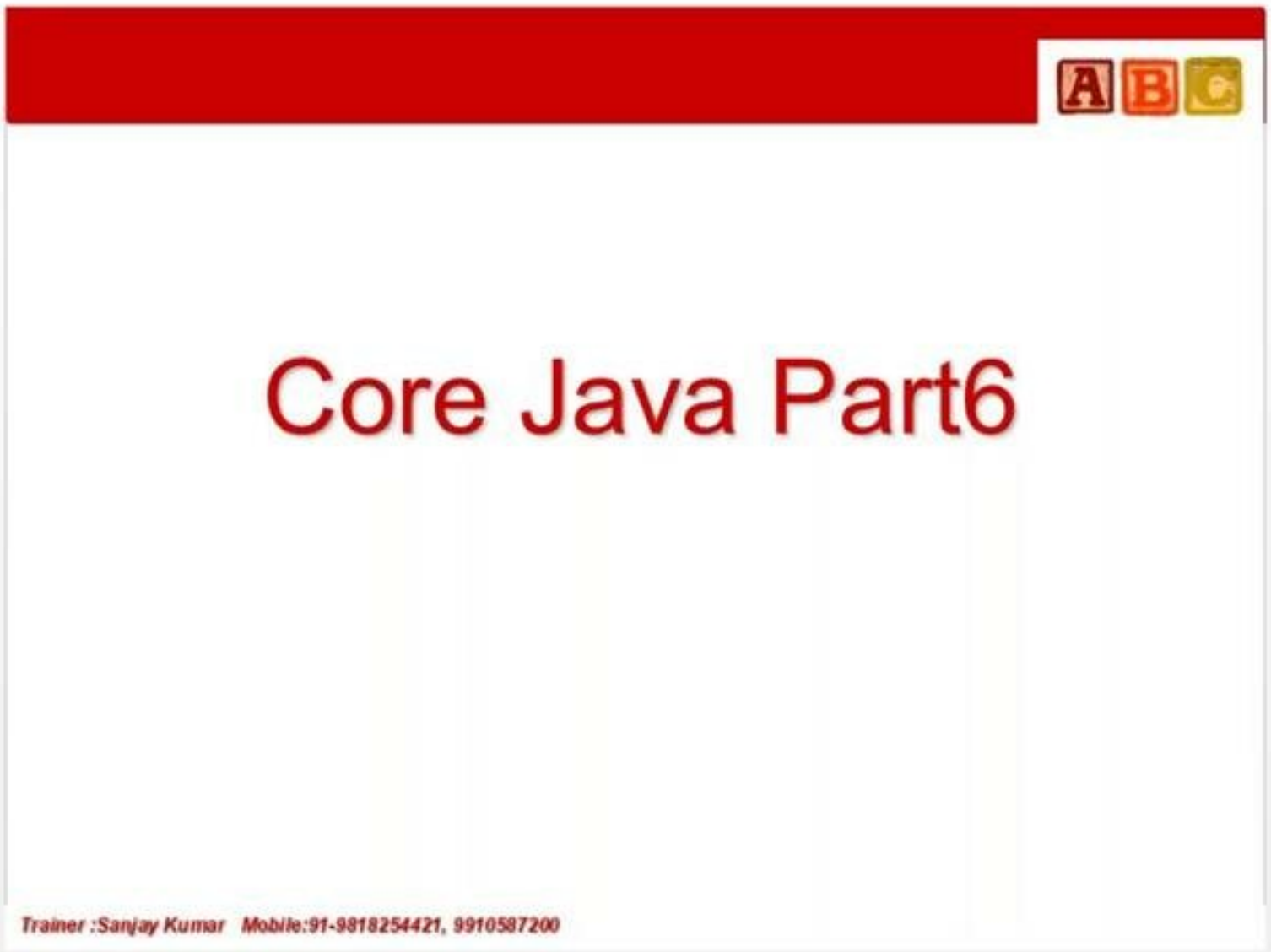

Kms
fac work
Classification of Foot The...
This PC
WhatsApp
BHAGAVAT...
Recycle Bin
Paadaigal-S...
BHAGAVAT... AADHAR
Adobe Reader XI
Spotify
GOKUL AADHAR
CCleaner
Irandam Ulagam(20...
MITHRAN
Firefox
An IOT based Alert Syste...
eclipse - Shortcut
Google Chrome
Irairvi (2016) Tamil 1080...
KMPlayer
Batch 29 Methodology
Microsoft Edge
COMMENT An IOT base...
Project
DXC Technology
Activate Windows
Go to Settings to activate Windows.
ENG IN
09:24
06-04-2022
22

