OBJECT ORIENTED PROGRAMMING WEEK-2

JAN 30- FEB 03, 2023

Instructor:

Abdul Aziz
Assistant Professor
(Software Engg. Department)
National University- FAST (KHI Campus)

ACKNOWLEDGMENT

- Publish material by Virtual University of Pakistan.

- Publish material by Deitel & Deitel.

- Publish material by Robert Lafore.

OBJECT

Technical Definition:

- “An Object is an Instance of a class”

- “An Object is the implementation of a class”

In general we say :

“ Any tangible thing for which we want to save Information”

Now onwards we treat object technically.

SOME EXAMPLES

Men sandal

Female sandal

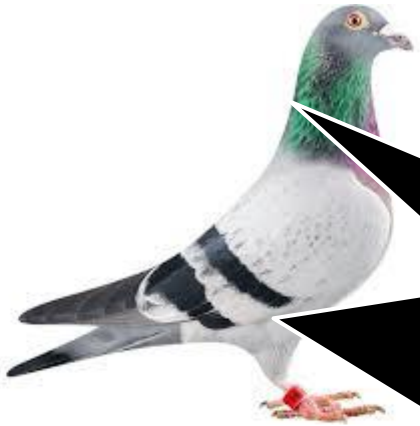
Loafers

Pumpies

Formal

Humans

FISH



BIRD

Animal

DISCUSSION

- Object belongs to a group.
- Which similar.
- Have some common attributes.
- Have some common behaviors.
- We can categorized objects on some basic features …. ?

CLASS

- Collection of Similar object.

- The objects that share some common features.

- It is the a design of an object.

- It is a detail of an object.

- It tell us what an object contains in it.

Technical Definition:

“ A class is blueprint of an object”

SUMMARIZE

A class :

- It’s a blue print .
- It’s a design or template.

An Object:

- Its an instance of a class.
- Implementation of a class.

NOTE: Classes are invisible, object are visible

GENERALIZED CLASS

- The class that only exhibits the common features of its objects.

Examples:

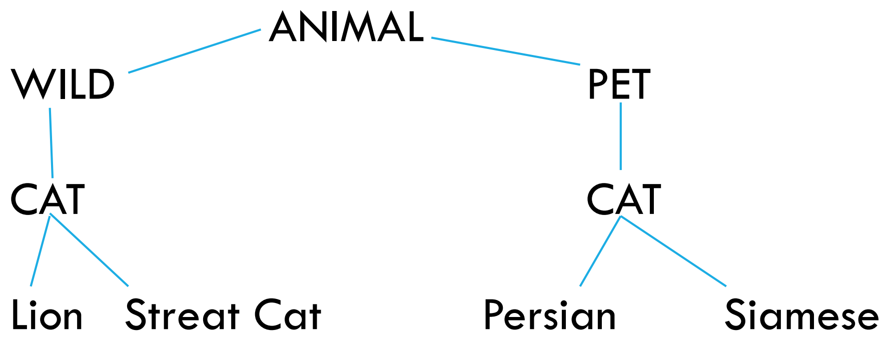
- ANIMAL
- BIRDS
- HUMAN
- No object of generalized class is found.

SPECIALIZED CLASS

- The class that exhibits different or unique features (behaviors)

ANIMAL (Generalized)

- Specialized:
  - Mammals
  - Cats
  - Dog



TYPE OF CLASSES

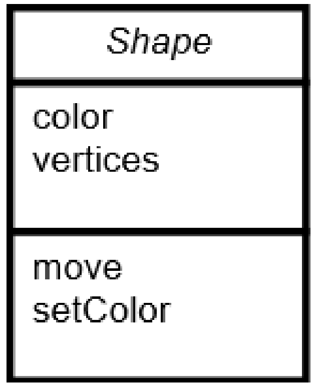
1- Abstract Class: The classes we make against abstract concepts are called abstract classes. Abstract Classes can not exist standalone.

