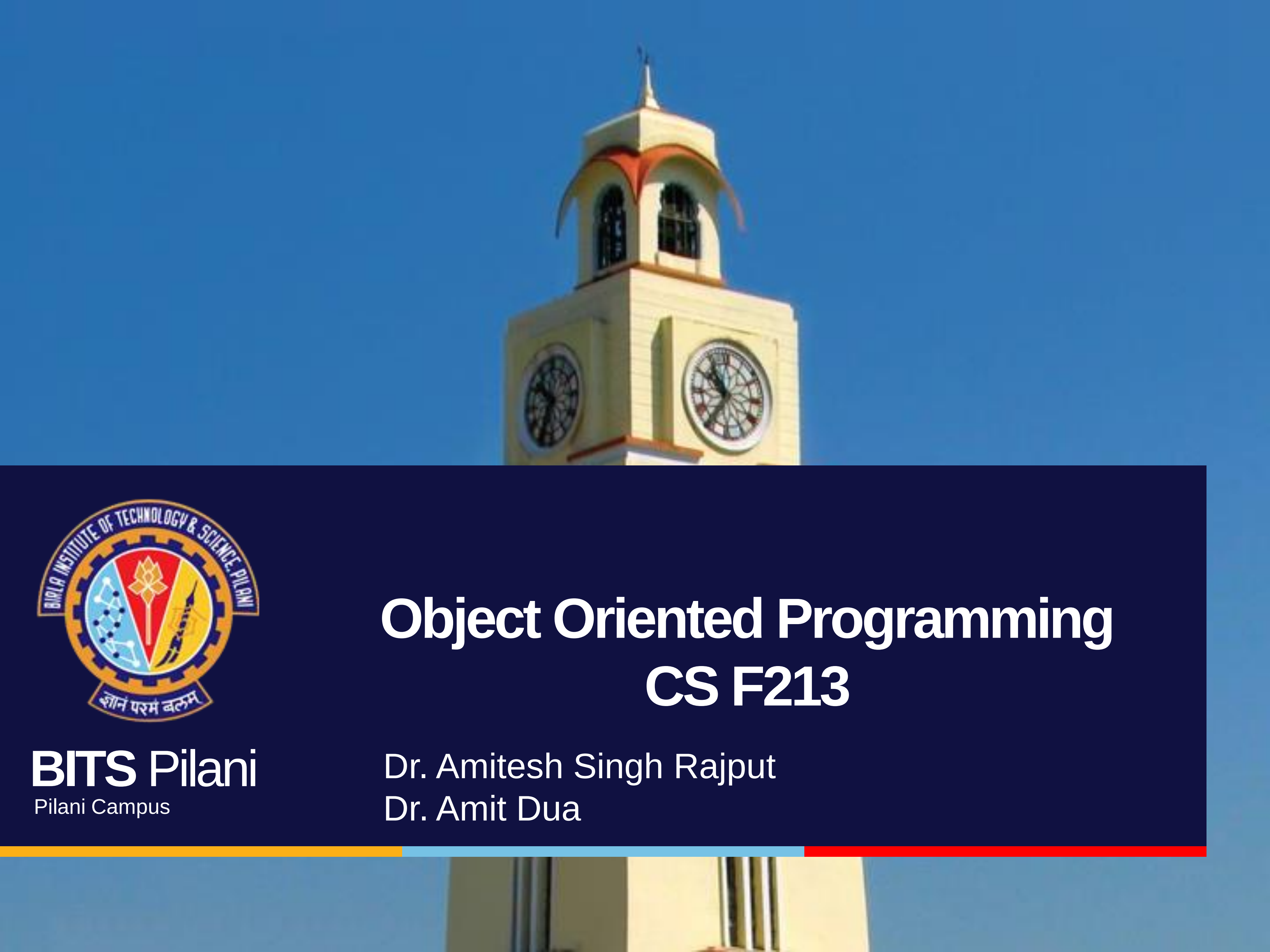


Object Oriented Programming CS F213

Dr. Amitesh Singh Rajput
Dr. Amit Dua

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani
Pilani Campus

THREE OOP PRINCIPLES

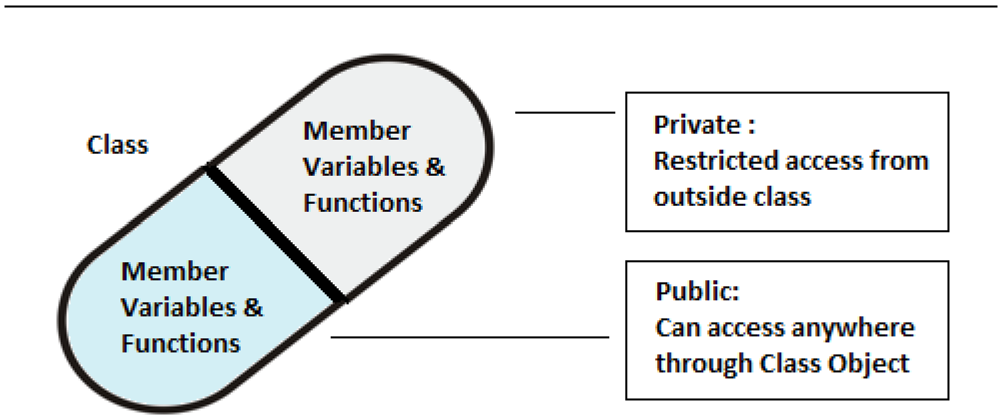
Recalling Class/Object Example

```
class Student{
 int id;
 String name;
}

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

Encapsulation

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



Class
Member Variables & Functions
Member Variables & Functions
Private : Restricted access from outside class
Public: Can access anywhere through Class Object

- The variable(s)/data of a class is hidden from any other class and can be accessed only through member functions of the same class in which it is declared.
- Encapsulation can be achieved by Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables

Example Program

```
class Student{
 private int id;
 private String name;
 void setId(int a){
   id = a;
 }
 int getId(){
   return id;
 }
 void setName(String str){
   name = str;
 }
 int getName(){
   return name;
 }
}
```

