



Object Oriented Programming

CS F213

BITS Pilani
Pilani Campus



OOP Basics

BITS Pilani
Pilani Campus

Basic OOP concepts



- Class
- Object
- Encapsulation
- Inheritance
- Polymorphism

FRIUT



Attribute
Name,color
Method
Cost()



Attribute
Name,color
Method
Cost()

Abstract Data Type (ADT)

- A structure that contains both **data** and the **actions** to be performed on that data.
- *Abstract Datatype* is special kind of *datatype*, whose behavior is defined by a set of values (data) and set of operations (actions)
- **Class** is an implementation of an Abstract Data Type.

Examples

innovate

achieve

lead

Person Objects



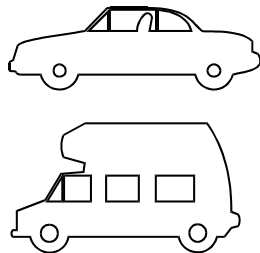
**Abstract
Into** →

Person Class

Attributes: Name, Age, Sex

Operations: Speak(), Listen(), Walk()

Vehicle Objects



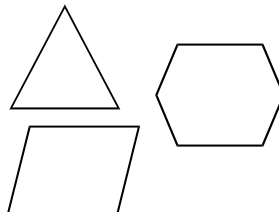
**Abstract
Into** →

Vehicle Class

Attributes: Name, Model, Color

Operations: Start(), Stop(), Accelerate()

Polygon Objects



**Abstract
Into** →

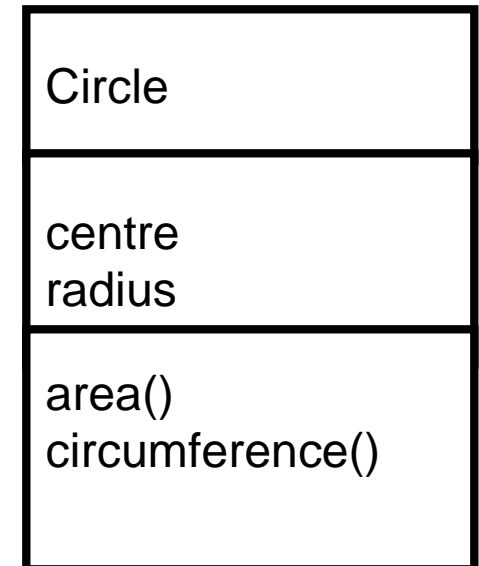
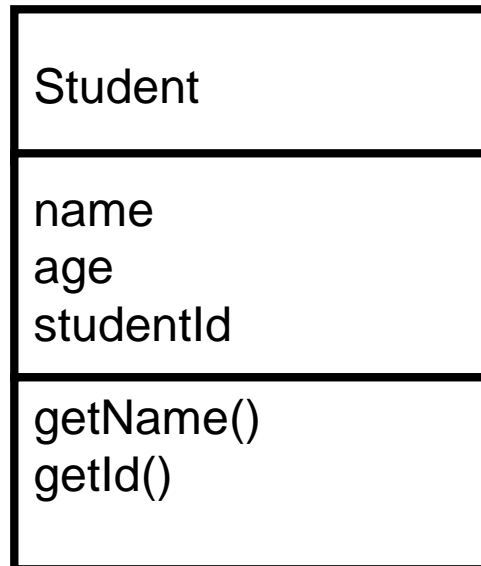
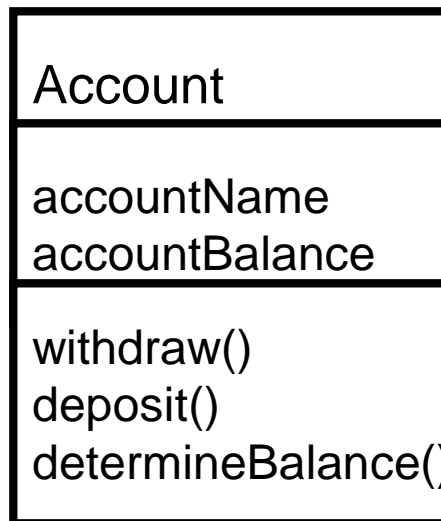
Polygon Class

Attributes: Vertices, Border,
Color, FillColor

Operations: Draw(), Erase(), Move()

Class

- Class is a set of *attributes* and *operations* that are performed on the attributes.
- A blueprint from which individual objects can be created.
- A class defines all the properties common to the object
 - *attributes* and *methods*.