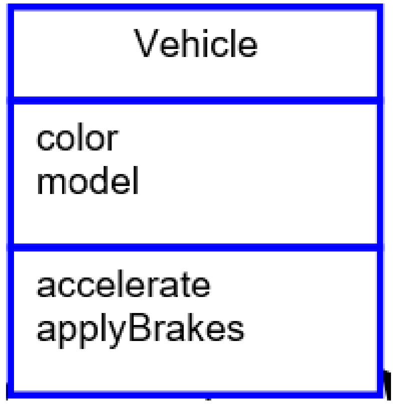
2- Concrete Class: The entities that actually we see in our real world are called concrete objects and classes made against these objects are called concrete classes.

3- Sub-type: Sub-typing means that derived class is behaviorally compatible with the base class. Also known as Extension.

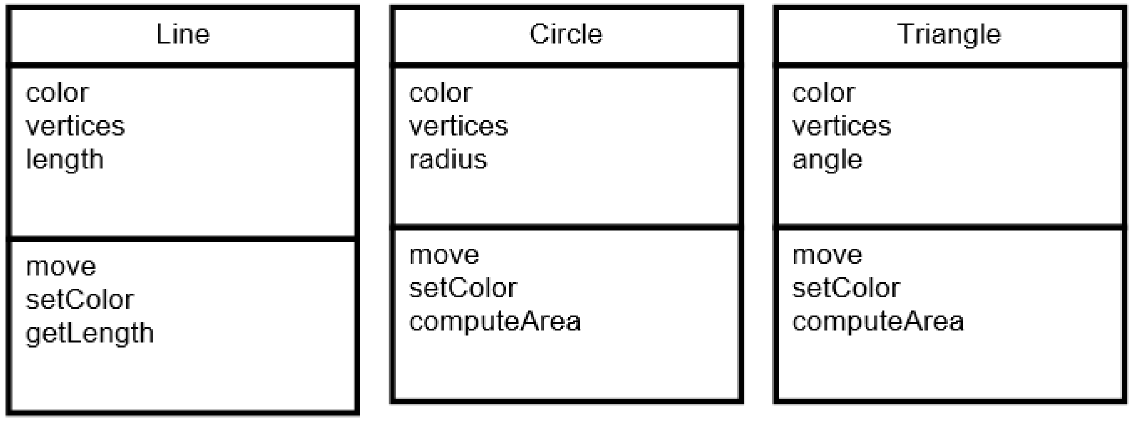
4- Specialized class: Specialization means that derived class is behaviorally incompatible with the base class

