Advanced Programming
Lecture 2

Md. Tariquzzaman (TZF)
Lecturer, BUBT

tariquzzamanfaisal@bubt.edu.bd

If you're tired, have some Java

Chapter 2 - An Overview of Java

- **Object Oriented Programming (OOP)**
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism

Classes and Objects in Java

```
class Main {
    public static void main(String[] args) {
        // create objects led and halogen
        Lamp led = new Lamp();
        Lamp halogen = new Lamp();
        led.turnOn();
        halogen.turnOn();
        led.turnOff();
    }
}
```

```
class Lamp {  4 usages
    boolean isOn;  4 usages
    void turnOn() {  1 usage
        isOn = true;
        System.out.println("Light on? " + isOn);
    }
    void turnOff() {  1 usage
        isOn = false;
        System.out.println("Light on? " + isOn);
    }
}
```


Object Oriented Programming (OOP)

Encapsulation:

How can you protect yourself, from yourself?

Object Oriented Programming (OOP)

- **Encapsulation:**
  - Encapsulation allows you to **hide** specific information and **control access** to the object's internal state. A getter method retrieves an attribute and a setter method changes it.
  - It refers to the **bundling of data/variables and methods** that operate on that data within a single unit, which is called a class in Java.

Object Oriented Programming (OOP)

Why Encapsulation?

- Better control of class attributes and methods
- Class attributes can be made **read-only** (if you only use the `get` method), or **write-only** (if you only use the `set` method)
- Flexible: the programmer can change one part of the code without affecting other parts
- Increased security of data

Object Oriented Programming (OOP)

Encapsulation in Code:

```java
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.name = "John";  // error
    System.out.println(myObj.name); // error
  }
}
```

```java
public class Person {
  private String name; // private = restricted access

  // Getter
  public String getName() {
    return name;
  }

  // Setter
  public void setName(String newName) {
    this.name = newName;
  }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Person myObj = new Person();
        myObj.setName("John");  // using setter
        System.out.println(myObj.getName());  /
    }
}
```

Object Oriented Programming (OOP)

- **Abstraction:**
  - The process of **hiding the internal details** of an application from the outer world. Abstraction is used to describe things in **simple** terms. ○ Objects are the building blocks of Object-Oriented Programming. An object contains some properties and methods. We can hide them from the outer world through **access modifiers**. We can provide access only for

Data **abstraction** is the process of hiding certain details and showing only essential information to the user.
Abstraction can be achieved with either **abstract classes** or **interfaces**

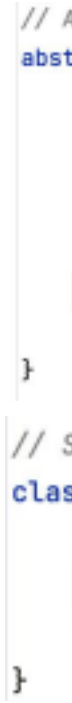
required functions and properties to the other programs.

Object Oriented Programming (OOP)

Abstraction:

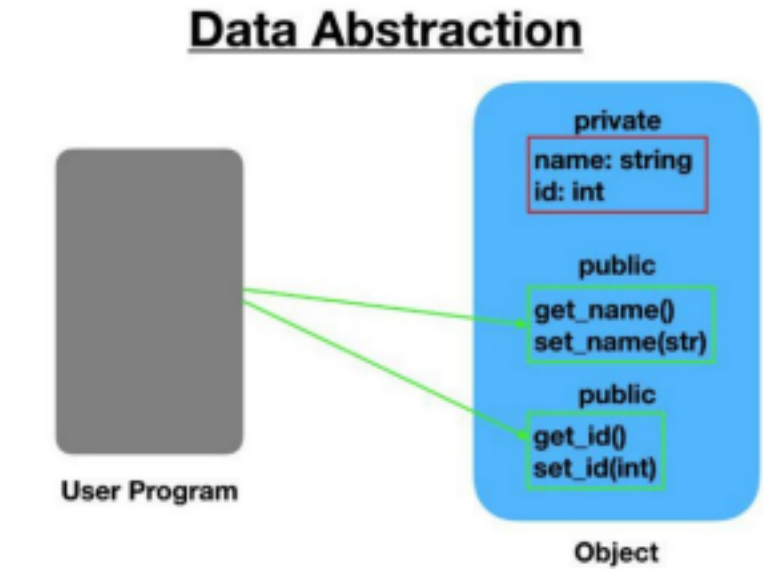
```
// Another subclass (inheriting from Shape)
class Rectangle extends Shape {  1 usage
    public void draw() {  2 usages
        System.out.println("Drawing a rectangle");
    }
}
```