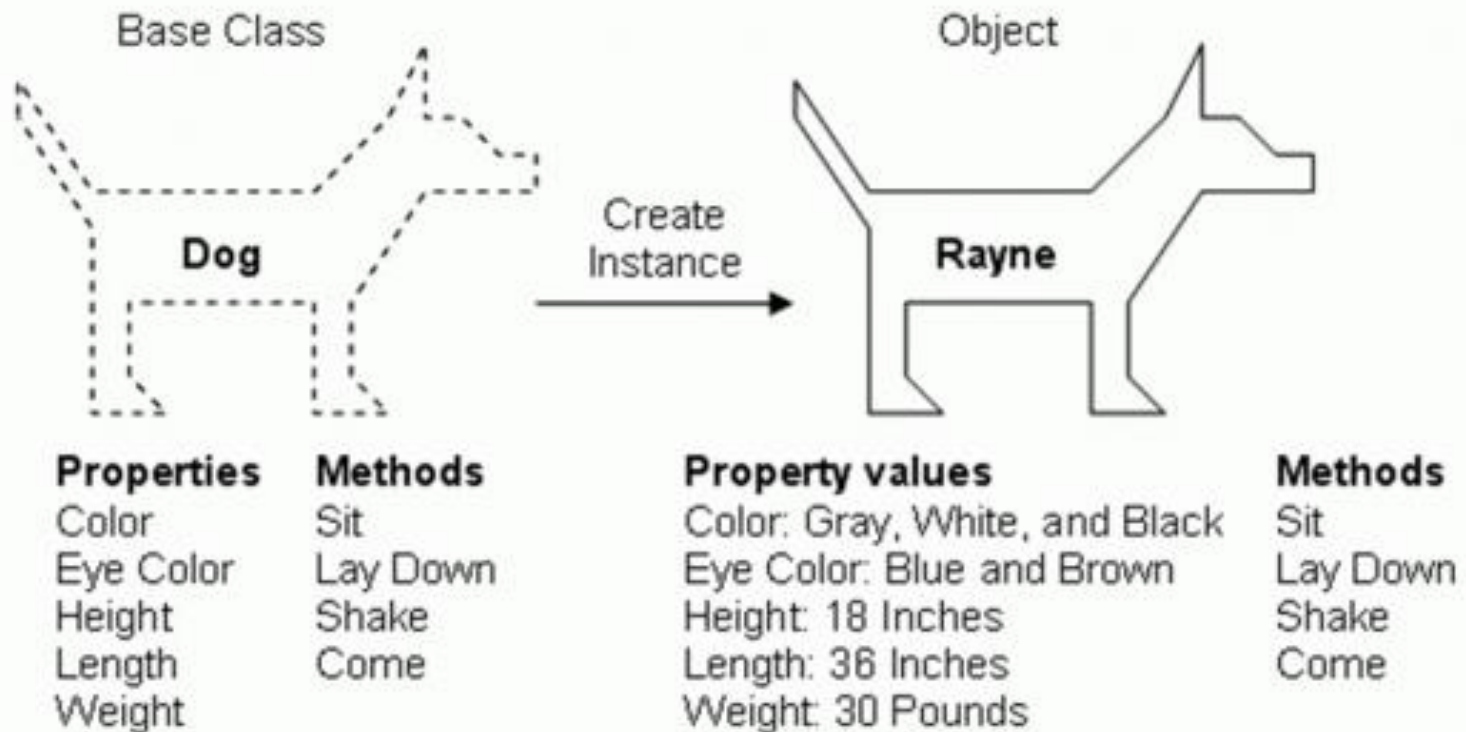


Objects

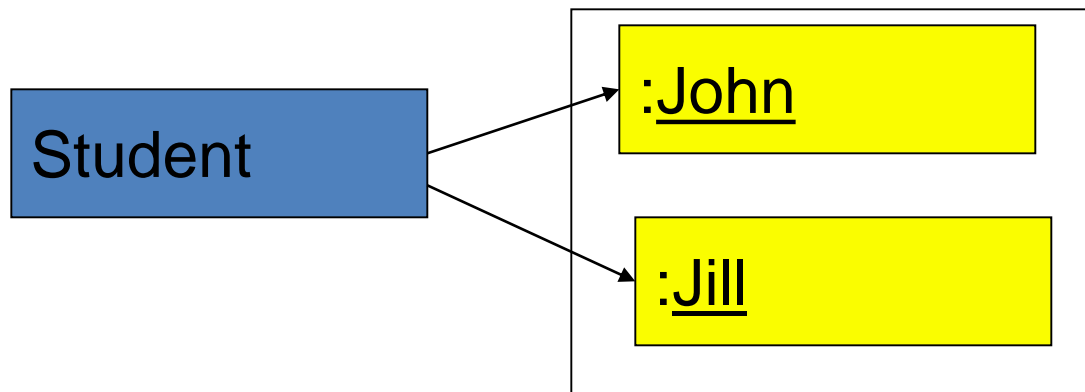
- Instance of the class
- Entity that has state and behavior
- Each object has an address and takes up memory
- It can communicate without knowing other object's code or data

Classes vs. Objects

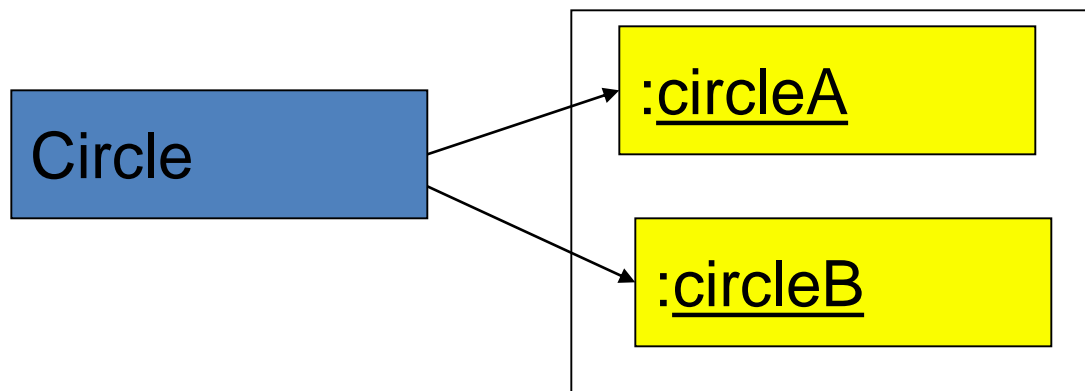
Software Object



Classes/Objects



John and Jill are
objects of class
Student



circleA and circleB
are
objects of class
Circle

Object

- Objects have state and classes don't.

John is an object (instance) of class Student.

name = "John", age = 20, studentId = 1236

Jill is an object (instance) of class Student.

name = "Jill", age = 22, studentId = 2345

circleA is an object (instance) of class Circle.

centre = (20,10), radius = 25

circleB is an object (instance) of class Circle.

