Object-Oriented Modeling Using Unified Modeling Language (UML):

The **Unified Modeling Language** (**UML**) is a general-purpose modeling language in the field of software engineering. The basic level provides a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. Higher levels cover process-oriented views of a system.

What is a Model?

A model is a simplification of reality.

A successful software organization is one that consistently deploys quality software that meets the needs of its users. An organization that can develop such software in a timely and predictable fashion, with an efficient and effective use of resources, both human and material, is one that has supportable business."
We build models so that we can see and better understand the system we are developing.

- Two most common ways in modeling software systems are
  - Algorithmic
    - Procedures or functions

- Object oriented
  - Objects or classes

- UML is a language for
  - Visualizing
  - Specifying
  - Constructing
  - Documenting

	Interpretation in the Real World	**Interpretation in the Model**
Object	An object is a thing that can be clearly identified.	An object has an identity, a state, and a behavior.
Class	A class represents a set of objects with similar characteristics and behavior. This objects are called the instances of the class.	A class characterizes the structure of states , and behaviors that are shared by all instances.

- Each of object has a unique identity.
- The state of an object is composed of a set of fields (data fields), or attributes.
- Each field has a name, a type, and a value.
- Behaviors are defined by methods.
- Each method has a name, a type, and a value.
- Each method may or may not return a value.
- Features are a combination of the state and the behavior of the object

- A class defines a template for creating its instances or objects.
- A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics
- A class defines-- the names and types of all fields the names, types, implementations of all methods

- The values of the fields are not defined or fixed in the class definition.
- The values of the fields are mutable.
- Each instance of the class has its own state.
- Different instances of the class may have different states.

Example

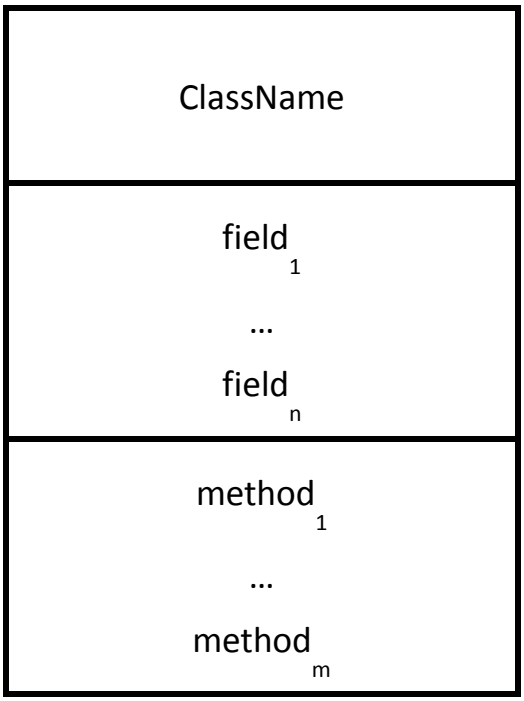
Class name: `Point class Point {`

Fields: `x, y  int x, y;`

Method: `move  public void move`

```
(int dx, int dy){
// implementation
}
```

UML Notation for Classes:



The top compartment shows the class name.

The middle compartment contains the declarations of the fields of the class.

The bottom compartment contains the declarations of the methods

Field Declaration:

- The name of the field is required in the field declaration.
- Field declaration may include:
-

`[Visibility][Type]Name =[Initial Value]`

- Visibility or accessibility defines the scope:
  - Public -- the feature is accessible to any class

- Protected -- the feature is accessible to the class itself, all the classes in the same package, and all its subclasses.
- Package -- the feature is accessible to the class itself and all classes in the same package.
- Private -- the feature is only accessible within the class itself.
-

Visibility syntax in Java and UML:

Visibilty	Java Syntax	UML Syntax
public	public	+
protected	protected	#
package		~
private	private	-

Examples:

Java Syntax	UML Syntax
Date birthday	Birthday:Date
Public int duration = 100	+duration:int = 100
Private Student students[0..MAX_Size]	-students[0..MAX_Size]:Student

Method Declaration:

- The name of the method is required in the method declaration.
- Method declaration may include:

`[`*`Visibility`*`][`*`Type`*`]`*`Name`*`([`*`Parameter`*`, ...])`

`[`*`Visibility`*`]`*`Name`*`([`*`Parameter`*`, ...])[:`*`Type`*`]`

- Each parameter of a method can be specified by -- *`Type Name`*

Java Syntax	UML Syntax
`void move(int dx, int dy)`	`~move(int dx, int dy)`
`public int getSize()`	`+ getSize():int`

Point
`private int x` `private int y`
`public void move(int dx,int dy)`

Point
`-x:int` `-y:int`
`+move(dx:int,dy:int)`

UML Notation for Object

ObjectName : ClassName	The top compartment shows the object name and its class.
$field_1 = value_1$ … $field_n = value_n$	The bottom compartment contains a list of the fields and their values.

objectName -- objectName whose class is of no interest

:ClassName -- anonymous object of ClassName which can be identify only through its relationship with other object.

Examples

P1:Point	Point p1 = new Point();
x = 0 y = 0	p1.x = 0; P1.y = 0;

P1:Point	Point p1 = new Point();
x = 24 y = 40	p1.x = 24; P1.y = 40;