



# Encapsulation and Abstraction

# Agenda

1

## **Encapsulation and Abstraction**

# Objectives

At the end of this module, you will be able to:

- Understand the relevance of Object Oriented Programming techniques
- Implement Encapsulation and Abstraction

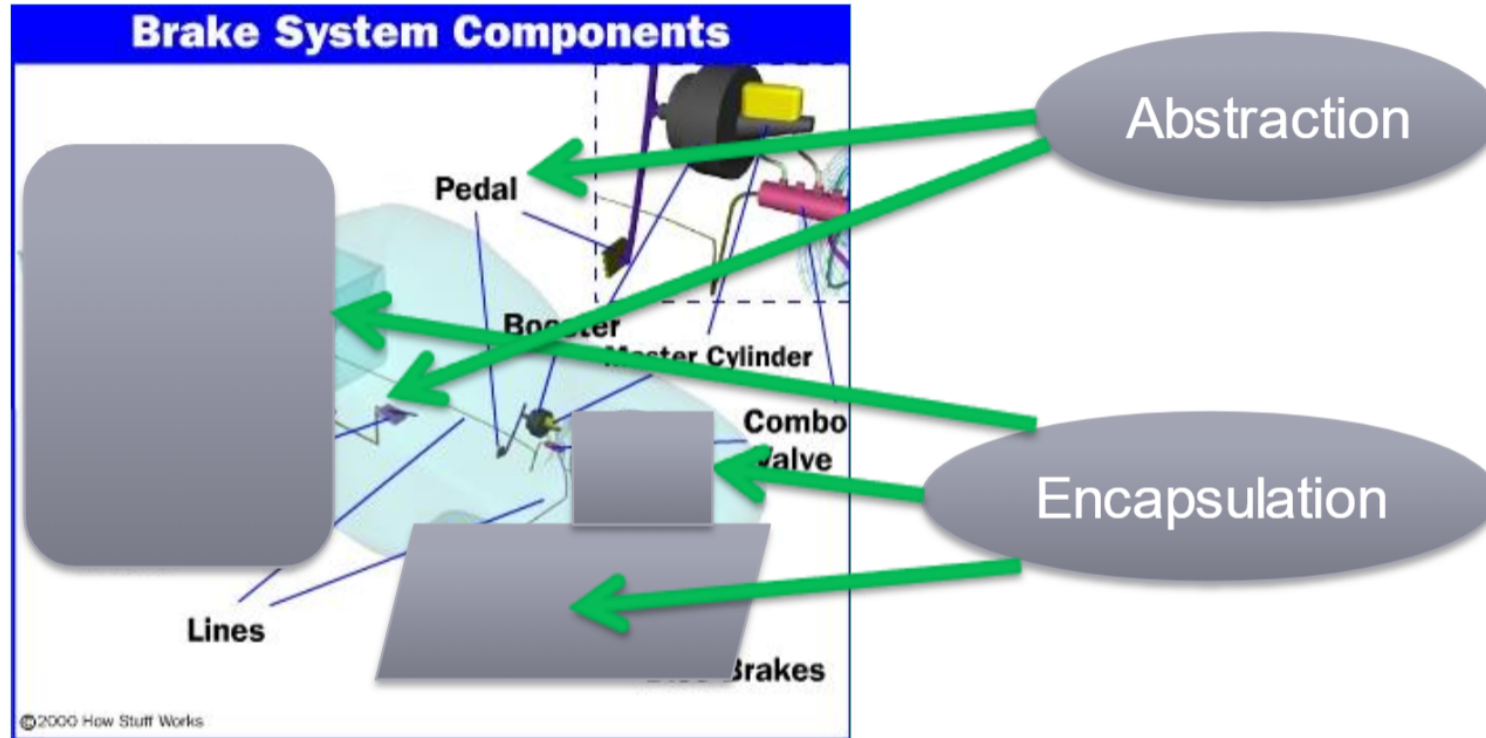
# Encapsulation And Abstraction



# Introduction to Object Oriented Programming

- Object Oriented Programming is a programming paradigm which uses "Objects" consisting of data fields and methods together with their interactions
- It is used to design applications and computer programs
- Programming technique may include features like encapsulation, abstraction, polymorphism and inheritance

# Encapsulation and Abstraction



Encapsulation is hiding the implementation level details

Abstraction is exposing only the interface

# Defining a Sample point Class

```
class Point {  
    int x;    int y;  
    void setX( int x){  
        x = (x > 79 ? 79 : (x < 0 ? 0 : x)); }  
    void setY (int y){  
        y = (y > 24 ? 24 : (y < 0 ? 0 : y)); }  
    int getX( ){ return x; }  
    int getY( ){ return y;}  
}
```

# Access Specifiers

- Java provides access specifiers to control access to class members
- Access specifiers help implement:
  - Encapsulation by hiding implementation-level details in a class
  - Abstraction by exposing only the interface of the class to the external world
- The **private** access specifier is generally used to encapsulate or hide the member data in the class
- The **public** access specifier is used to expose the member functions as interfaces to the outside world



# Class Declaration for Point

```
class Point{
    private int x;
    private int y;
    public void setX( int x){
        x= (x > 79 ? 79 : (x < 0 ? 0 : x));
    }
    public void setY (int y){
        y= (y > 24 ? 24 : (y < 0 ? 0 : y));
    }
    public int getX( ){
        return x;
    }
    public int getY( ){
        return y;
    }
}
```

# Class Declaration for Point (Contd.).

```
class PointDemo {  
    public static void main(String args[ ] ){  
        int a, b;  
        Point p1 = new Point( );  
        p1.setX(22);  
        p1.setY(44);  
        a = p1.getX( );  
        System.out.println("The value of a is "+a);  
        b = p1.getY( );  
        System.out.println("The value of b is "+b);  
    }  
}
```

## Expected Output :

The value of a is 22  
The value of b is 24

## Actual Output :

The value of a is 0  
The value of b is 0

?

# Class Declaration for Point - modified

```
class Point{
    private int x;
    private int y;
    public void setX( int x) {
        this.x= (x > 79 ? 79 : (x < 0 ? 0 :x));
    }
    public void setY (int y) {
        this.y= (y > 24 ? 24 : (y < 0 ? 0 : y));
    }
    public int getX( ){
        return x;
    }
    public int getY( ){
        return y;
    }
}
```

# Class Declaration for Point - modified (Contd.).

```
class PointDemo {  
    public static void main(String args[ ] ){  
        int a, b;  
        Point p1 = new Point( );  
        p1.setX(22);  
        p1.setY(44);  
        a = p1.getX( );  
        System.out.println("The value of a is "+a);  
        b = p1.getY( );  
        System.out.println("The value of b is "+b);  
    }  
}
```

## Output:

The value of a is 22  
The value of b is 24

# Summary

In this module, we were able to:

- Understand the relevance of Object Oriented Programming techniques
- Implement Encapsulation and Abstraction



Thank You