CODING SHUTTLE

OOPS - 3

In This Lecture

1. Java Packages ✓
2. Access modifiers ✓
3. Java Encapsulation ✓
4. Data Hiding ✓
5. The static keyword ✓

Java Packages

A package is simply a container that groups related types (Java classes, interfaces, enumerations, and annotations).

To define a package in Java, you use the keyword package.

✓ Java uses file system directories to store packages.

For example:

```
└── com
    └── test
        └── Test.java  ✓
```

Test.java

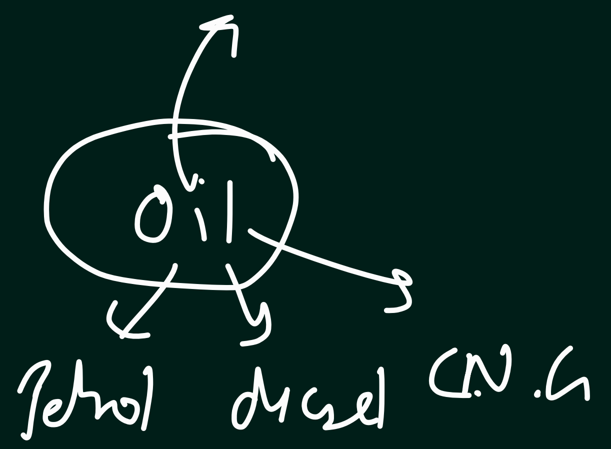
```
→ package com.test;

→ class Test {
      public static void main(String[] args){
          System.out.println("Hello World!");
      }
  }
```

Importing a Package

- Java has an import statement that allows you to import an entire package (as in earlier examples), or use only certain classes and interfaces defined in the package.

```
import java.util.Date; // imports only Date class
import java.io.*;      // imports everything inside java.io package
```



- In Java, the import statement is written directly after the package statement (if it exists) and before the class definition.

```
package package.name;
import package.ClassName; // only import a Class

class MyClass {
    // body
}
```