ABSTRACT CLASS





CONCRETE CLASS



Line is shape Circle is a shape Triangle is a shape

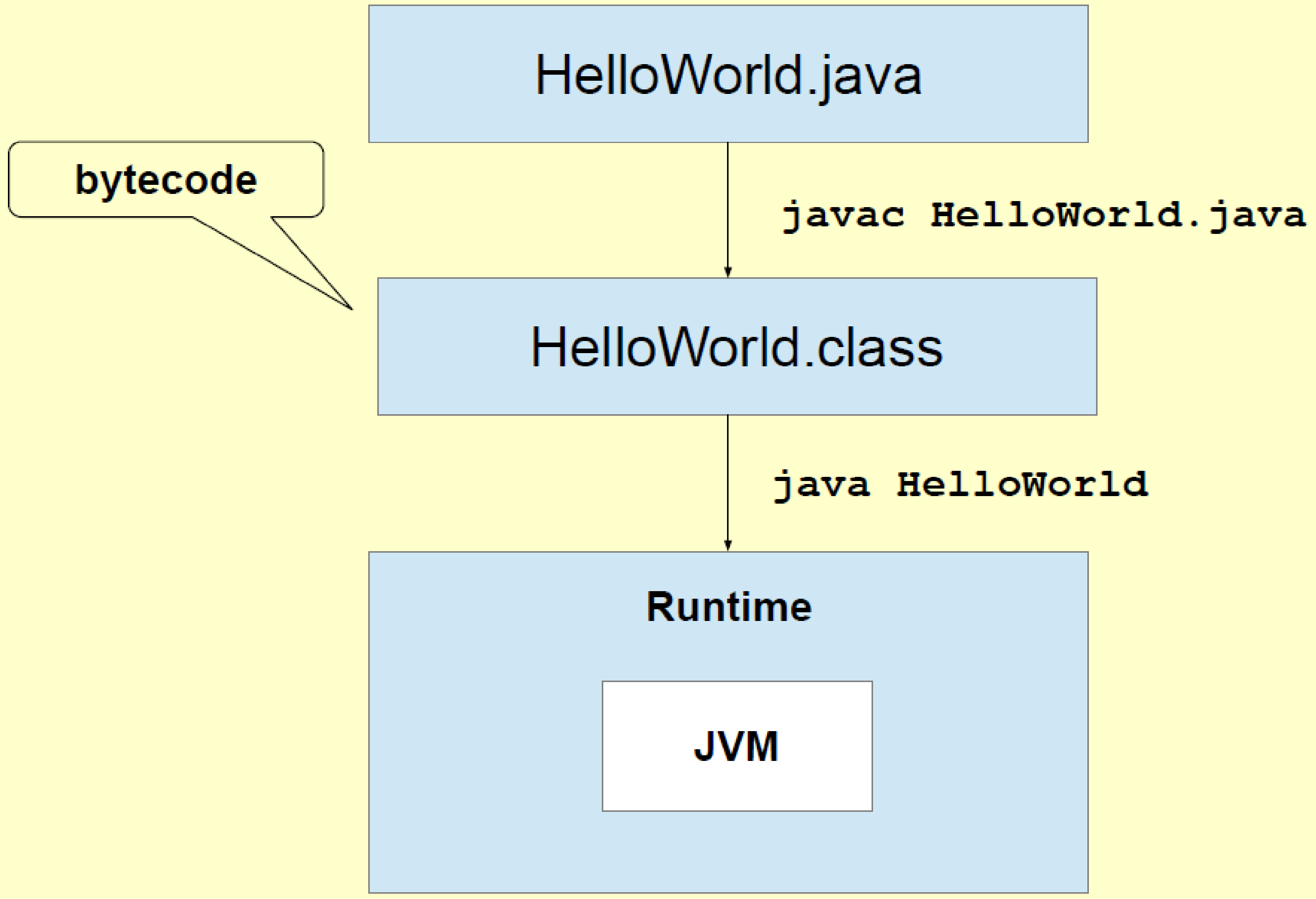
Hello World Application

1. Write the source code: HelloWorld.java