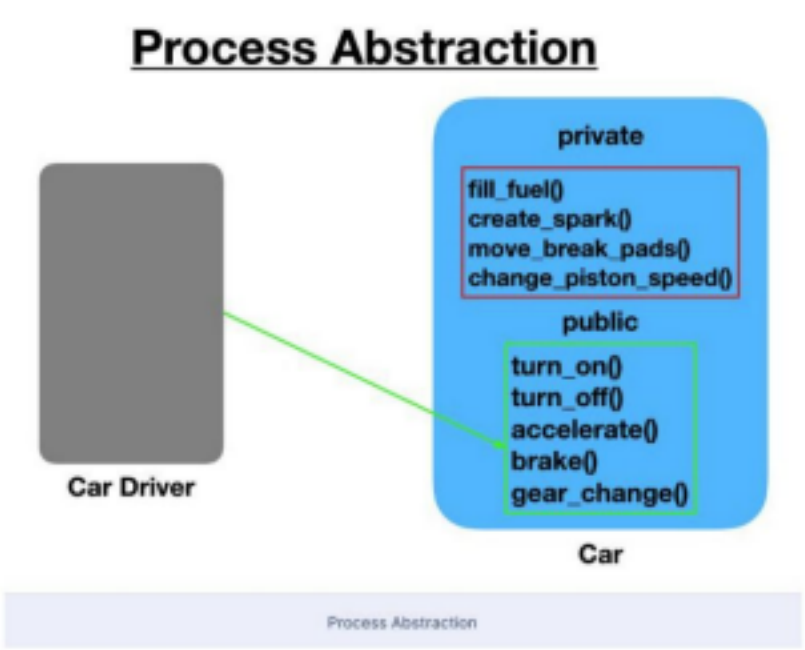
```
// Main class to demonstrate abstraction
public class Main {
    public static void main(String[] args) {
        // Create a Circle object
        Shape myCircle = new Circle();
        myCircle.draw();        // Outputs: Drawing a circle
        myCircle.display();     // Outputs: Displaying the shape

        // Create a Rectangle object
```



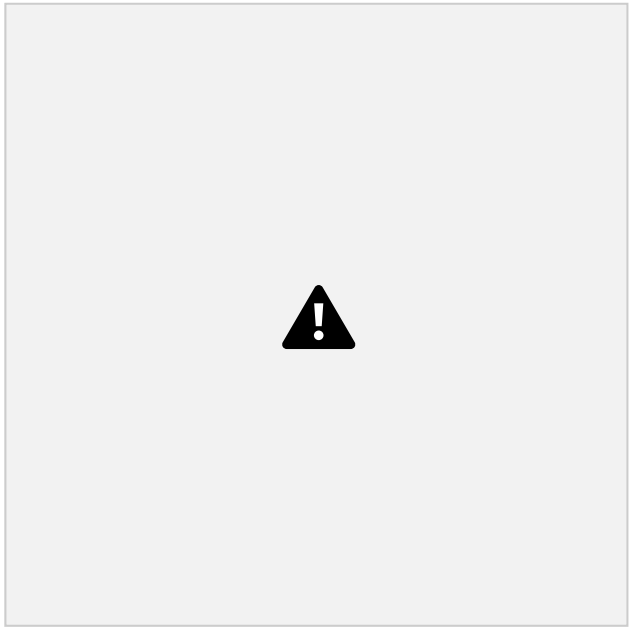
Object Oriented Programming (OOP)

- **Abstraction:**





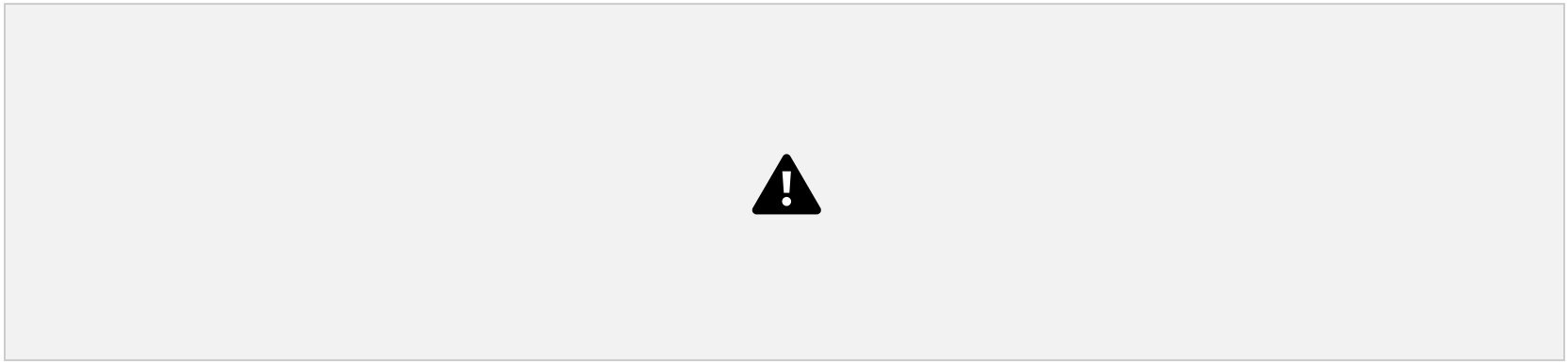
Object Oriented Programming (OOP)