```
class TestStudent{
 public static void main(String args[]){
   Student s1 = new Student();
   s1.setId(5);
   s1.id = 10;
   System.out.println(s1.getId());
 }
}
```

Encapsulation - Advantages

- **Data Hiding:** The user will have no idea about the inner implementation of the class. It is simply passing values to a setter method and variables will get initialized with that value.

- **Increased Flexibility:** Depending on the requirement, the variables of the class can be made read-only or write-only. For read-only, ignore the setter method, whereas getter method should be ignored for write-only.

- **Reusability:** Re-usability is improved and can be changed with new requirements.

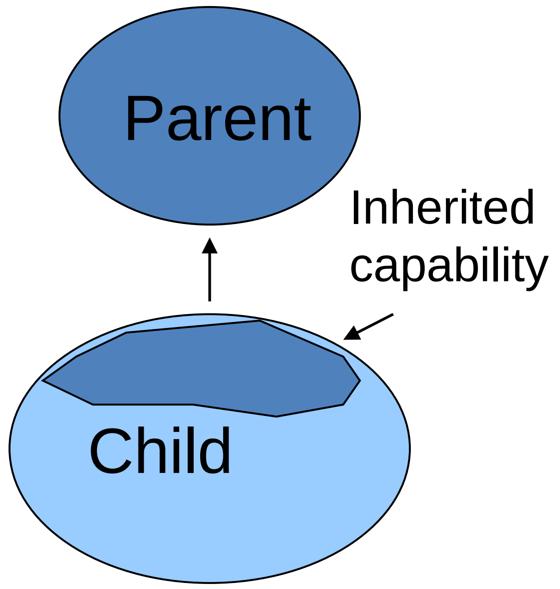
- **Easy code testing:** Encapsulated code is easy to test for unit testing.

innovate achieve lead

Inheritance

Inheritance

- One class acquires the properties of another class.
- Parent Class (Super Class) – Child Class (Sub Class).
- Child inherits properties from the parent class.

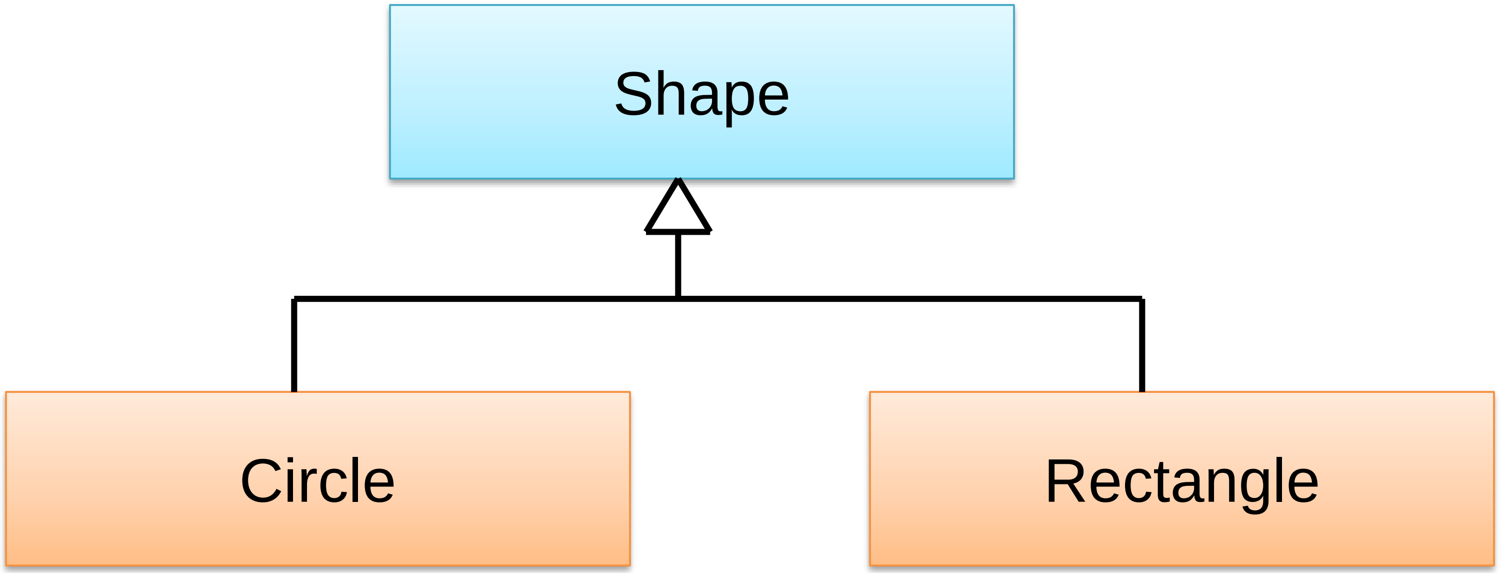