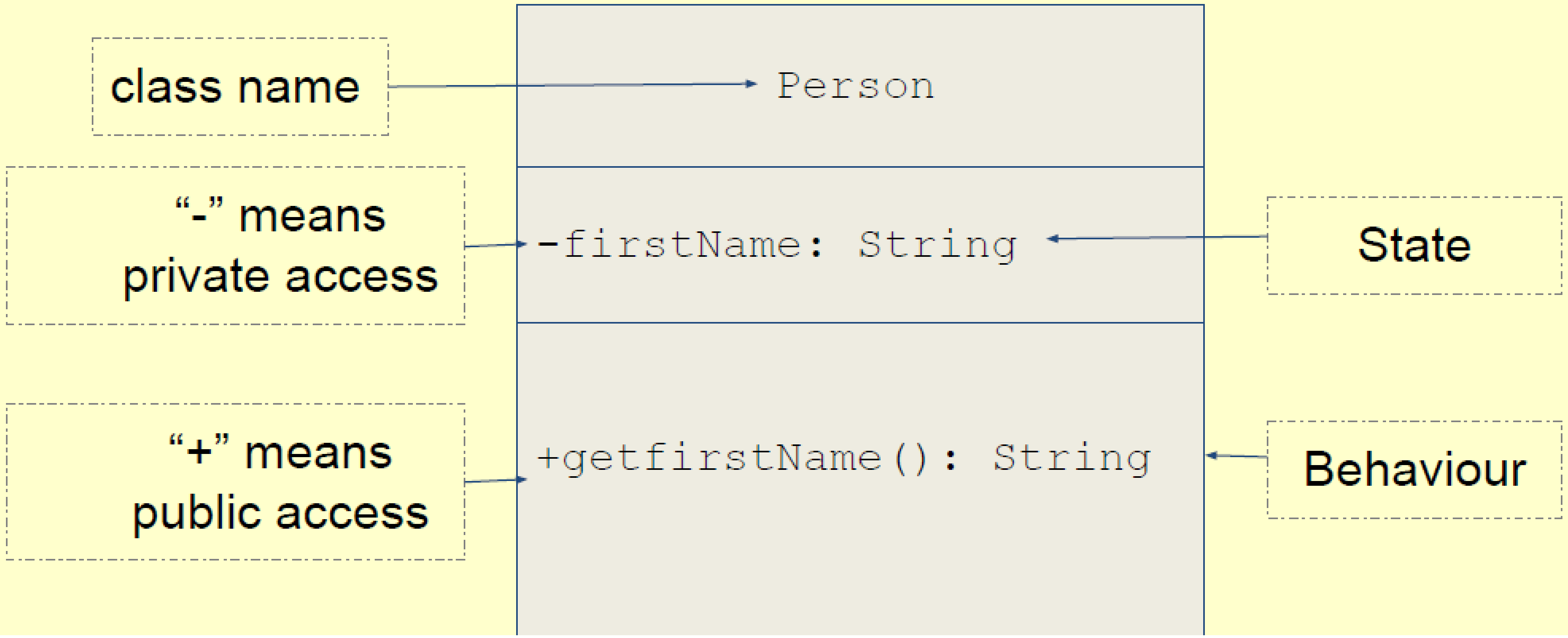
```
public class HelloWorld{
    public static void main( String args[] ){
        System.out.println("Hello world");
    }
 }
```