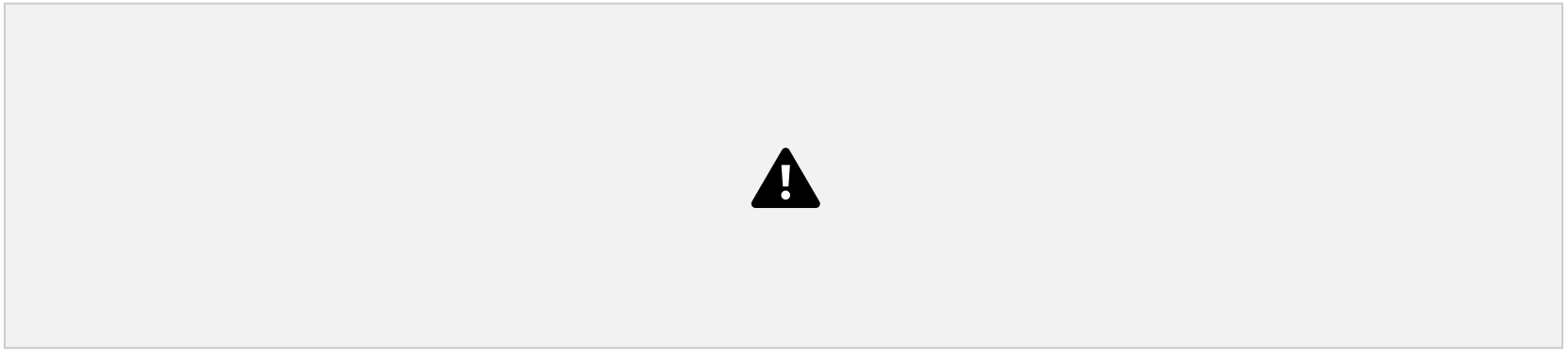


What is the difference between Abstraction and Encapsulation?

Object Oriented Programming (OOP)

Abstraction vs Encapsulation:





Object Oriented Programming (OOP)

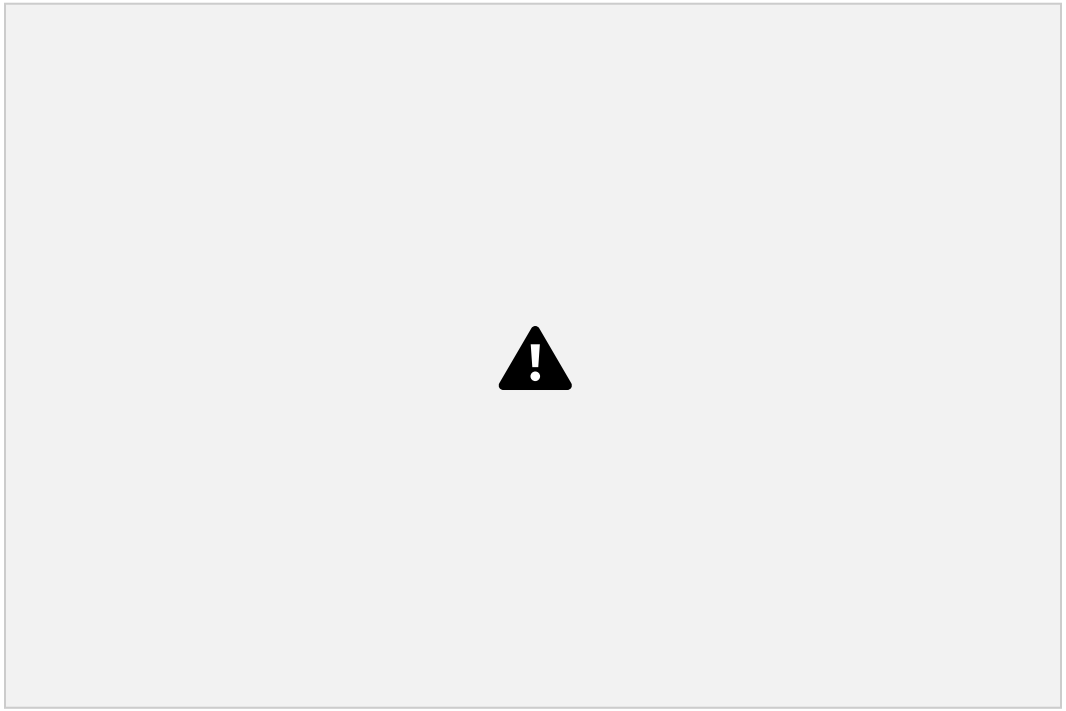
- **Abstraction** focuses on hiding the complexity by exposing only essential features and functionalities.
- **Encapsulation** focuses on hiding the internal state and protecting it from outside interference, ensuring that the object's data and behavior are used correctly.

Combined Use:

- ***Abstraction is often achieved through encapsulation.*** By encapsulating the details of an object's implementation, you create a simpler, abstract interface for interacting with that object.
- For example, in the Car class, the methods accelerate, brake, and getSpeed abstract the details of how speed is managed, while encapsulation ensures that the speed variable is protected and accessed only through these methods.

Object Oriented Programming (OOP)

Inheritance



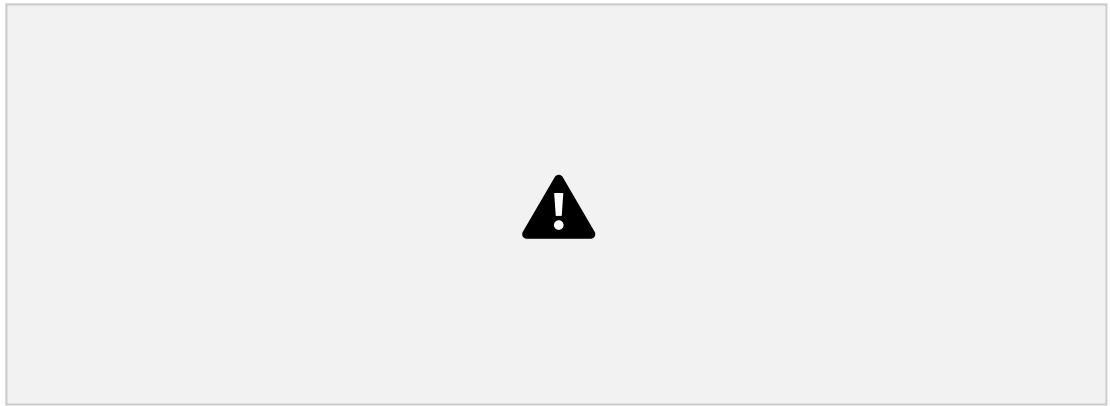
Object Oriented Programming (OOP)

Inheritance

- It’s a programming procedure that allows you to reuse code by referencing the
- behaviors and data of an object.
- In other words, a class that inherits from another class shares all the attributes and methods of the referenced class.

