# Object Oriented Programming

OOP Concept: Class, Object, Encapsulation

## A C Program to Calculate Rectangle Area

```c
#include <stdio.h>

double calculateRectangleArea(double width, double height) {
    return width * height;
}

int main() {
    double width;
    double height;

    width = 5;
    height = 7;
    printf ("Area = %.2lf\n", calculateRectangleArea(width, height));

    return 0;
}
```

## A C Program to Calculate Rectangle Area

```
int main() {
    double width[5];
    double height[5];

    width[0] = 5;
    height[0] = 7;
    printf("Area = %.2lf\n", calculateRectangleArea(width[0], height[0]));

    width[1] = 5;
    height[1] = 7;
    printf("Area = %.2lf\n", calculateRectangleArea(width[1], height[1]));

    return 0;
}
```

In procedual programming languages we break down the program into a collection of variables, data structures and subroutines. In this lecture, we will explore other paradiam called object-oriented programming which is widely used nowadays and is supported by most modern programming languages.

## Class / Object Definitions

- **Class** acts as a blueprint that describe the type of data and methods (actions) that its object will have
- **Object** is an instance of a class. Every objects of the same class has same type of data and methods, but they maintain their own state i.e., the value of each data item
- Data items are sometimes called fields or instance variables
- An object-oriented program consists of objects from various classes interacting with one another

## Example 1: A Simple Rectangle Class in Java

```
class Rectangle {
    double width;
    double height;

    double getArea() {
        return width * height;
    }

    double getPerimeter() {
        return width * 2 + height * 2;
    }
}
```

Instance variables

Methods

- Instance variables are declared in the same way as a variable declaration
- Methods define actions that the object of this class can perform. Method may use the instance variables to perform its function. It can take arbitrarily number of parameter and can return a value similar to function in C.

# Object Instantiation / Usage (I)

```java
class Main {
    public static void main(String[] args) {
        Rectangle rect1 = new Rectangle();
        rect1.width = 5;
        rect1.height = 7;

        double area = rect1.getArea();
        System.out.println(area);
    }
}
```

- In Java, everything must be in a class even the main method!!!
- The `new` operator is used to create an object of a class
- The variable `rect1` is a variable with the type Rectangle. It stores a reference to the object created by the new operator
- The instance variables and methods can be refered to by the object using the `.` operator follow by its name

## Object Instantiation / Usage (II)

```
class Main {
    public static void main(String[] args) {
        Rectangle rect1 = new Rectangle();
        rect1.width = 5;
        rect1.height = 7;
        System.out.println(rect1.getArea());

        Rectangle rect2 = new Rectangle();
        rect2.width = 10;
        rect2.height = 30;
        System.out.println(rect2.getArea());
    }
}
```

- `rect1` and `rect2` is a separate object of the class Rectangle thus has its own state
- System is a class in Java Class Library. out is a static instance variable of type PrintStream which contain the method println. Static instance variables are shared among all object of a class and are referred to by the class name

## Constructor/Destructor

```
class Rectangle {
    double width;
    double height;

    Rectangle() {}

    Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }
    // methods
}
```

No-Argument Constructor (create automatically if there isn't any other constructor defined in the class)

Constructor with 2 parameters which is used to initialize the instance variables

- Constructor is used for initialized the object at the creation time
- It is a method with the same name as the class (in Java and most programming languages)
- Some programming languages contain a destructor which is invoked when the object is destroyed. In Java, most object is reclaimed automatically by the garbage collector and resource is close using the explicit close method.
- The expression `new Rectangle()` invoke a constructor to create a new object

## OOP Principles

1. **Encapsulation:** hiding the internal implementation and provide a well-defined public interface so that the object is safe from being misuse and to facilitates code refactoring and decoupling
2. **Inheritance:** promote code reused by inherited data and method from the parent class. Inheritance allows classes to be arranged in a hierarchy that represents is-a-type-of relationships
3. **Polymorphism:** allow objects of the parent class and the derived class to be used interchangeably to allow the operation to be performed on object from class in the same hierarchy

## Java's Encapsulation

In Java, we hide the information using the access modifier and provide accessor/mutator methods