Core_Java_part6 - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW

Sign in



Click to add notes

SLIDE 1 OF 32 ENGLISH (INDIA) NOTES COMMENTS 82%

Objectives

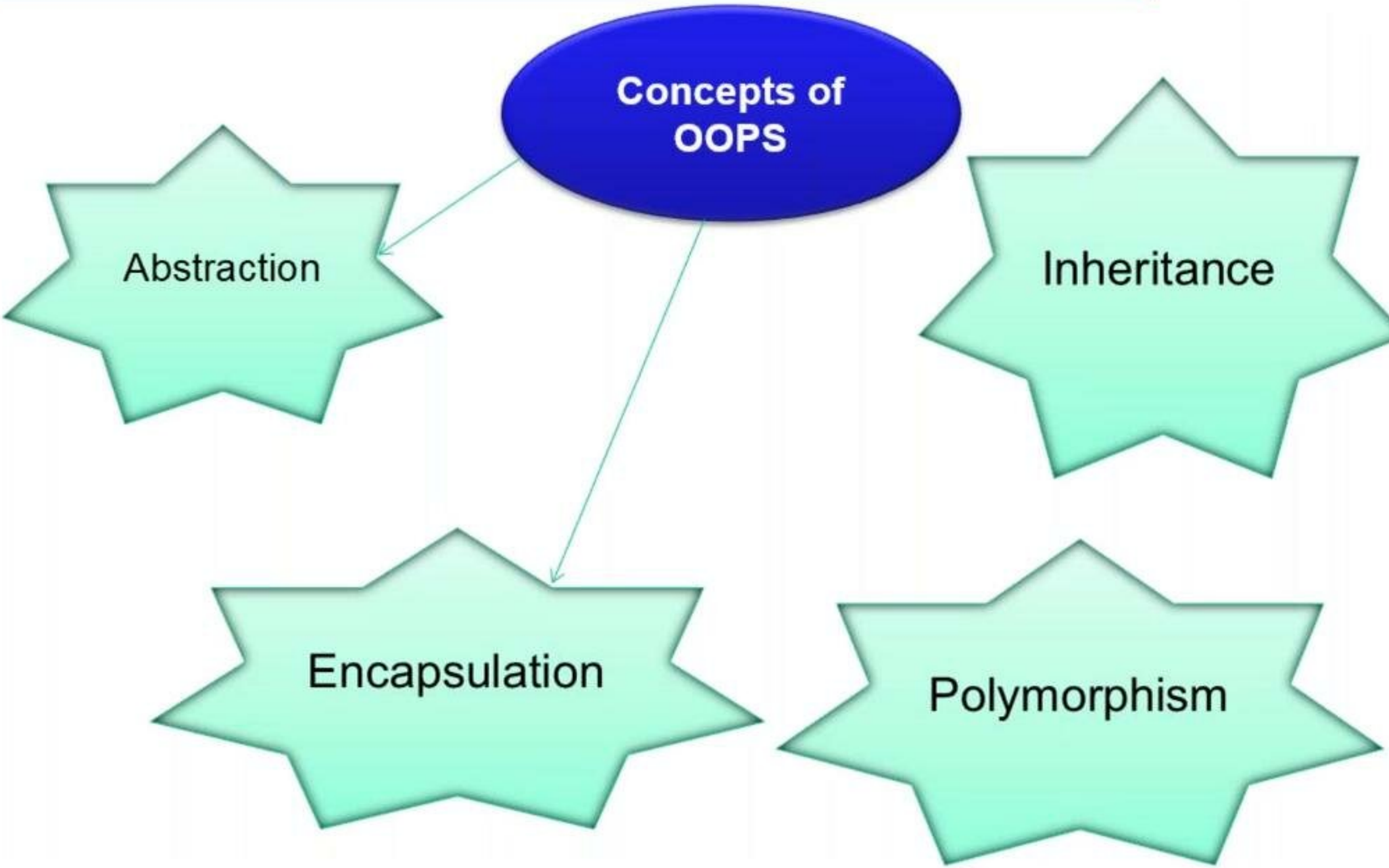
- Implementing OOP concepts
- Encapsulation
- Abstraction
- Constructors
  - Default Constructor
  - Parameterized Constructor
  - Copy Constructor
- Inheritance
  - Single Inheritance
  - Use of Super Keyword
  - Multilevel Inheritance
  - Hierarchical Inheritance
- Polymorphism
  - Compile Time Polymorphism
    - Method Overloading
  - Run Time Polymorphism
    - Method Overriding
- This Type Cast

