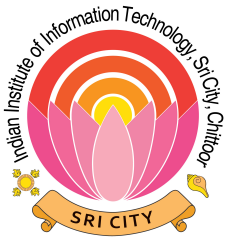


Monsoon 2019

Polymorphism

Object Oriented Programming

by

Dr. Rajendra Prasath

Indian Institute of Information Technology

Sri City – 517 646, Andhra Pradesh, India

Recap: Objects in JAVA ?

- An entity that has **state** and **behaviour** is known as an object
  - **Examples:** Chair, bike, marker, pen, table, car etc
  - It can be physical or logical
- An object has three characteristics:
  - **State:** represents data (value) of an object
  - **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw and so on
  - **Identity (Internally used):**
    - Signature (unique) of the object
    - Object identity is typically implemented via a unique ID
    - The value of the ID is not visible to the external user
    - But, Internally by JVM to identify each object uniquely

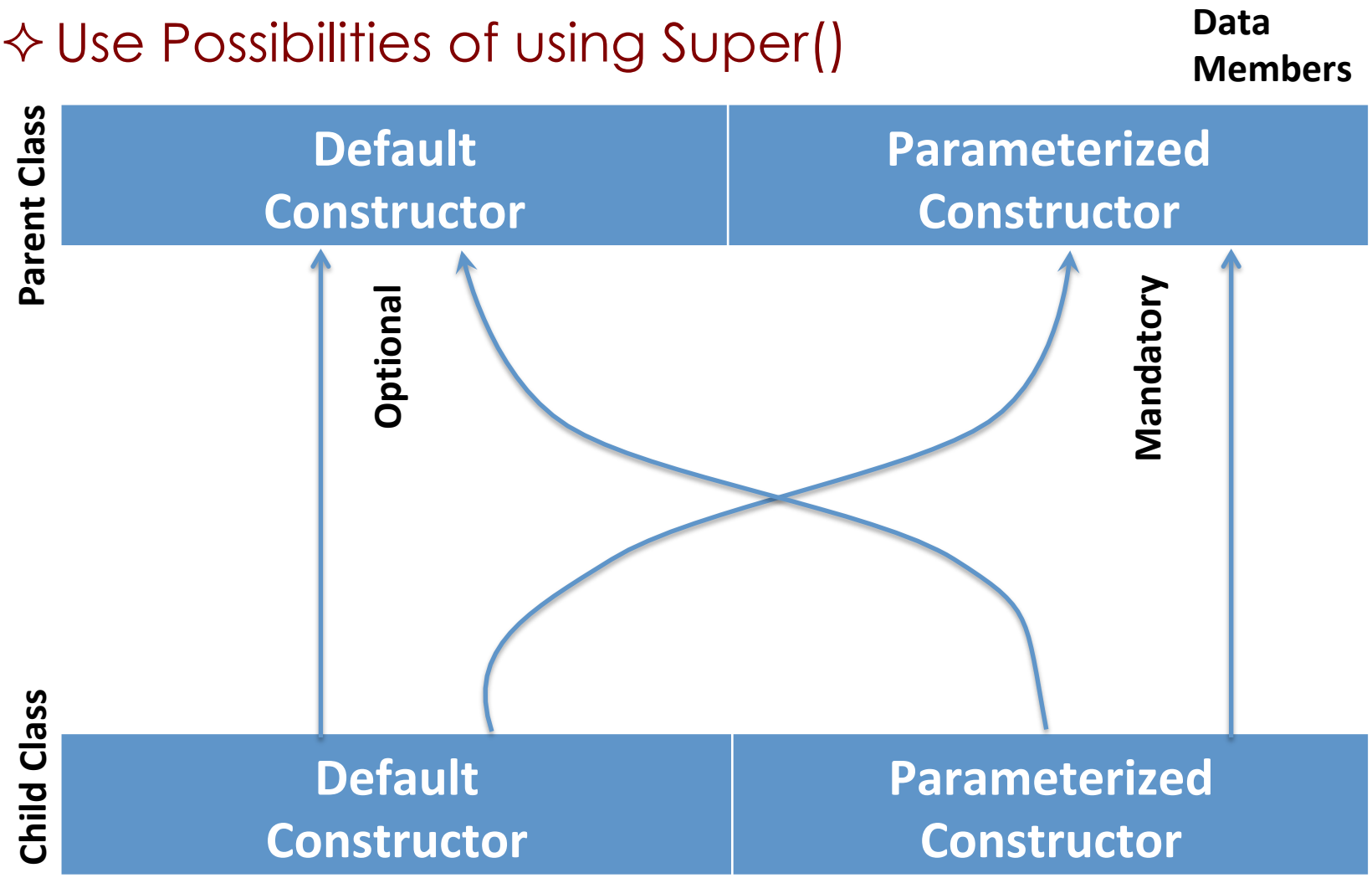
Recap: Inheritance & Constructors

- Constructor of the base with no arguments gets automatically called in the derived class constructor

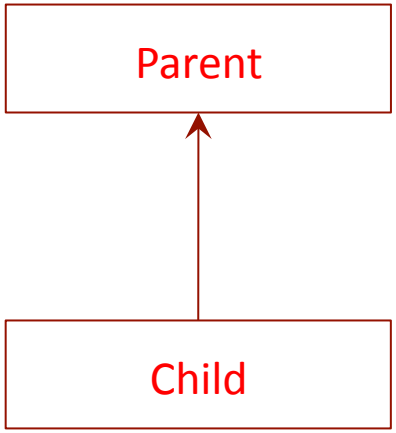
```
Class Parent {
        Parent() {
                System.out.println("Parent Class Constructor");
        }
}
Class Child extends Parent {
        Child() {
                System.out.println("Child Class Constructor");
        }
}
public class CodeTester {
        public static void main(String[] args) {
                Child child = new Child();
        }
}
```


Recap: Using Super class

✧ Use Possibilities of using Super()



Recap: Class Casting



✧ Let us look at the following:

Create an Object:

Parent p = new Parent();

Upcasting:

Parent p = new Child();

Downcasting:

Child c = new Parent() ➔ Compile time error

Child c = (Child) new Parent();

➔ This throws Class CastException at run time.

➔ How to solve this issue?

Downcasting (Instanceof)

✧ Example of Downcasting

```
class Parent { }

class Child extends Parent {
        static void doDowncast(Parent p) {
                if (p instanceof Child) {
                        Child c = (Child) p; // Downcasting
                        System.out.println("Downcasted!!");
                }
        }
        public static void main(String args[]) {
                Parent p = new Child();
                Child.doDowncast(p);
        }
}
```