Inheritance - Example 1

- 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 2

- Circle and Rectangle are subclasses of a parent class – Shape.

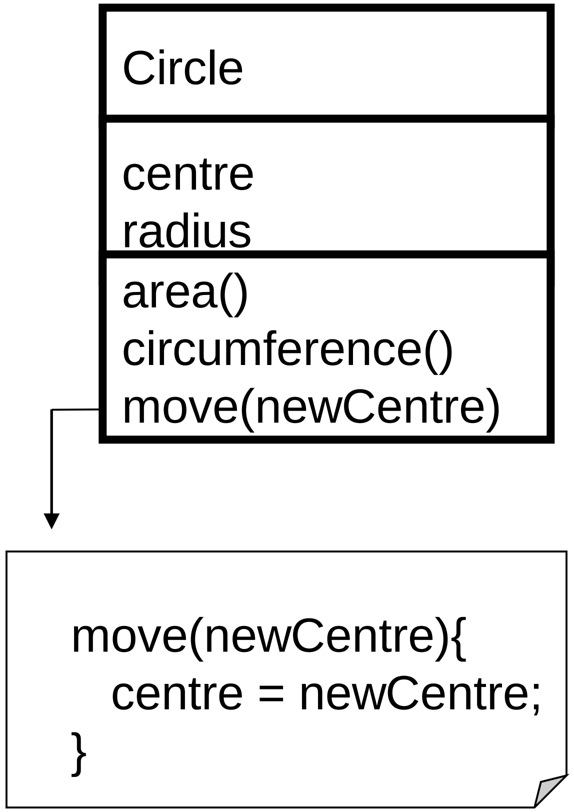


Reuse: The advantage of Inheritance

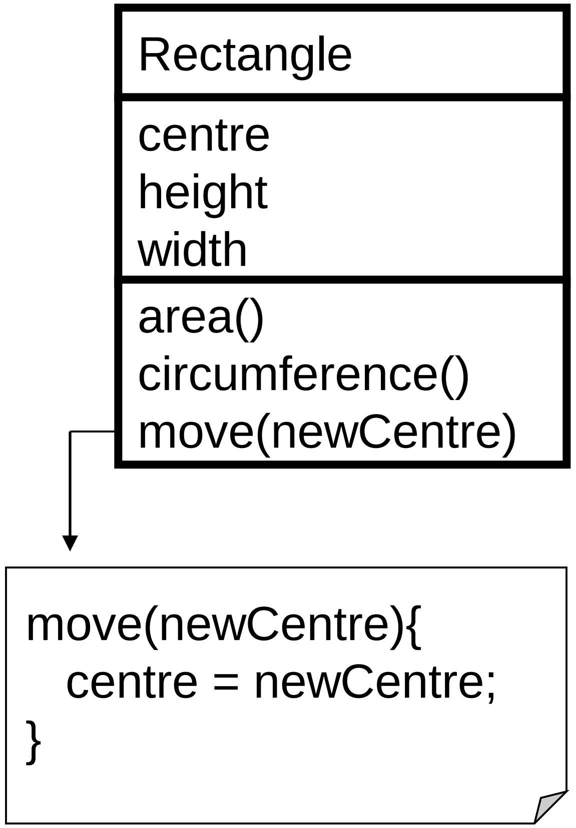
- If multiple classes have common attributes/methods, these methods can be moved to a common class - parent class.

- This allows reuse since the implementation is not repeated.

Reuse-Example

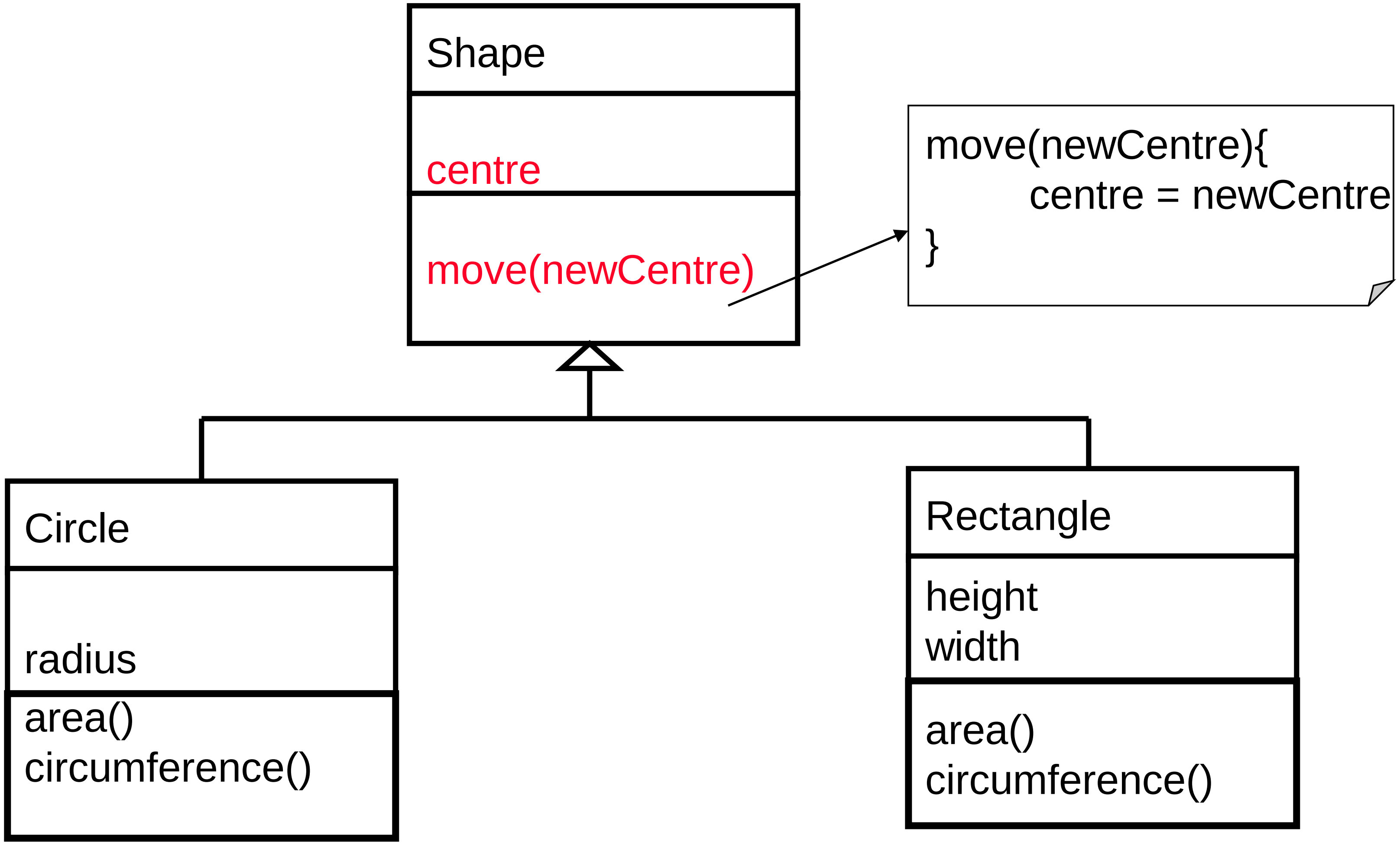