Core Java Part6

Objectives

- Implementing OOP concepts
- Encapsulation
- Abstraction
- Constructors
  - Default Constructor
  - Parameterized Constructor
  - Copy Constructor
- Inheritance
  - Single Inheritance
  - Use of Super Keyword
  - Multilevel Inheritance
  - Hierarchical Inheritance
- Polymorphism
  - Compile Time Polymorphism
    - Method Overloading
  - Run Time Polymorphism
    - Method Overriding
- This Type Cast

Concepts Of OOPS



Encapsulation

- Information Hiding
- Hiding implementation details of an object from its user
- Packing things together and presenting them in their new integrated form.
- For example, two or more chemicals form a capsule
- Packing the methods and attributes together in a single unit.
- Units are implemented in the form of classes

"The process of putting attributes, methods or details of implementation in one unit ,is called *Encapsulation*."

Abstraction

Technique of dealing with the complexity of an object.

Focussing only on the essential details and overlooking the non-essential details of an object.

Data Abstraction

- Concept of abstraction applied to the attributes (data) of a class.

- Implemented in programming languages using Abstract Data Types (ADT).

"The process of identifying and grouping attributes and actions related to a particular entity as relevant to the application is *Data Abstraction*."

Constructors 7-1

Constructors 7-1

- Method invoked whenever an instance of a given class is created.
- Same name as the class and no return type.
- Java allocates memory for the object, initializes the instance variables and calls the constructor methods.
- Two types of constructors
  - Parameterized constructors
  - Implicit constructors

Derived Class Constructors 7-2

Base class constructor is invoked before the derived class

To invoke base class constructor super keyword is used

Constructor Inheritance

Derived class constructor has same name as that of the class

Derived Class Constructors 7-3

Derived Class Constructors 7-3

- The syntax to invoke the superclass constructor is:

```
super(parameter_list) or super();
```

- The parameters required by the constructor of the superclass are specified in the

parameter_list

Derived Class Constructors 7-4

- Whenever a subclass needs to refer to its immediate super class, use keyword super.
- If your method overrides one of its super class's methods, you can invoke the overridden method through the use of the keyword super.
- You can also use super to refer to a hidden field (although hiding fields is discouraged).
- It has advantage so that you don't to have to perform operations in the "parent class" again.
- Only the immediate "parent class's " data and methods can be accessed.
- Super has two general forms:
  - The first calls the super class ' constructor.
  - The second is used to access a member of the super class that has been hidden by a member of a super class.

Super Class Constructor 7 - 6

- If the super class does not have a no-argument constructor, you will get a compile - time error.

- Object does have such a constructor, so if Object is the only super class, there is no problem.
- If a sub class constructor invokes a constructor of its super class, either explicitly or implicitly, you might think that there will be a whole chain of constructors called, all the way back to the constructor of Object. In fact, this is the case. It is called constructor chaining, and you need to be aware of it when there is along line of class descent.