Java Access Modifiers

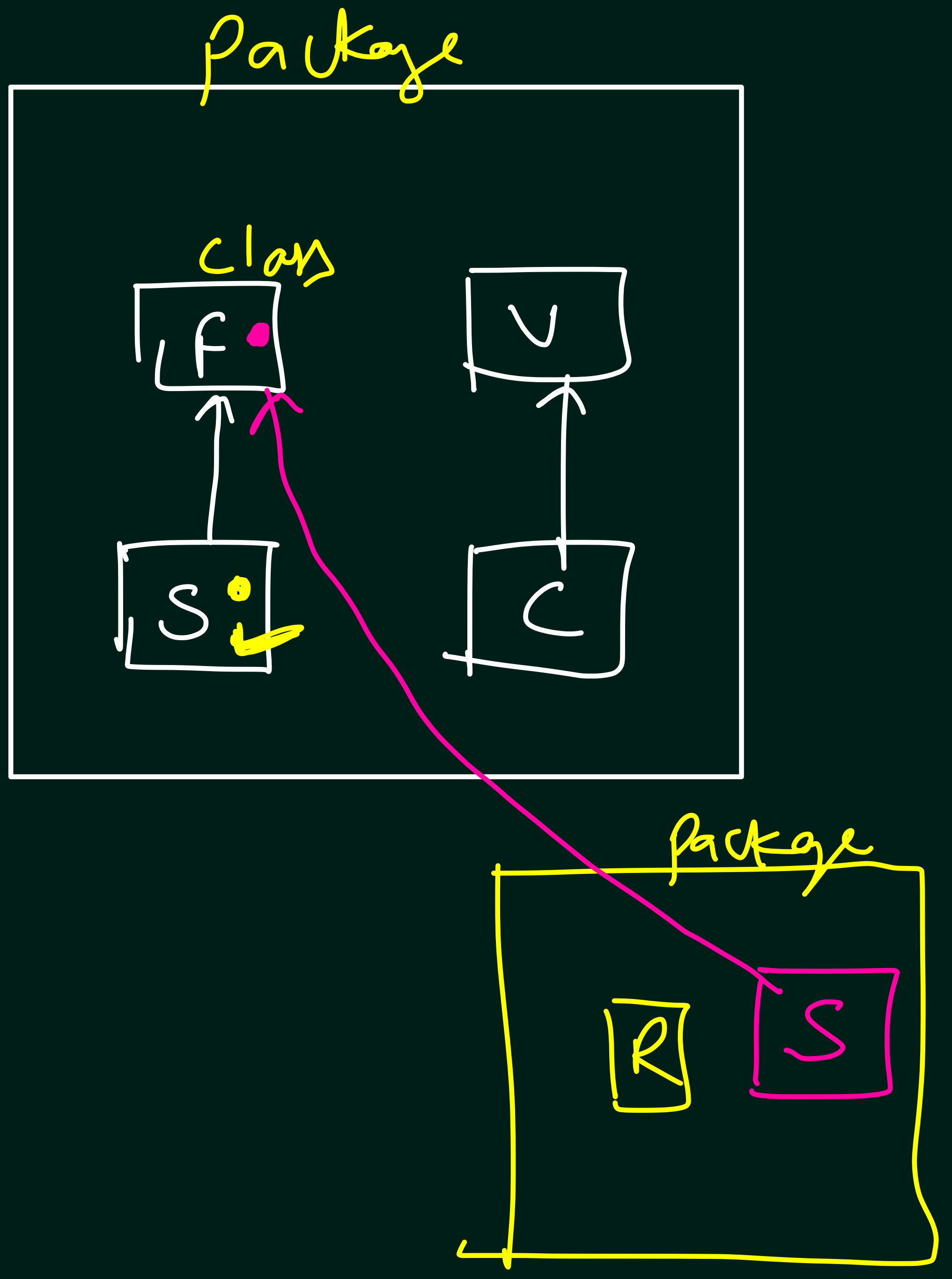
In Java, access modifiers are used to set the accessibility (visibility) of classes, interfaces, variables, methods, constructors, data members, and the setter methods. For example,

```
class Animal {
    public void method1() {...}
      ^
   private void method2() {...}
}     ^
```

