Object-Oriented Software Development

- Object-Oriented models are composed of **objects**.
- Objects contain data and make computations.
- The decomposition of a complex system is based on the structure of classes, objects, and the relationship among them. (divide-and-conquer).
- When we divide our problems into sub-problems, we will try to design classes to solve these sub-problems.
- We will use graphical notation to describe object-oriented analysis and design models. This notation is based on *Unified Language Modelling (UML)*.

Classes and Objects

- **Objects** and **classes** are two fundamental concepts in the object-oriented software development.
- An *object* has a unique *identity*, a *state*, and *behaviors*. In the real life, an object is anything that can be distinctly identified.
- A *class* characterizes the structure of states and behaviors that shared by all its instances.
- The terms *object* and *instance* are often interchangeable.
- The *features* of an object is the combination of the *state* and *behaviors* of that object.
  - The state of an object is composed of a set of *attributes* (fields) and their current values.
  - The behavior of an object is defined by a set of *methods* (operations, functions, procedures).
- A class is a template for its instances. Instead of defining the features of objects, we define features of the classes to which these objects belong.

Classes in Java

- A class in Java can be defined as follows:

```
class Rectangle {
    int length, width;
    public int area() {……}
    public void changeSizes(int x, int y) { …… }
```

- The name of the class is: Rectangle
- Its attributes are: length width
- Its methods are: area changeSizes
- This Rectangle class is a template for all rectangle objects. All instances of this class will have same structure.