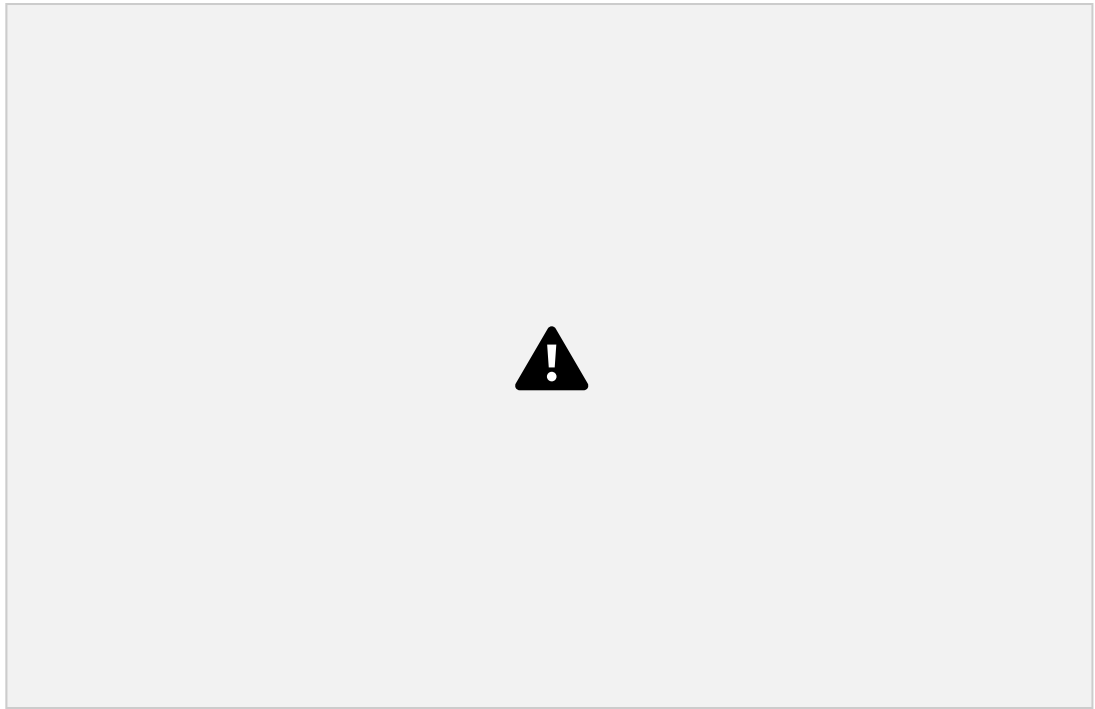
Object Oriented Programming (OOP)

Inheritance



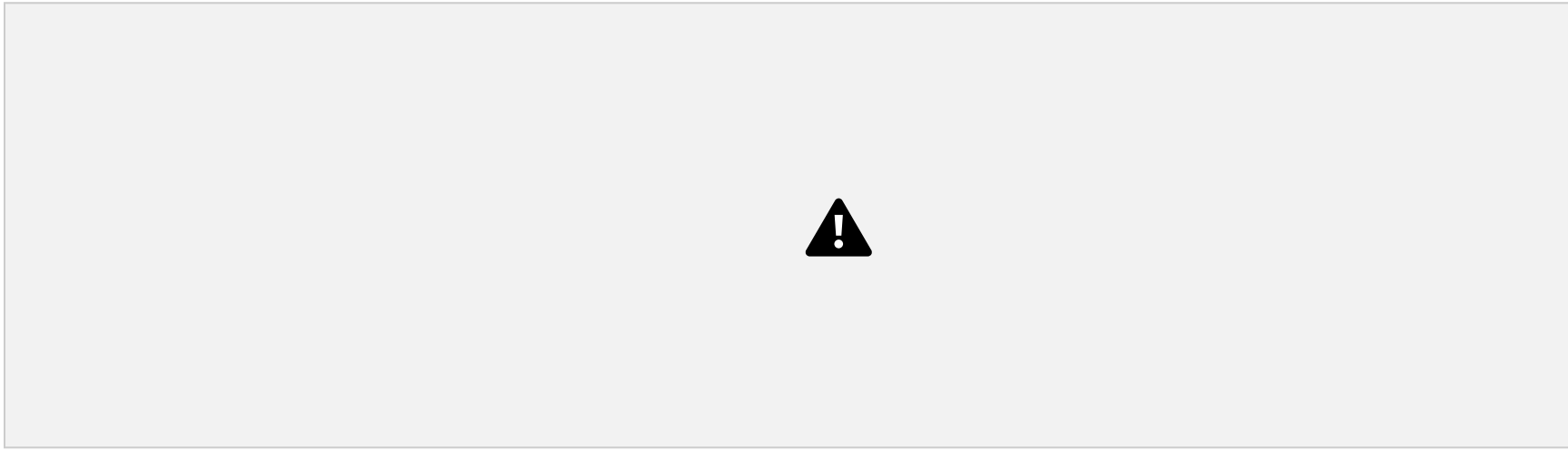
Object Oriented Programming (OOP)