centre = (0,0), radius = 10

Class/Object Example

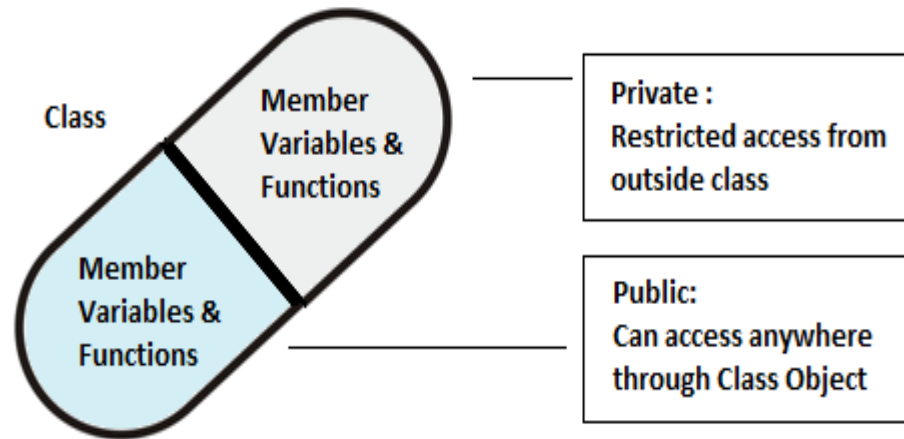
```
class Student{
    int id;
    String name;
}
class TestStudent{
    public static void main(String args[]){
//Creating object
        Student s1=new Student();
//Initializing object
        s1.id=253;
        s1.name="Sathish";
//Printing data
        System.out.println(s1.id+" "+s1.name);
    }}
```



THREE OOP PRINCIPLES

Encapsulation

- Encapsulation is:
 - Binding the data with the code that manipulates it.
 - It keeps the data and the code safe from external interference



- Information can be manipulated through operations performed on the object. Implementation is hidden from the user.
- Object support *Information Hiding/Abstraction* – Some attributes and methods can be hidden from the user.

Encapsulation Example

```
class Student{
    private int rollno;
    private String name;
    void insertRecord(int r, String n){
        rollno=r;
        name=n;
    }
    void displayInformation(){System.out.println(rollno+" "+name);}
}

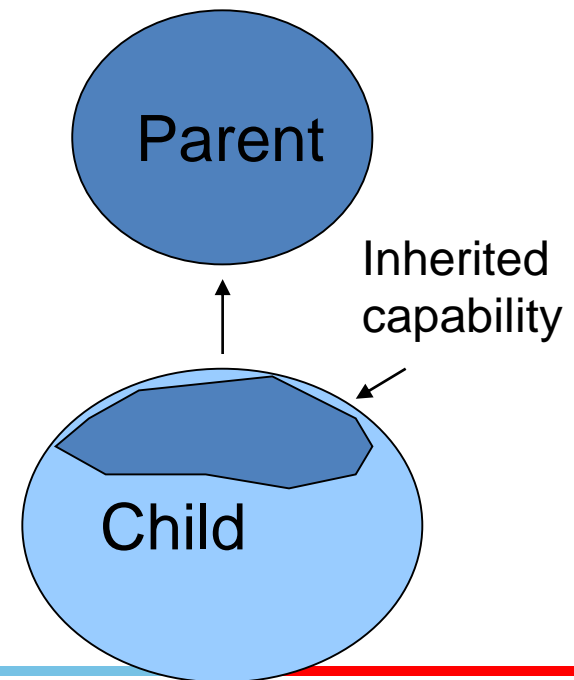
class TestStudent{
    public static void main(String args[]){
        Student s1=new Student();
        s1.insertRecord(111,"Karan");
        s1.displayInformation();
    }
}
```

Inheritance



Inheritance

- New data types (classes) can be defined as extensions to previously defined types.
- Parent Class (Super Class) – Child Class (Sub Class)
- Subclass inherits properties from the parent class.