Polymorphism

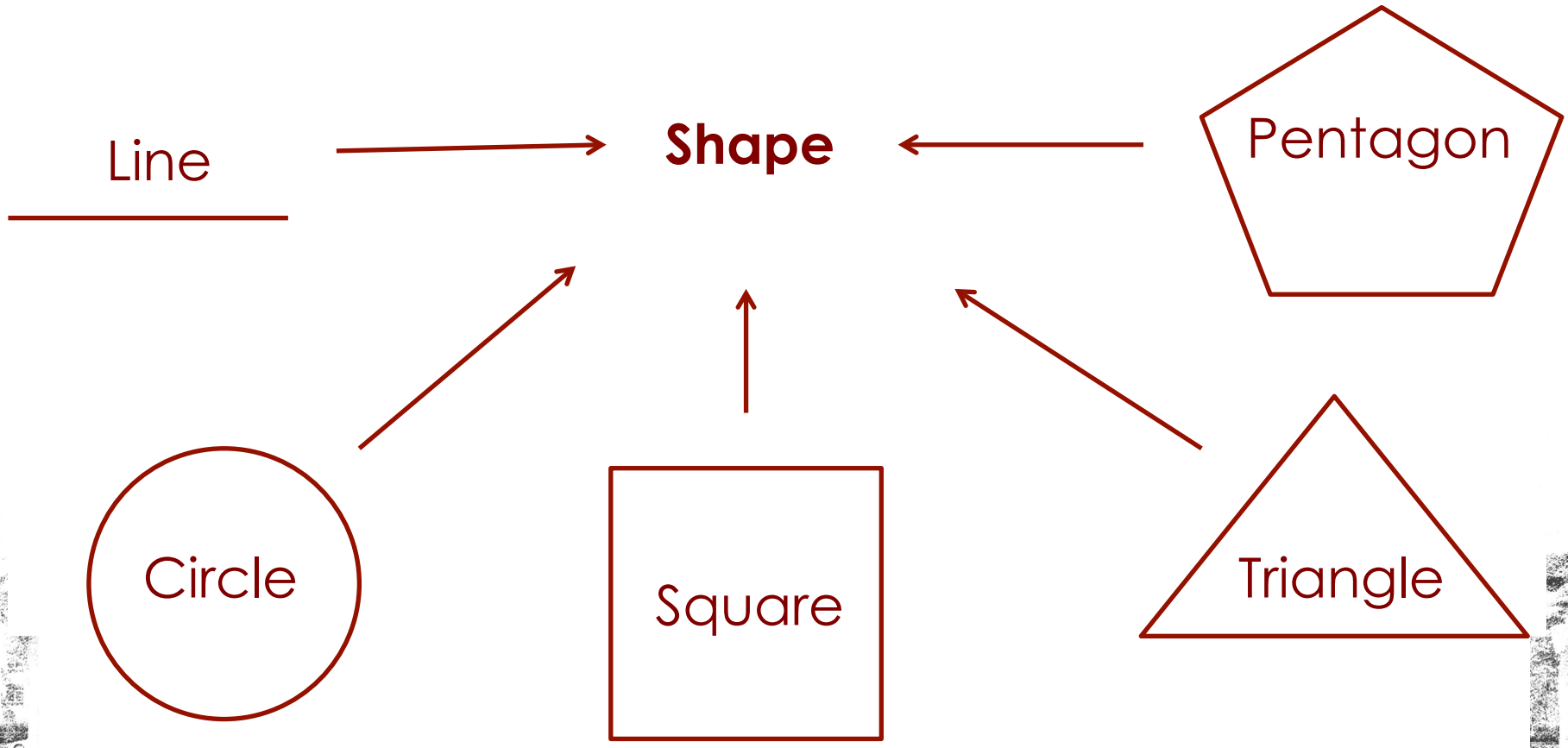
- Polymorphism in java is a concept by which we can perform a single action by different ways
- Polymorphism is derived from: **poly** and **morphs**
- The word "**poly**" means many and "**morphs**" means forms → So polymorphism means many forms

- **Example:**
  - Person in a class room – Faculty / Student
  - Person in market – Customer
  - Person at home – Son or Daughter
  - Person at Work – Founder / Employee