Inheritance



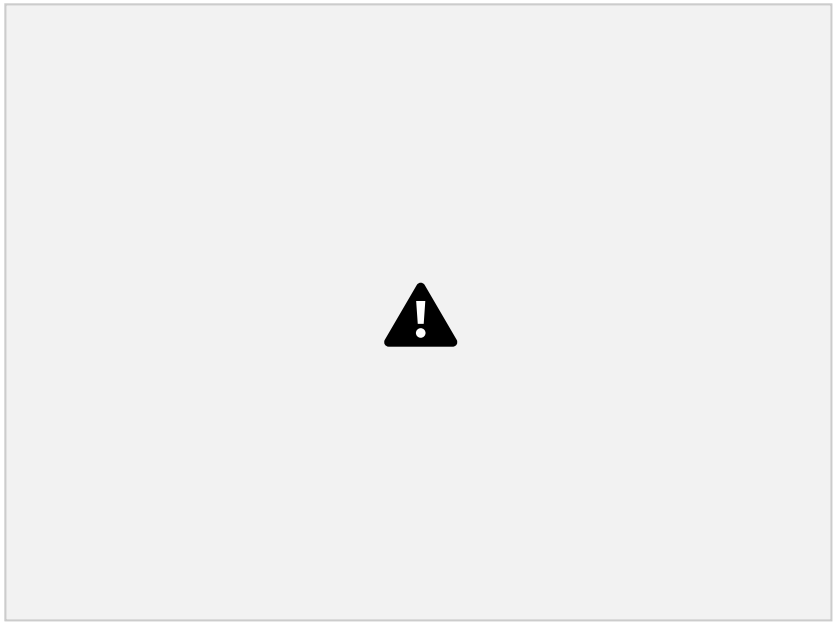
Object Oriented Programming (OOP)