```
move(newCentre){
   centre = newCentre;
}
```

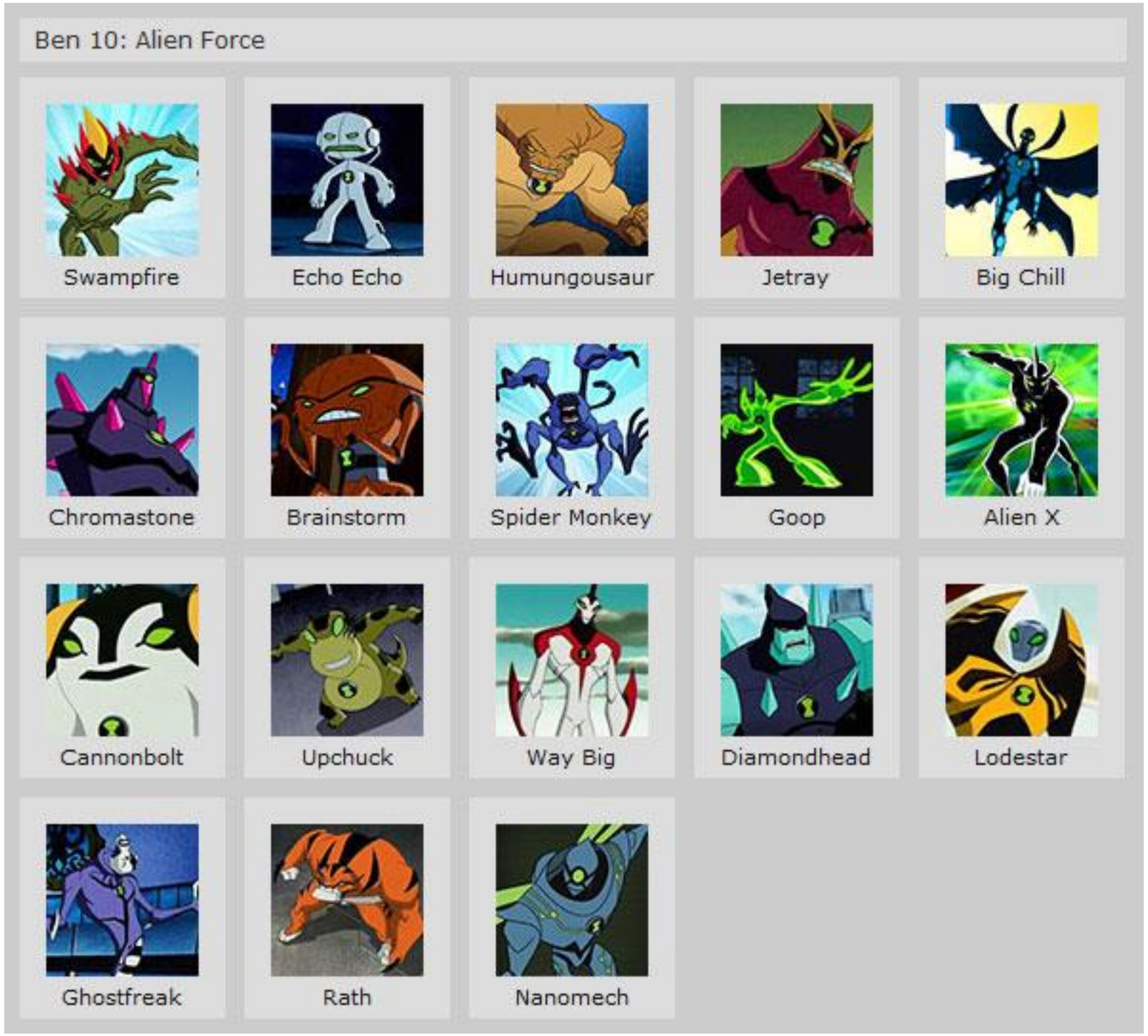


```
move(newCentre){
   centre = newCentre;
}
```

Reuse-Example



Polymorphism



Polymorphism

- Polymorphic which means “many forms” has Greek roots.
  - Poly – many
  - Morphos – forms

- In OO paradigm polymorphism has many forms.

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 input arguments.

```
initialize(2)     Method 1 will be called.
initialize(2,4)   Method 2 will be called.
```

innovate achieve lead

BITS Pilani
Pilani Campus

Thank You!