    → one person present in different behaviors

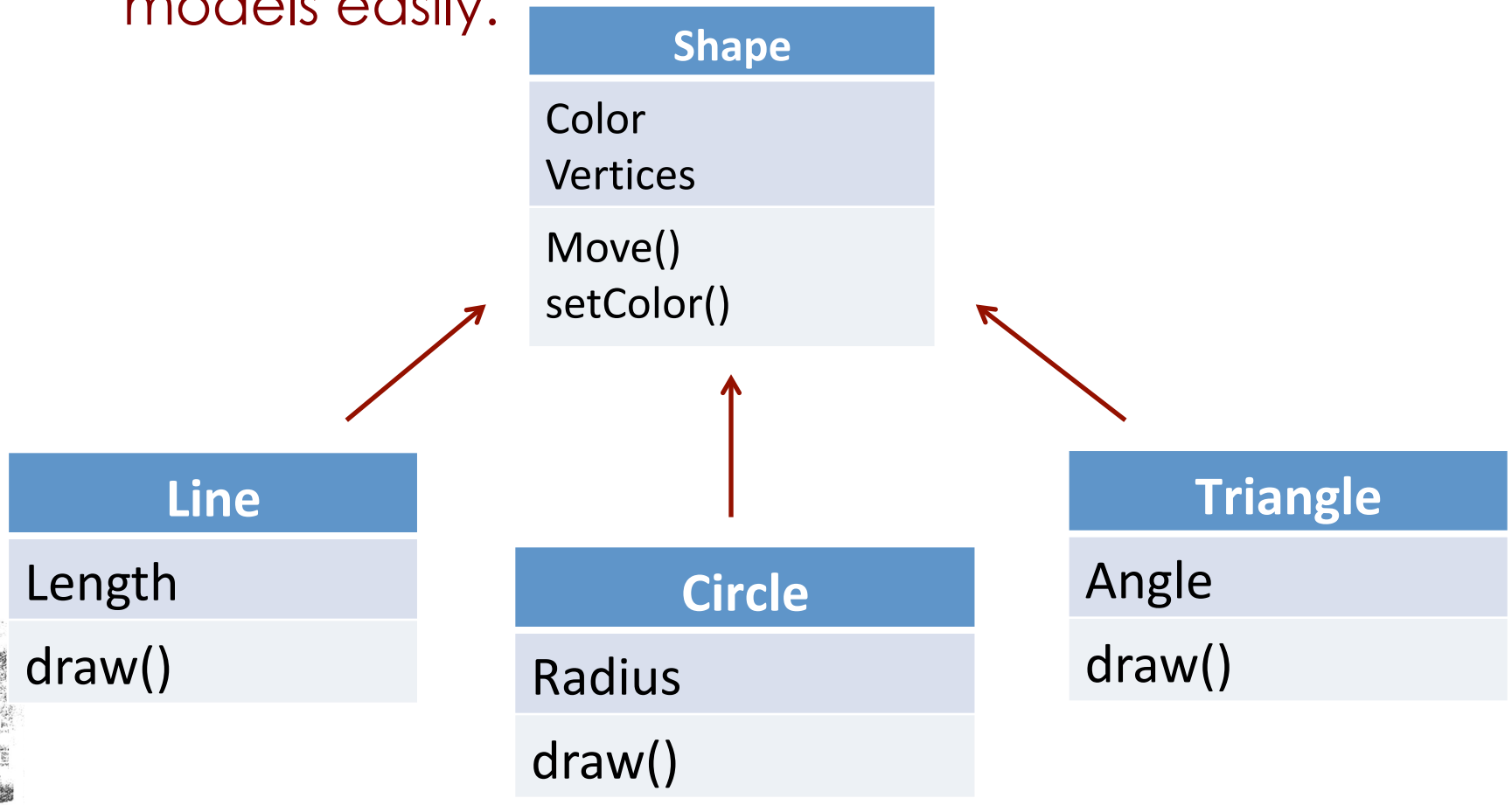
Polymorphism

✧ Let us look at example:



Class Diagrams

- New Classes can be added with the existing models easily.



Polymorphism in JAVA

- This can be achieved in Type ways:
  - Static or compile time polymorphism
    - Method Overloading
  - Dynamic or runtime polymorphism
    - Method Overriding
    - Virtual Methods

Example of Runtime polymorphism:

```
class Parent {}
class Child extends Parent {}
Parent p = new Child();          // Upcasting
```

Runtime Polymorphism - Example

✧ Runtime Polymorphism in JAVA

```
class Parent{
    void walk() {
        System.out.println("Speed: 20 kms!");
    }
}
class Child extends Parent {
    void walk() {
        System.out.println("Speed: 10 kms!");
    }
    public static void main(String args[]) {
        Parent p = new Child();          // Upcasting
        p.walk();
    }
}
```

Parent Reference and Child Object

Another Example

✧ Java Runtime Polymorphism with data Member

```
class Person{
        int salary=1000;
}
class Employee extends Person{
        int salary=2000;

        public static void main(String args[]){
                Person obj = new Employee();
                System.out.println("Salary is: "+obj.salary);
        }
}
```

Output:
Salary is: 1000

Non-Virtual Methods

```
class Animal{
        public void getAnimal(animal anim) {
                anim.ganim();
        }
        private void gAnim() {
                System.out.println("This is the Animal");
        }
}
class Cat extends Animal{
        public void gAnim() { System.out.println("This is a cat"); }
}
class Dog extends Animal{
        public void gAnim() { System.out.println("This is a Dog"); }
}
public class Test {
        public static void main(String[] args) {
                Animal a=new Animal();  Cat c=new Cat(); Dog b=new Dog();
                a.getAnimal(c); a.getAnimal(b);
        }
}
```

Virtual Method

```
class Animal{
        public void getAnimal(animal anim) {
                anim.ganim();
        }
        public void gAnim() {
                System.out.println("This is the Animal");
        }
}
class Cat extends Animal{
        public void gAnim() { System.out.println("This is a cat"); }
}
class Dog extends Animal{
        public void gAnim() { System.out.println("This is a Dog"); }
}
public class Test {
        public static void main(String[] args) {
                Animal a=new Animal();  Cat c=new Cat(); Dog b=new Dog();
                a.getAnimal(c); a.getAnimal(b);
        }
}
```