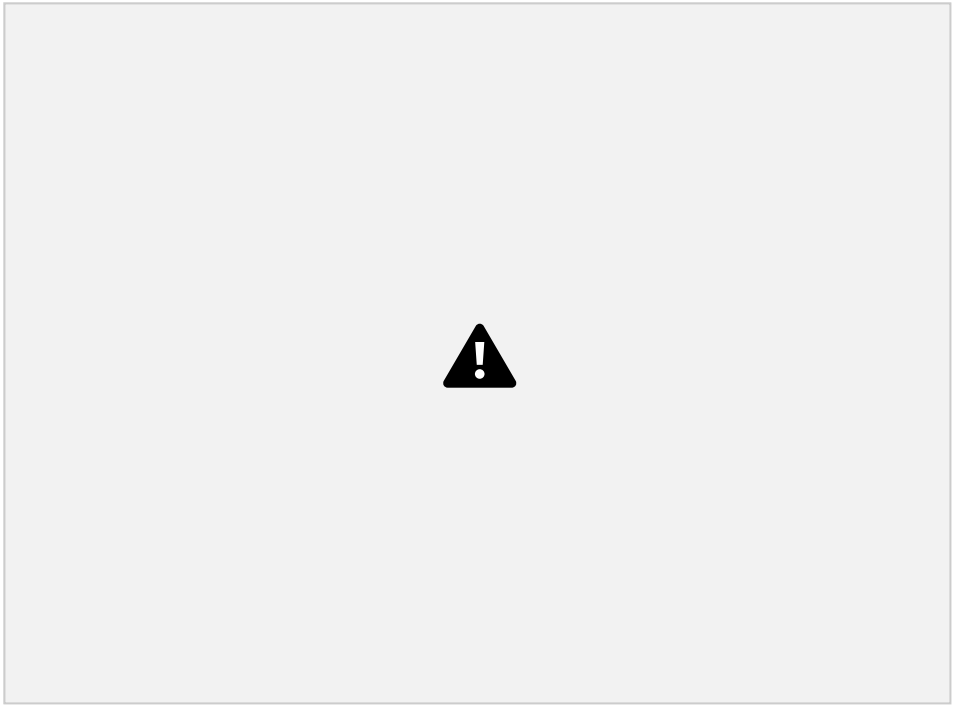
Polymorphism



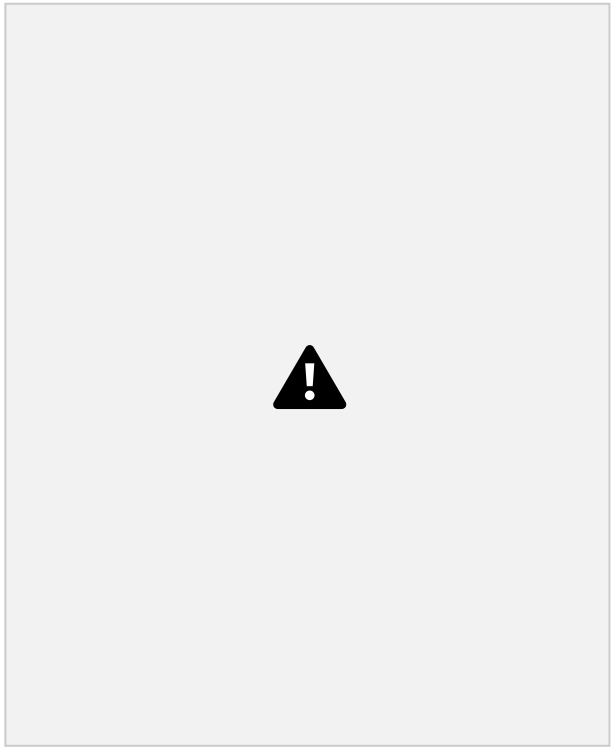
Object Oriented Programming (OOP)

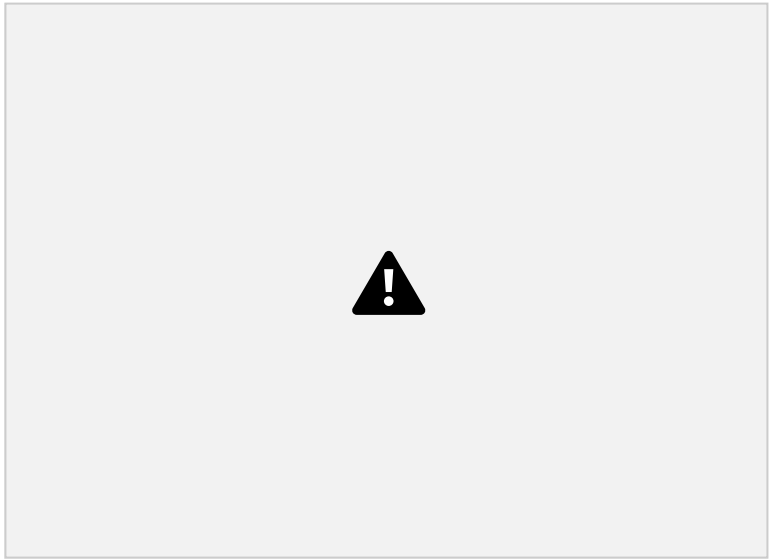
Polymorphism





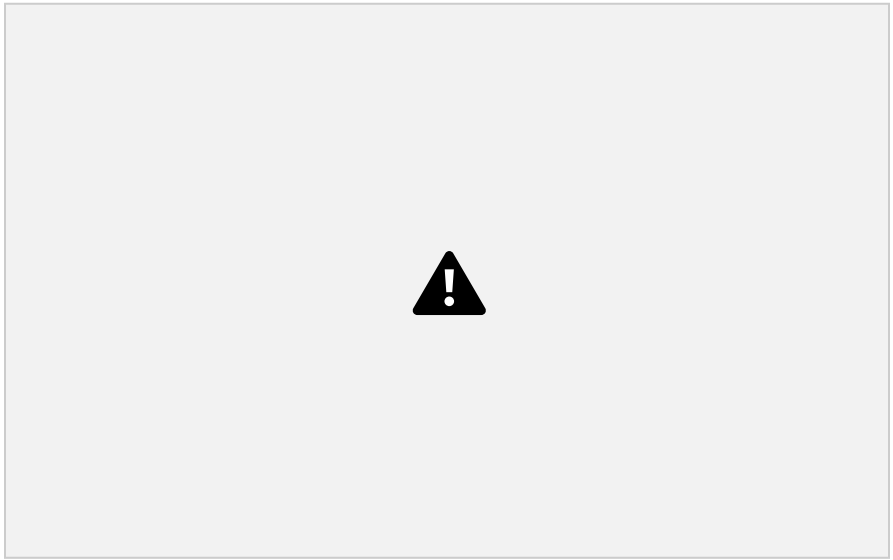
Object Oriented Programming (OOP)

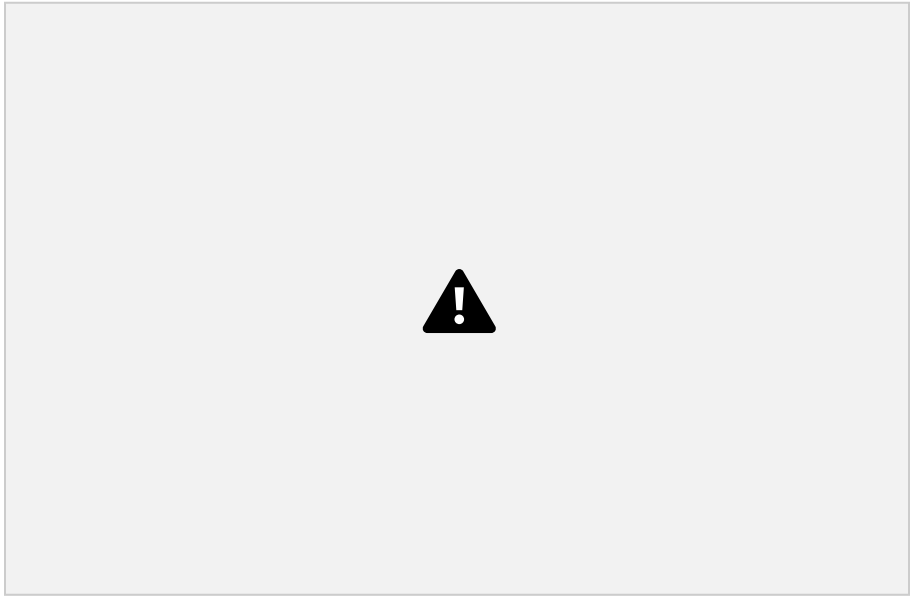




Object Oriented Programming (OOP)