Types of Access Modifiers

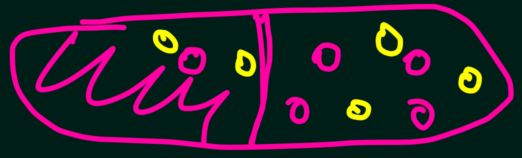
Package Private

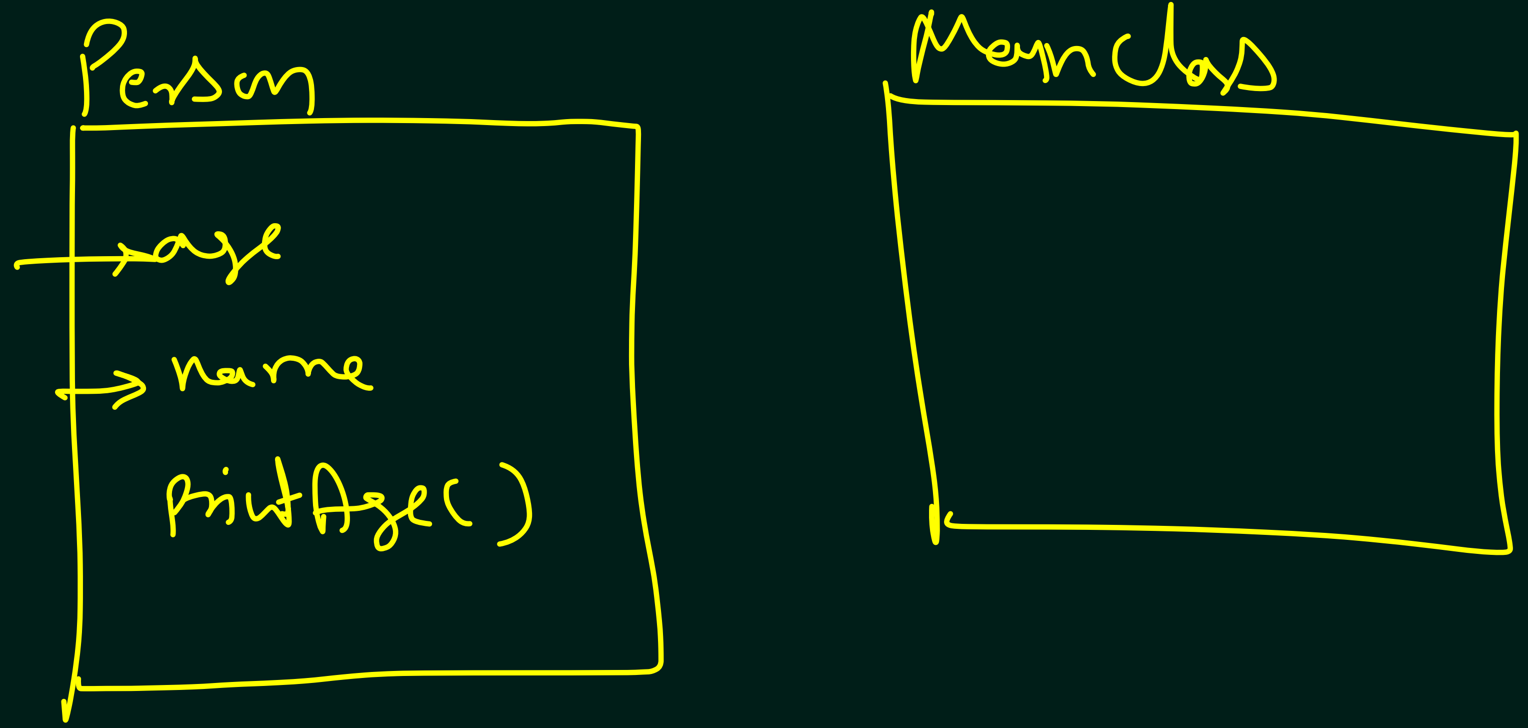
Access Modifier	Same class	Same package subclass	Same package non-subclass	Difference Package subclass	Different package non-subclass
Default	Yes	Yes	Yes	No	No
✓ Private	Yes	No	No	No	No
✓ Protected	Yes	Yes	Yes	✓ Yes	No
✓ Public	Yes	Yes	Yes	Yes	Yes



Java Encapsulation

Encapsulation refers to the bundling of fields and methods inside a single class. It prevents outer classes from accessing and changing fields and methods of a class. This also helps to achieve data hiding.



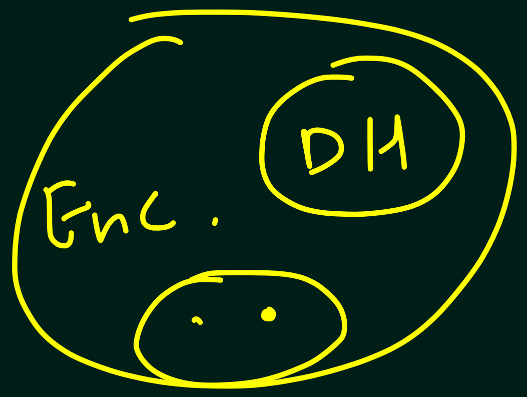


Data Hiding

Data hiding is a way of restricting the access of our data members by hiding the implementation details. Encapsulation also provides a way for data hiding.

We can use access modifiers to achieve data hiding.