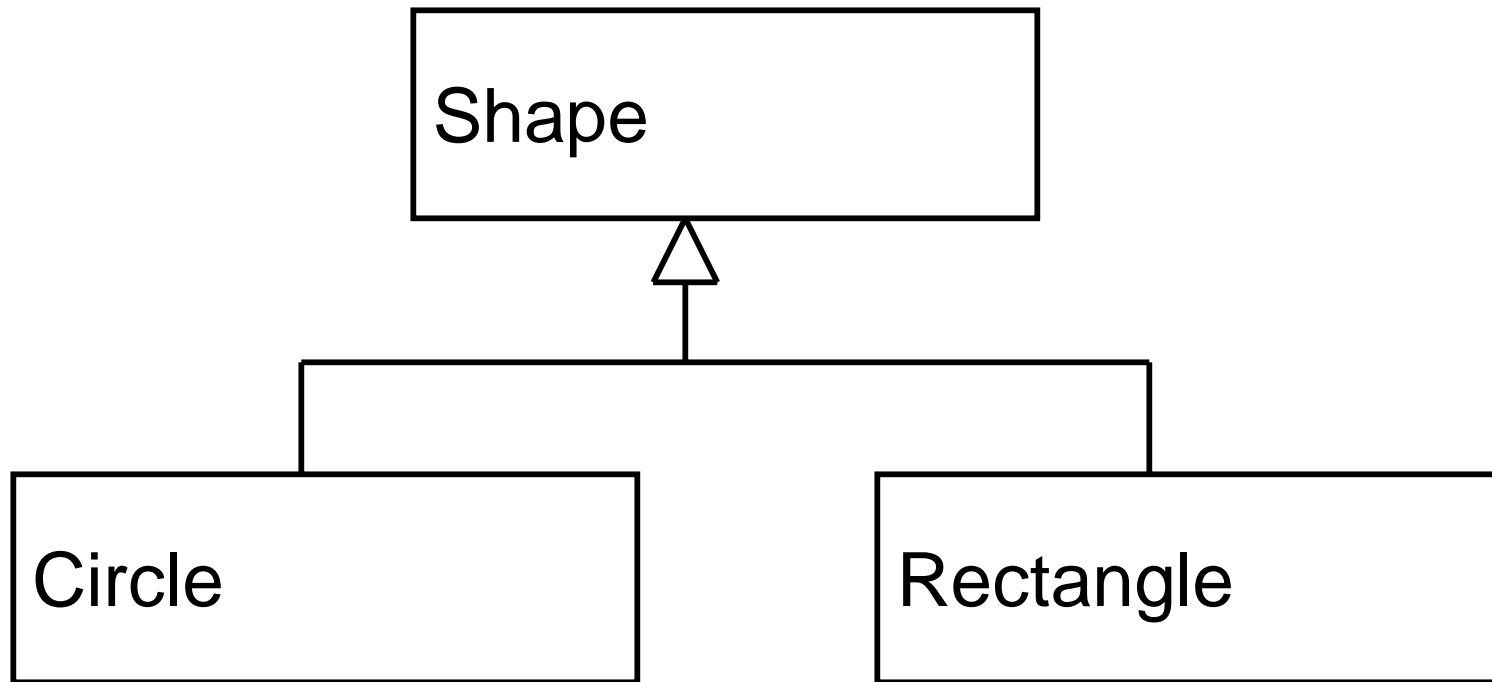


Inheritance - Example

- Example
 - Define **Person** to be a *class*
 - A **Person** has *attributes*, such as **name**, **age**, **height**, **gender**
 - Define **student** to be a *subclass* of **Person**
 - A **student** has all attributes of **Person**, plus attributes of his/her own (**student no**, **course_enrolled**)
 - A **student** *inherits* all attributes of **Person**
 - Define **lecturer** to be a *subclass* of **Person**
 - **Lecturer** has all attributes of **Person**, plus attributes of his/her own (**staff_id**, **subjectID1**, **subjectID2**)