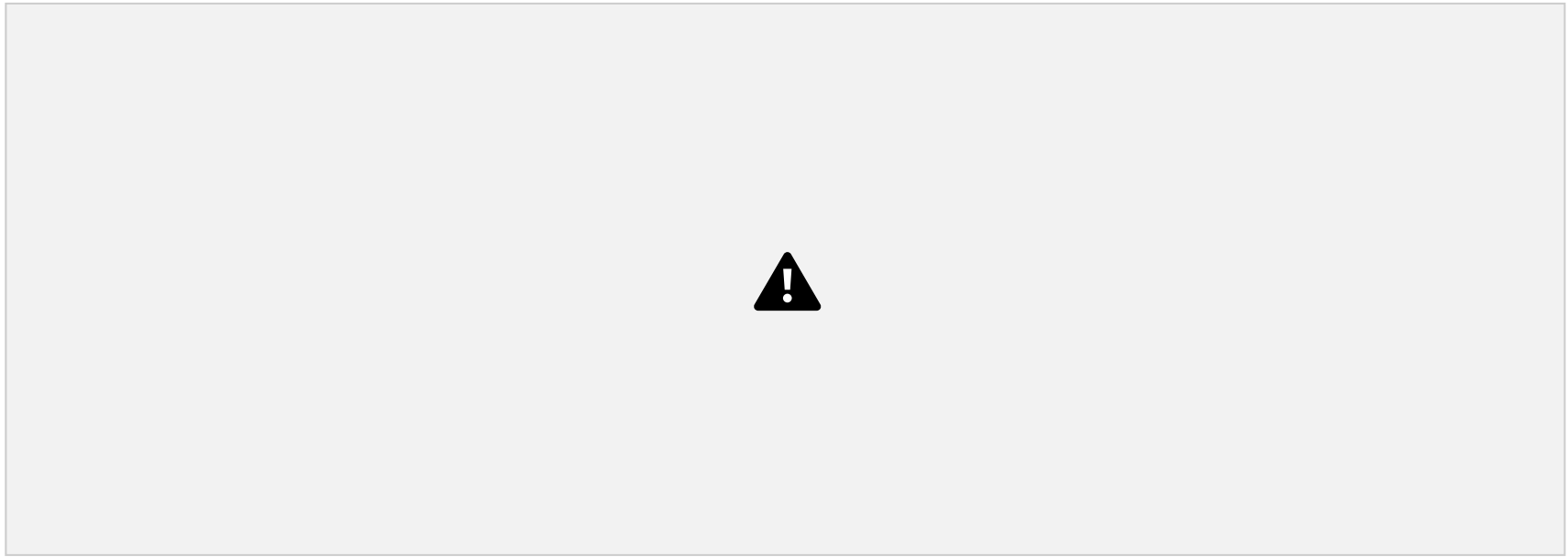
Polymorphism: Overriding vs Overloading





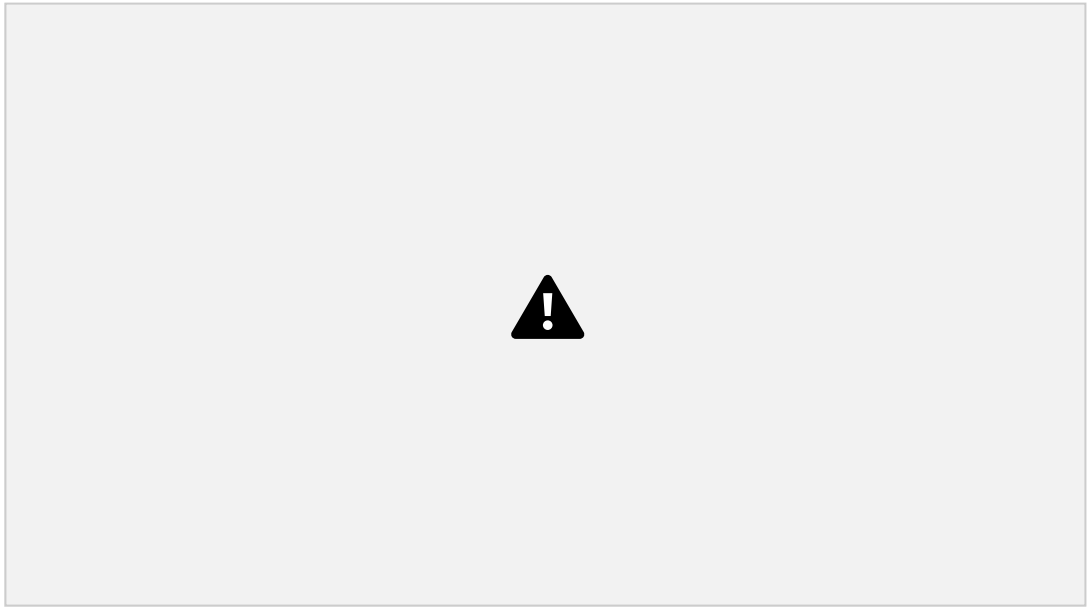
Object Oriented Programming (OOP)

