside BookDriver.java
public static void main(...){
  Book lotr = new Book("Lord...", 11.99);
  lotr.setPrice(lotr.getPrice() * 0.8);
}
```

**Line 1**
Product(price) — protected — In Product — from Book — Visible from child — Not in same pkg

**Line 2**
setPrice(price) — protected — In Product — from Book — Visible from child — Not in same pkg

(although in diff packages, a child class is trying to access it so it works)

**Line 3**
setPrice(price) — protected — In Product — from BookDriver — NOT visible from same pkg or from child