Inheritance - Example

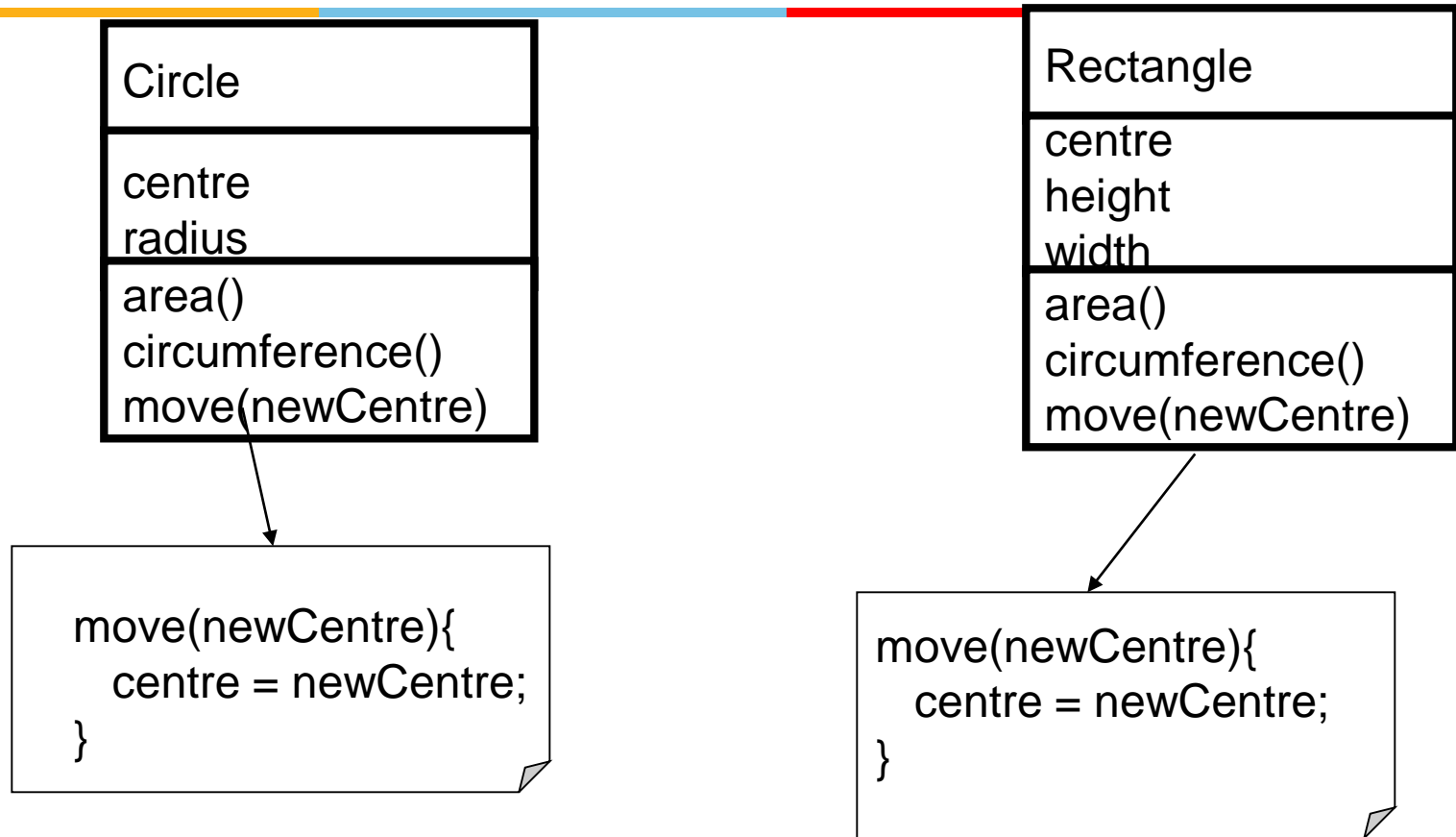
- Circle Class can be a subclass (inherited from) of a parent class - Shape



