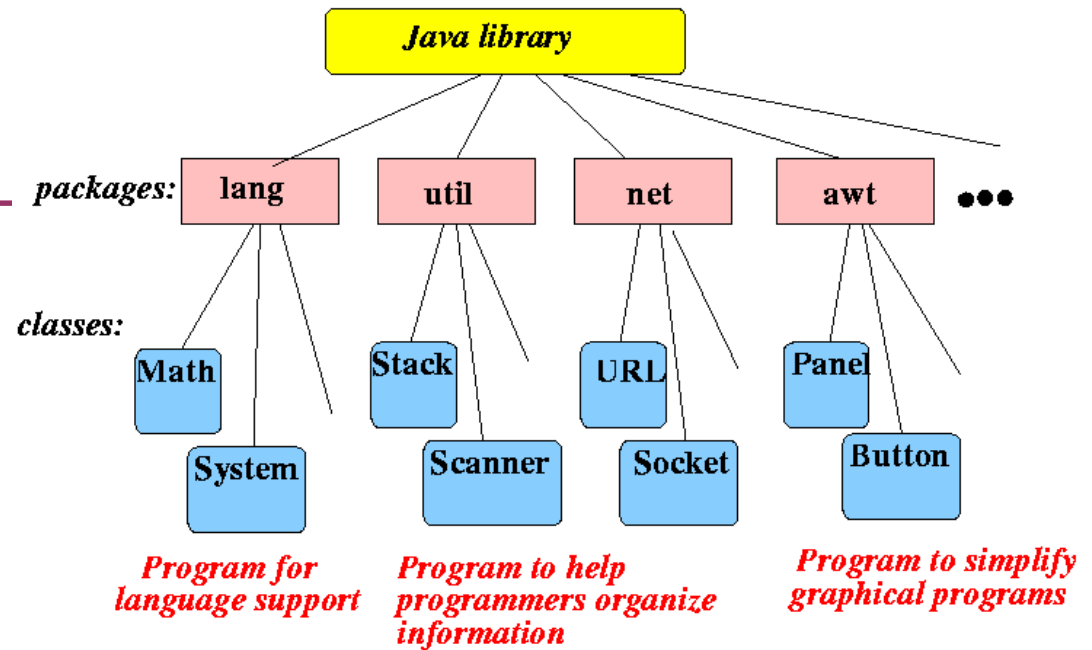


---

# **Packages, Visibility (review?)**

Maria Zontak

# Packages



What?

**A package** – a group of related classes, interfaces and sub-packages

Responsibilities?

- **Provides organization** based on related functionality
- **Provides access protection** - hides classes required for package implementation, but that should not be used by client code
- **Removes naming collisions** - a package defines a **namespace**

Can reuse common type names in different packages (List, ...)

# Package and Type Names

---

- Every class and interface has a *fully qualified* name:

package name + type name

`java.awt.Color`

`java.lang.String`

`java.awt.Rectangle`

- Can always refer to a type using its fully qualified name

```
java.lang.String a = new java.lang.String();
```

- Each type also has a *simple name*

`Color, ArrayList, Rectangle`

- Can use `import` declarations to refer to type by *simple name*
- Why don't we import `java.lang.String`, but still use `String` by a simple name?

# Import Declarations (1)

---

To import

- a single type - provide its fully qualified name

```
import java.awt.Color;
```

- all types in a package – use the package name and ‘\*’

```
import java.util.*;
```

- Have to import each package explicitly

```
import java.*; // does NOT import java.util.*,
```

must do

```
import java.util.*;
```

# Some Standard Packages

The standard Java libraries contain over 3000 classes and interfaces in over 150 packages.

Common examples:

- `java.lang` – core classes;

**imported automatically**

includes Math, Integer, Double, Char..

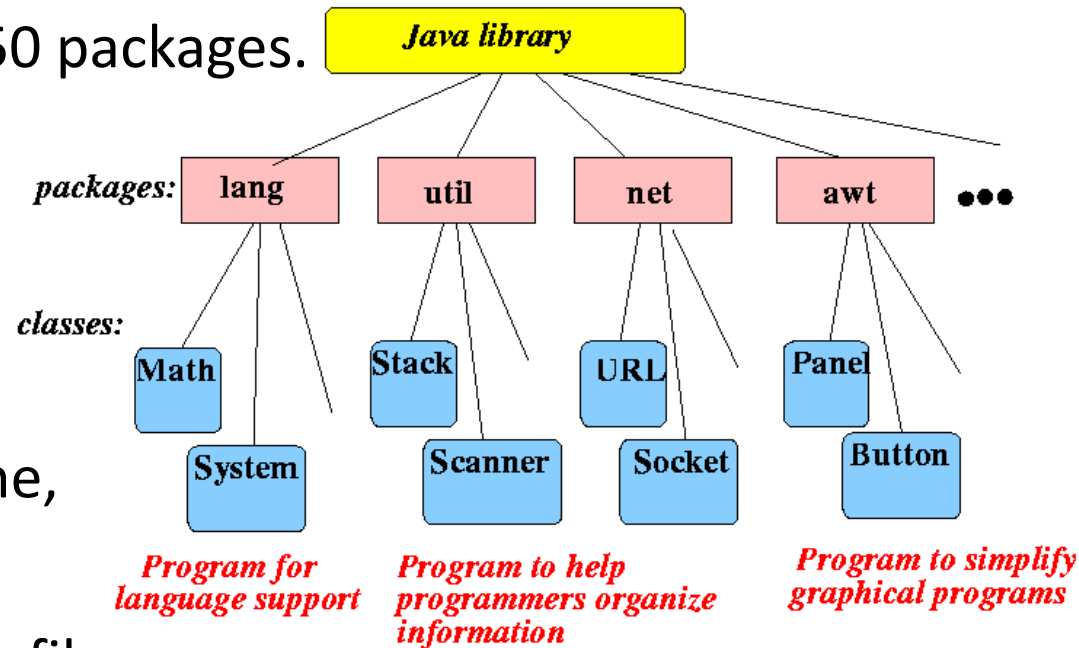
- `java.util` – collections, date/time, random number generator, ...

- `java.io` – input/output streams, files

- `java.net` – network I/O, sockets, URLs

- `java.awt` – original graphical user interface (GUI)

- `javax.swing` – extension of awt, more sophisticated GUI



# Visibility of Fields and Methods

---

- Four possibilities
  - **private** – visible only in the class containing the declaration
  - default (no keyword) – visible in the declaring class and in all other classes in the same package  
(this is the default if nothing specified; no keyword !)
  - **protected** – like package, but also visible in any class that extends this class, even if in another package
  - **public** – visible anywhere the class is visible
- Corollary: if you forget to specify private, it is visible outside the class within the package. **Careful!**

# Guidelines

---

- **Instance variables** should almost always be private
  - Provide appropriate methods to give client code controlled access (if needed)
  - Perhaps protected if the class is intended to be extended and we do not want to make getters methods public: **Consider carefully**
- **Methods** should be public if part of the published interface of a class; private otherwise
- Protected and package visibility only after careful consideration; not by default!

# Visibility of Classes

---


- **Public**

visible anywhere the package is visible

- **Package – protected**

visible only to the code in the same package.

```
public class Example { ... } // public class name
```

```
 class anotherExample{ ... } // class with  
package scope
```

- A Java source file contains only ONE public class or interface, and filename must match the public name