Inheritance

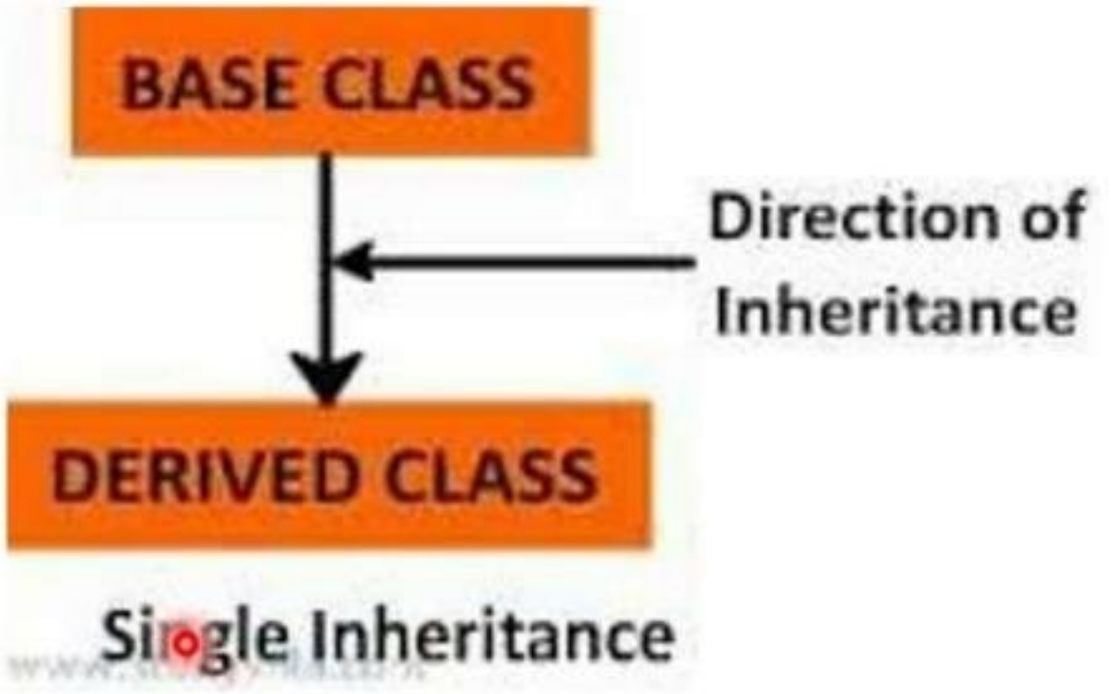
- Java Inheritance defines an is-a relationship between a superclass and its subclasses.

- This means that an object of a subclass can be used wherever an object of the superclass can be used.

- Class Inheritance in java mechanism is used to build new classes from existing classes.

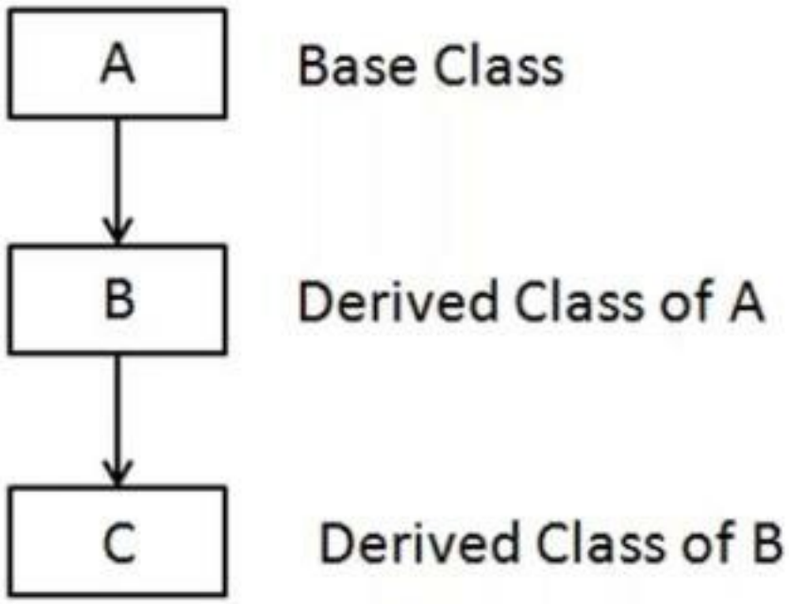
- if class x extends class y, then a class z, which extends class x, will also inherit class y.