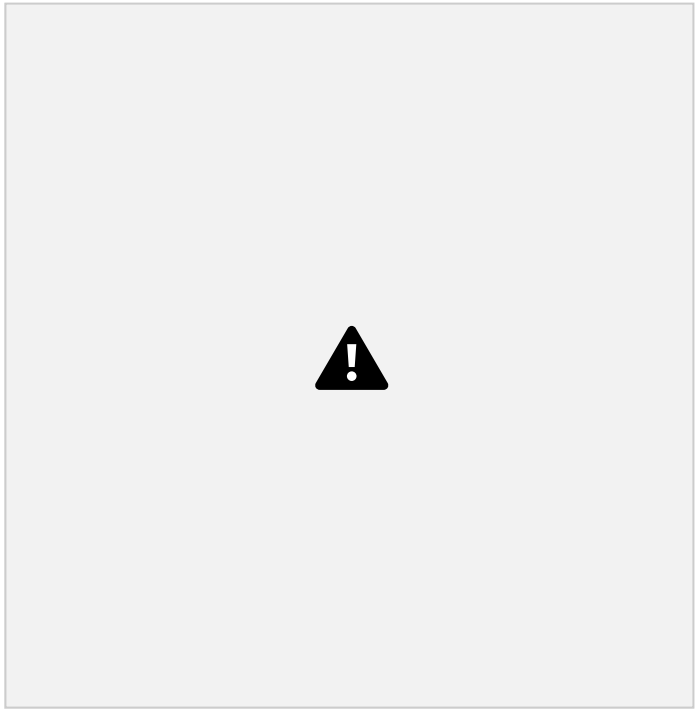
Polymorphism: Overriding



Object Oriented Programming (OOP)

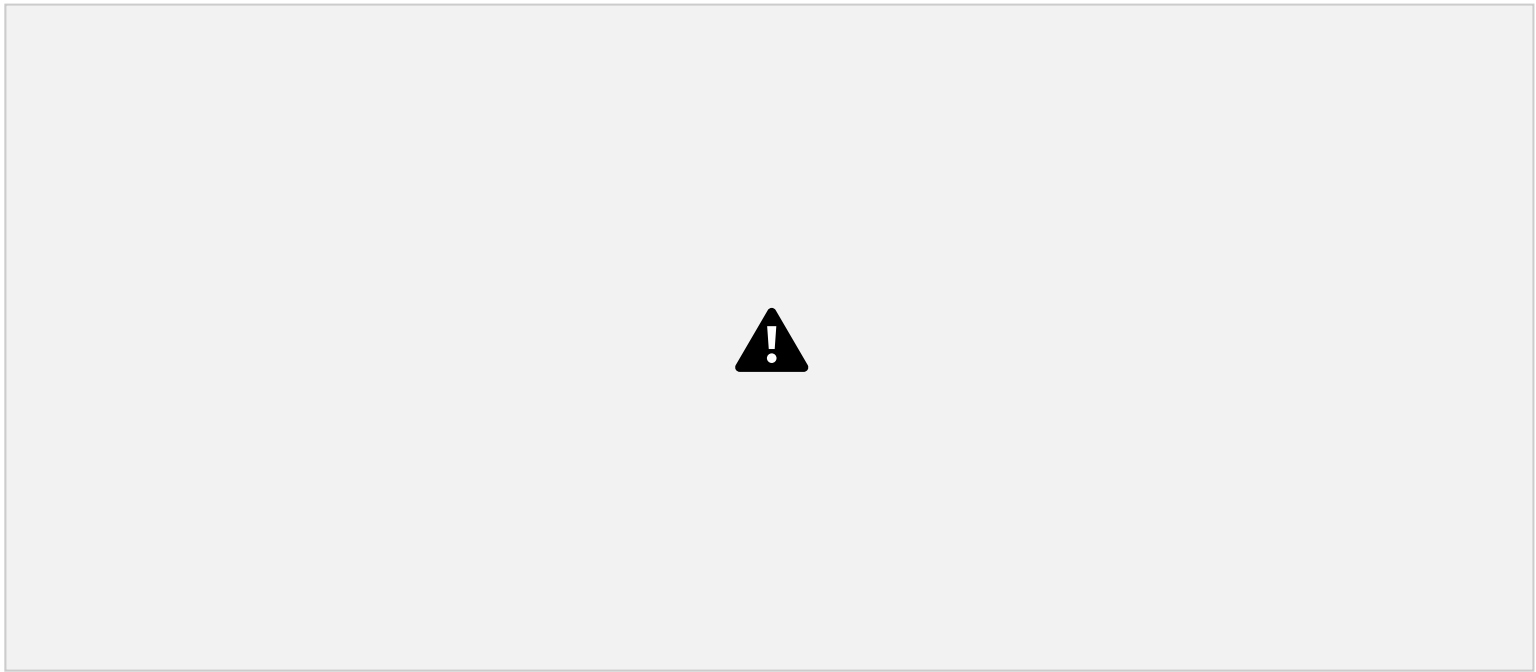
Polymorphism: Overloading





Object Oriented Programming (OOP)

Polymorphism: Overloading



That's All!