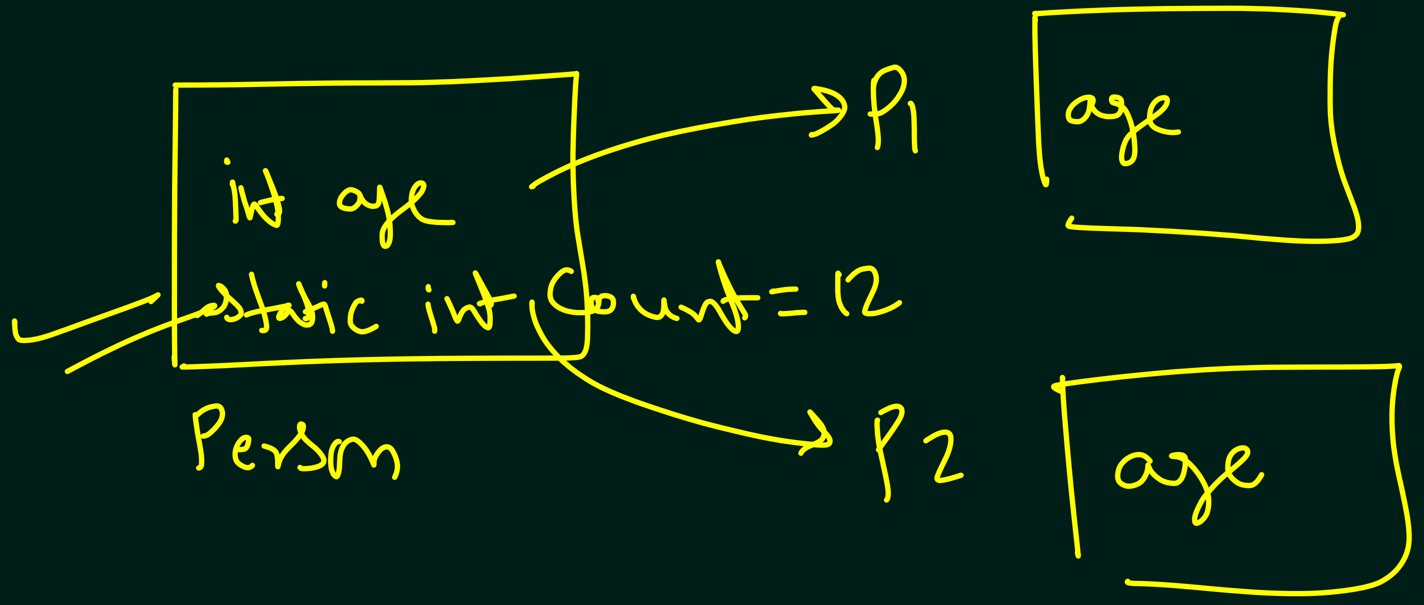
Note: People often consider encapsulation as data hiding, but that's not entirely true. Encapsulation refers to the bundling of related fields and methods together. This can be used to achieve data hiding. Encapsulation in itself is not data hiding.



The static keyword

If we want to access class members without creating an instance of the class, we need to declare the class members static.

Static variables can be accessed by calling the class name of the class. There is no need to create an instance of the class for accessing the static variables because static variables are the class variables and are shared among all the class instances.



The static keyword

Static Variables

- Only a single copy of the static variable is created and shared among all the instances of the class.
- Because it is a class-level variable, memory allocation of such variables only happens once when the class is loaded in the memory.
- If an object modifies the value of a static variable, the change is reflected across all objects.
- Static variables can be used in any type of method: static or non-static.
- Non-static variables cannot be used inside static methods. It will throw a compile-time error.

The static keyword

Static Methods

- The static members and methods belong to the class rather than the instance of the class. When the implementation of the particular method is not dependent on the instance variables and instance methods, In this case, we can make that method to be static.
- They are accessed by the name of the class.
- The keywords such as this and super are not used in the body of the static method.
- The modification of the static field value is not allowed.