Single Inheritance



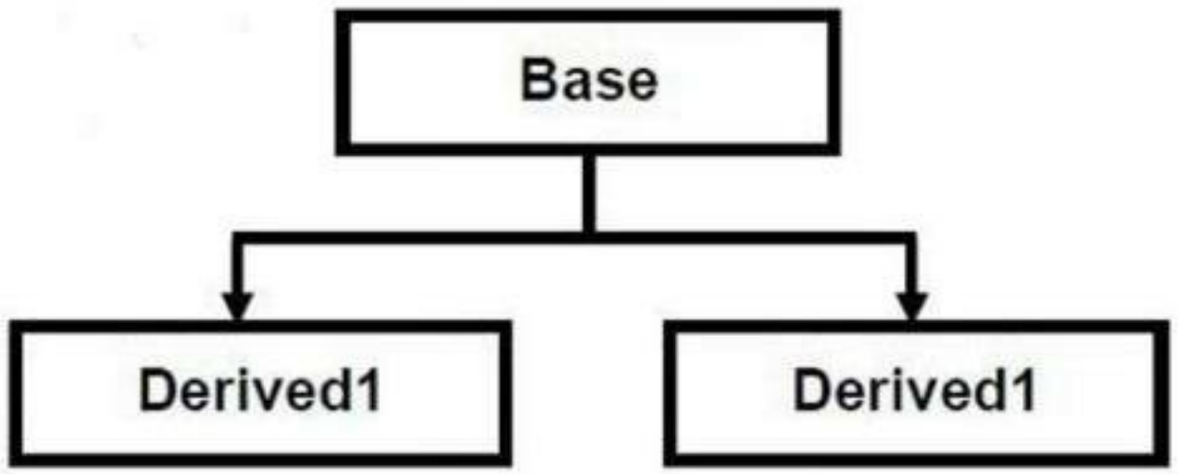
- Single Inheritance

Multilevel Inheritance



- Multilevel Iheritance

Hierarchical Inheritance



- Hierarchical Inheritance

Polymorphism

- Polymorphism means “many forms”.
- It is described as “one interface, many implementations”.
- It helps to write code that does not depend on specific types.
- It is the capability of a method to do different things based on the object that is acting upon it.
- The ability of a reference variable to change behavior according to what object instance it is holding.

Benefits of Polymorphism

- Simplicity
  - If you need to write code that deals with a family of types , the code can ignore type-specific details and just interact with the base type of the family
  - Even though the code thinks it is using an object of the base class, the object's class could actually be the base class or any one of its subclasses
  - This makes your code easier for you to write and easier for others to understand

- Extensibility
  - Other subclasses could be added later to the family of types, and objects of those new subclasses would also work with the existing code

Example #1: Polymorphism

- polymorphism is when we try to pass a reference to methods as a parameter

- Suppose we have a method printInformation that takes in a Person reference as parameter and other not.

```
public void printInformation( Person p )
{
    // It will call getName() method of the actual object instance that is passed
    p.getName();
}

public void printInformation( )
{
    // It will call getName() method of the actual object instance that is passed
    p.getName();
}
```