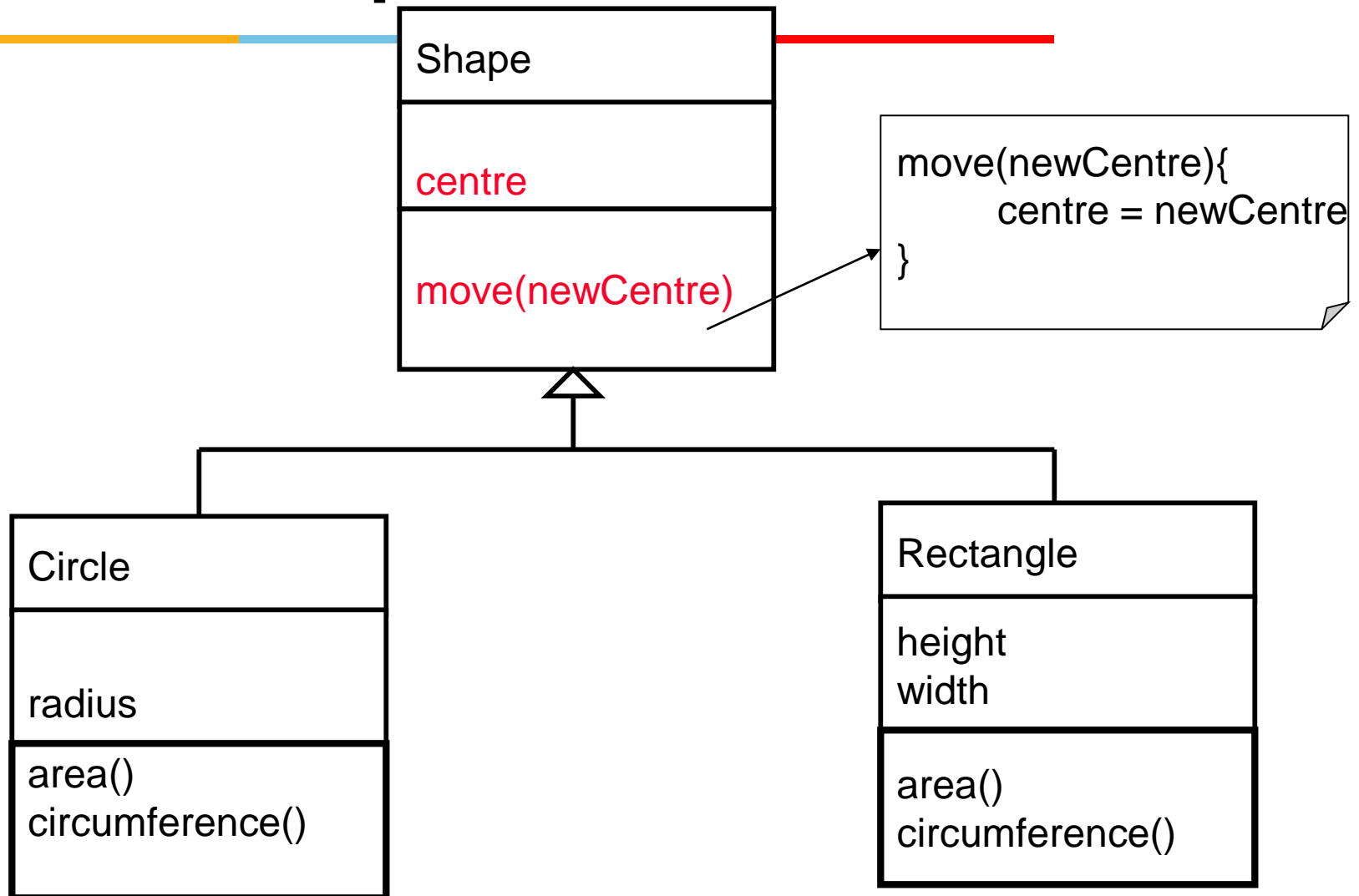
Uses of Inheritance - Reuse

- If multiple classes have common attributes/methods, these methods can be moved to a common class - parent class.
- This allows reuse since the implementation is not repeated.