Exercise - 9

- **Perform Code Walkthrough of any one of the Apache projects, specifically the Nutch Framework**
  - Understand the workflow
  - Explore the design of framework
  - Find the subsystems that perform a specific task
  - Learn Pluggable subsystems of specific tasks

Assignments / Penalties

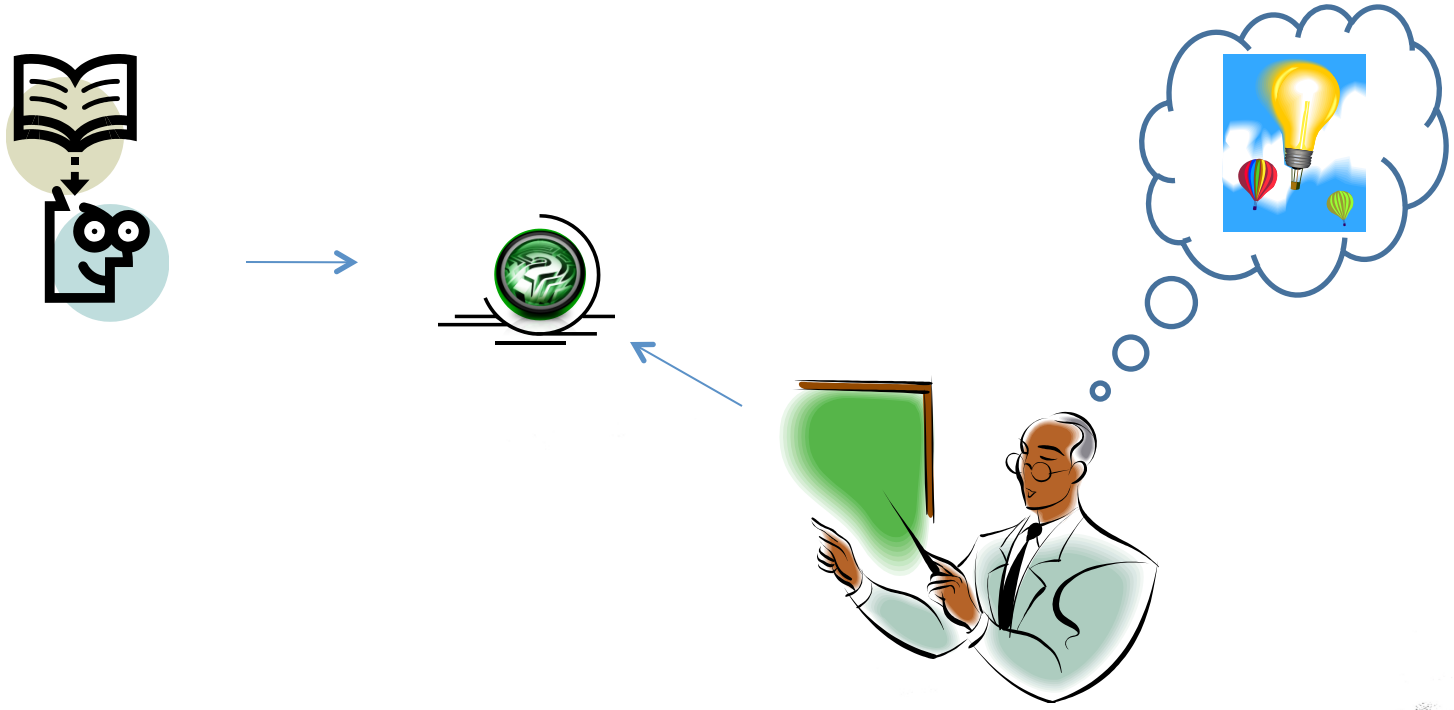
- Every Student is expected to complete the assignments and strictly follow a fair Academic Code of Conduct to avoid severe penalties

- Penalties would be heavy for those who involve in:
  - **Copy and Pasting** the code
  - **Plagiarism** (copied from your neighbor or friend – in this case, both will get "0" marks for that specific take home assignments)
  - If the candidate is **unable to explain his own solution**, it would be considered as a "copied case" !!
  - **Any other unfair means** of completing the assignments

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may leave me an email any time
  (email is the best way to reach me faster)

Thanks …



… Questions ???