2. Compile: `javac HelloWorld.java`
3. Run: `java HelloWorld`

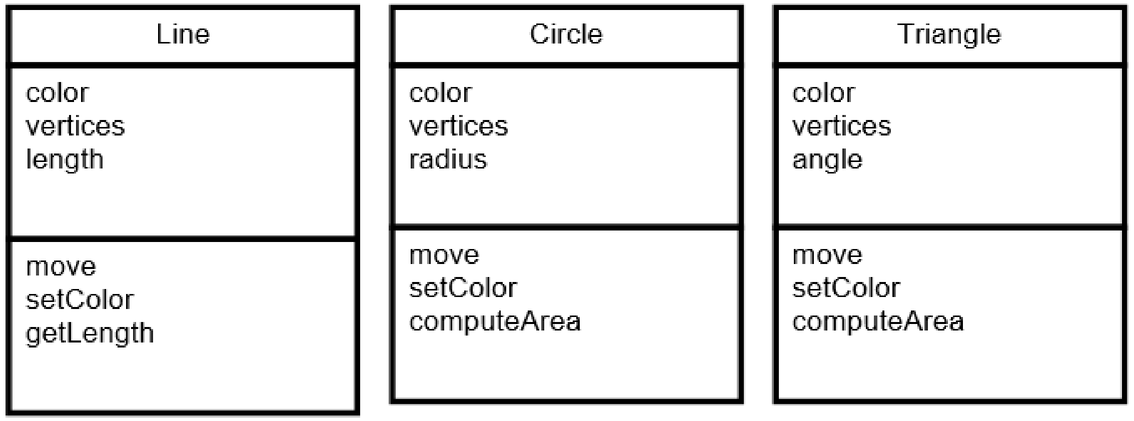
Hello World Application



UML - Graphical Class Representation



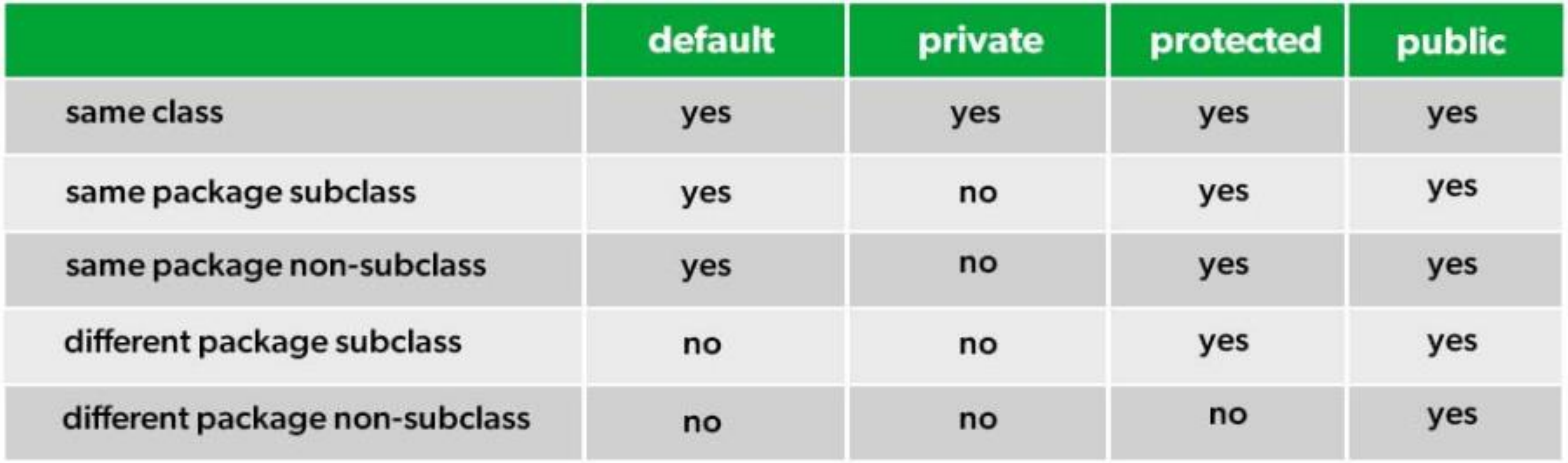
EXAMPLE OF CLASS DIAGRAM



Line is shape   Circle is a shape   Triangle is a shape

ACCESS MODIFIERS

- **public:** The member can be accessed by any other code **(Everyone)**.
- **private:** The member can only be accessed within the class OR by object itself only **(only Me)**.
- **protected:** The member is accessed by the class, sub-class, non-sub class even in package. **(Me & my Family)**
- **default:** It is also referred to as no modifier. Whenever we do not use any access modifier it is treated as default where this allows us to access within a class, within a subclass, and also non-sun class within a package but when the package differs now be it a subclass or non-class we are not able to access.



	default	private	protected	public
same class	yes	yes	yes	yes
same package subclass	yes	no	yes	yes
same package non-subclass	yes	no	yes	yes
different package subclass	no	no	yes	yes
different package non-subclass	no	no	no	yes

Class

- Is a **user-defined** type
  - Describes the *data* **(attributes)**
  - Defines the *behavior* (**methods**)

  } **members**

- Instances of a class are **objects**

Declaring Classes

- **Syntax**

```
<modifier>* class <class_name>{
  <attribute_declaration>*
  <constructor_declaration>*
  <method_declaration>*
}
```

- **Example**

```
public class Counter{
   private int value;
   public void inc(){
      ++value;
   }
   public int getValue(){
      return value;
   }
}
```

Declaring Attributes

- **Syntax**

```
<modifier>* <type> <attribute_name>[= <initial_value>];
```

- **Examples**

```
public class Foo{
   private int x;
   private float f = 0.0;
   private String name =”Anonymous”;
}
```

Declaring Methods

- **Syntax**

```
<modifier>* <return_type> <method_name>( <argument>* ){
    <statement>*
}
```

- **Examples**

```
public class Counter{
   public static final int MAX = 100;
   private int value;

   public void inc(){
      if( value < MAX ){
         ++value;
      }
   }
   public int getValue(){
      return value;
   }
}
```

Accessing Object Members

- Syntax

  `<object>.<member>`

- Examples

```
Counter c =  new   Counter();
c.inc();
int i = c.getValue();
```

CODED EXAMPLE

- Class
- Save class with same name
- Data and functions
- Setter & Getter
- Access Modifiers