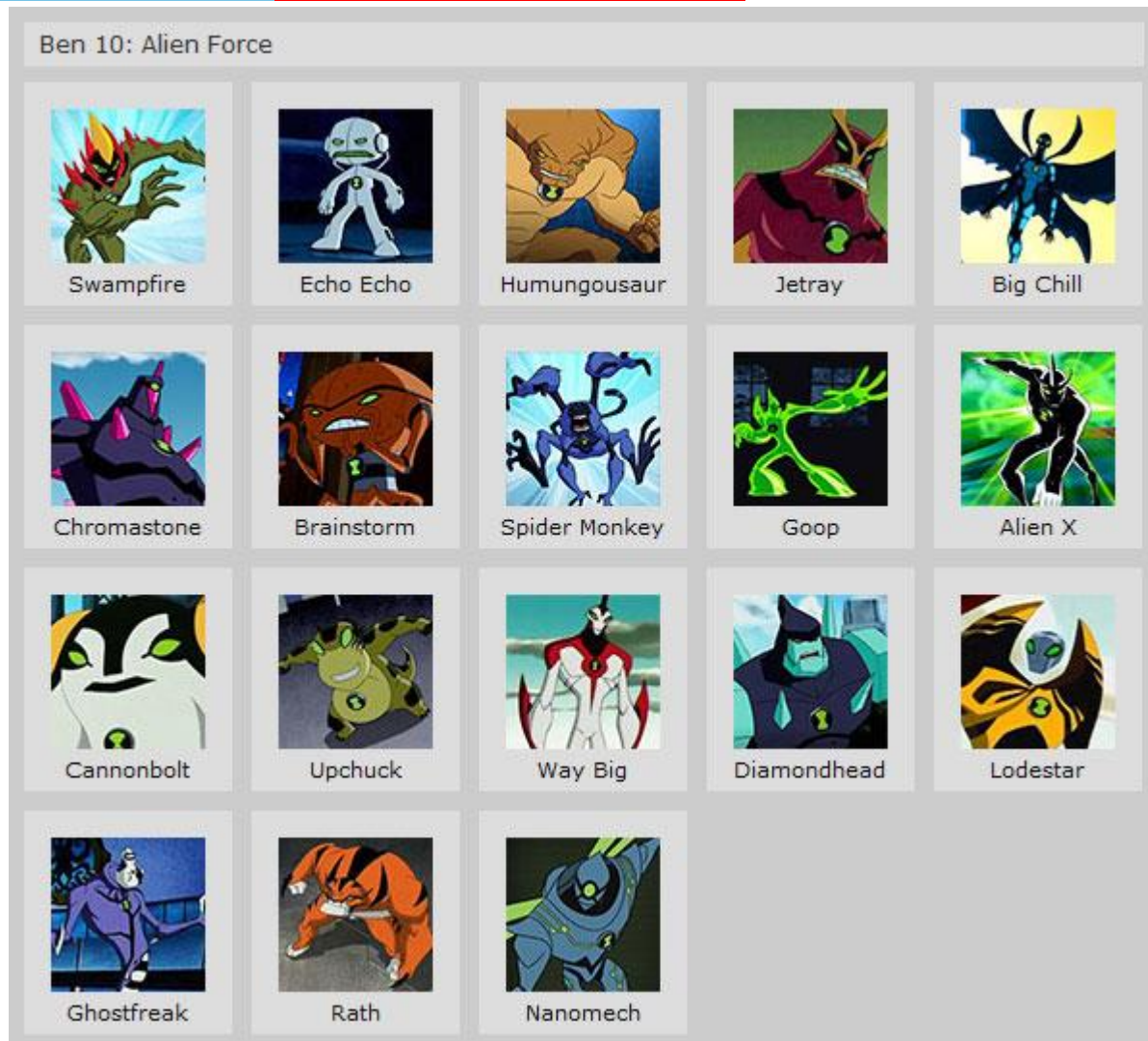
Reuse-Example



Reuse-Example



Polymorphism



Polymorphism

- Polymorphic which means “many forms” has Greek roots.
 - Poly – many
 - Morphos - forms.
- In OO paradigm polymorphism has many forms.
- Allow a single *object, method, operator* associated with different meaning depending on the type of data passed to it.

Polymorphism – Method Overloading

- Multiple methods can be defined with the same name, different input arguments.

Method 1 - initialize(int a)

Method 2 - initialize(int a, int b)

- Appropriate method will be called based on the input arguments.

initialize(2) **Method 1 will be called.**

initialize(2,4) **Method 2 will be called.**