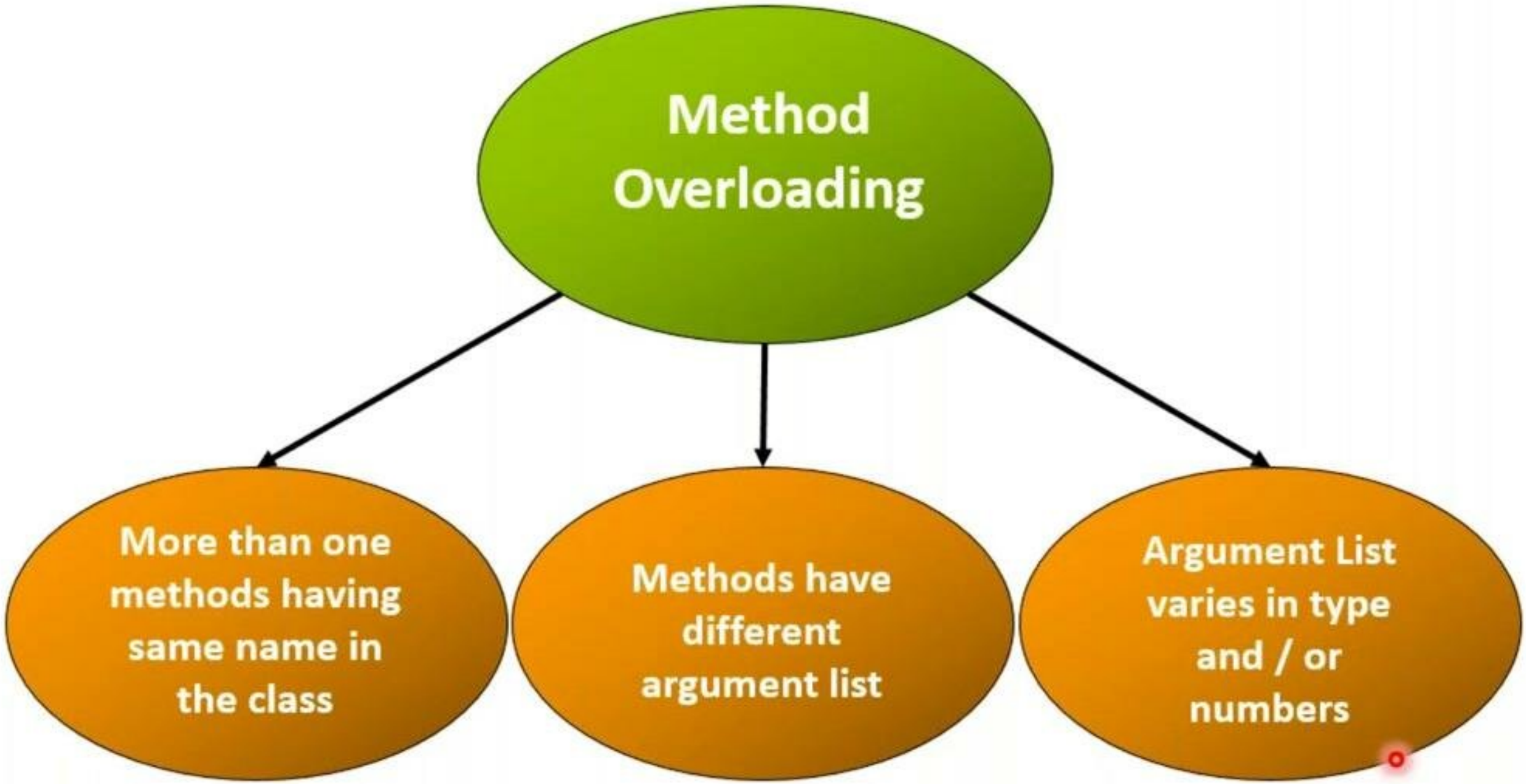
Forms of Polymorphism

- Method overloading
  – Methods in same class have many forms.
- Method overriding
  – Methods of a subclass override the methods of a base class
- Method overriding (implementation) of the abstract methods
  – Methods of a subclass implement the abstract methods of an abstract class
- Method overriding (implementation) through the Java interface
  – Methods of a concrete class implement the methods of the interface

Introduction to overloading

- Same name, different arguments.
- Code deals with different argument types rather than forcing the caller to do conversions prior to invoking your method.
- It CAN have a different return type.
- Argument list MUST be different.
- Access modifier CAN be different.
- New exceptions can be declared.
- A method can be overloaded in the *same* class.

Method Overloading 2-1



Method Overloading 2-2

- Usage of methods with same name.
- Argument lists varying in type and / or number.

```
void display(double count) {
    System.out.println("Inside the display(double) method :" + count);
}
```

```
public static void main(String [] arg) {
    int count = 25;
    DisplayNumber dispObj = new DisplayNumber();
    dispObj.display();
    dispObj.display(10 , 20);
    /* No method with a single argument of type int exists,
      int value is automatically promoted to double
    */
    System.out.println("Invoking the display(double) method"
     + "with an int variable: " + count);
    dispObj.display(count);
    dispObj.display(25.5);
}
```

Method accepting a `double` argument is invoked

Method Overriding 2-1