List of Java's access modifier

| Modifier | Class | Package | Subclass | World |
|---|---|---|---|---|
| public | Y | Y | Y | Y |
| protected | Y | Y | Y | N |
| no modifier (package-private) | Y | Y | N | N |
| private | Y | N | N | N |

# Example 2: A Better Rectangle Class in Java

```
public class Rectangle {
    private double width;
    private double height;

    public Rectangle() {}

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }

    public double getWidth() { return width; }
    public void setWidth(double width) { this.width = width; }

    public double getHeight() { return height; }
    public void setHeight(double height) { this.height = height; }

    public double getArea() { return width * height; }
    public double getPerimeter() { return width * 2 + height * 2; }
}
```

Instance variables are usually declared as private

Public accessor/mutator methods

## Accessor and Mutator Methods

```java
public class Rectangle {
    private double width;
    private double height;

    public double getWidth() {
        return width;
    }

    public double getArea() {
        return width * height;
    }

    public boolean setWidth(double width) {
        if (width < 0) {
            return false;
        }
        this.width = width;
        return true;
    }
}
```

Accessor method can return field with different name and type or even compute the value when it is being called

- Mutator method can be used to validate input to meet the condition required
- We can also omit all mutators in an immutable class

## Inheritance Concept

- Inheritance allows a new class to be created by extend from another class, called the base class.
- The new (derived) class has all the instance variables and methods of the base class.
- The new (derived) class can define additional methods/instance variables and/or override the method with new a definition
- In Java and most programming languages, a class can only be derived from a single base class. C++ and Python are among a few language to support multiple inheritance while other languages prefer interface/protocol to achieve similar result.

## Example 2: A Square Class

```java
public class Square extends Rectangle {
    public Square() { super(); }
    public Square(double length) {
        super(length, length);
    }

    @Override
    public void setWidth(double width) {
        super.setWidth(width);
        super.setHeight(width);
    }

    @Override
    public void setHeight(double height) {
        super.setWidth(height);
        super.setHeight(height);
    }
}
```

The extends keyword is used to specify the base class

super() is used to refer to the base class constructor

@Override is an optional annotation used by a compiler to generate warning when the method is not actually overriding any base class method

super is used to refer to the base class instance variables / methods while this is used to refer to our own instance variables / methods

## Example 2: A Square Class

```
class Main {
    public static void main(String[] args) {
        Rectangle rect1 = new Rectangle(5, 7);
        System.out.println(rect1.getArea());
        System.out.println(rect1.getPerimeter());

        Square square1 = new Square(5);
        System.out.println(square1.getArea());
        System.out.println(square1.getPerimeter());
    }
}
```

- Constructors are not automatically inherited thus the 2 parameters constructor from Rectangle is not available
- The `getArea()` and `getPerimeter()` is inherited from the Rectangle class thus we don't have to redefine them

## Interface

- Interface specifies a set of method that any class that implements the interface must have
- Interface is used extensively in Java for example
  - We use interface as a variable type and type of method parameters so that they can accept object from any class implement the interface e.g. all methods that accept a List can work with instance of the ArrayList and LinkedList class
  - The sort function can work with object of any class that implement the Comparable interface

# A Simple Shape Interface

```java
public interface Shape {
    double getArea();
    double getPerimeter();
}

public class Rectangle implement Shape {
    // instance variables and methods
}

class Main {
    public static void main(String[] args) {
        Shape shape1 = new Rectangle(5, 7);
        System.out.println(shape1.getArea());
        System.out.println(shape1.getPerimeter());

        Shape shape2 = new Square(5);
        System.out.println(shape2.getArea());
        System.out.println(shape2.getPerimeter());
    }
}
```

Interface definition consists of the method declaration without its implementation

The implement keyword is used to implement the interface. Class can implement multiple interfaces by listing them separate by a comma

## Lab 1

**Objective:** This lab give you a chance to write a simple object-oriented software in Java and to pratice programming language document reading skill.

**Problem:** Write a program to calculate area and perimeter of a rectangle, triangle, square or circle by asking the type of shape and its dimension from the user.

Some useful online resource:

- https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/util/Scanner.html
- https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/lang/String.html