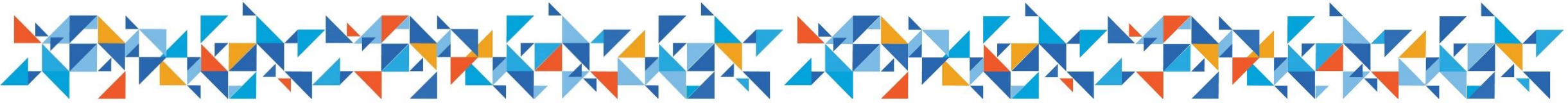
پوهنتون کاردان
KARDAN UNIVERSITY

Object Oriented Programming (Java)

Inheritance

Learning Outcomes

- Should know about What is inheritance?
- Should learn Inheritance Examples
- Understand Types of Inheritance
- What is IS-A relationship
- How to use Extends Keyword
- How to use Methods Overriding
- What is Super Keyword

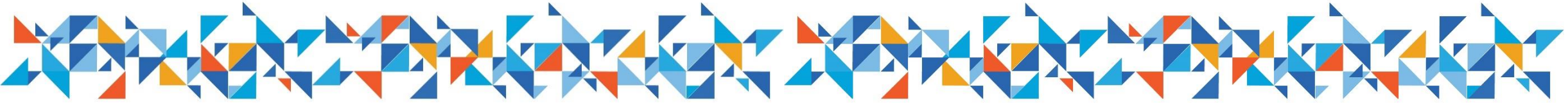


Inheritance

- The process by which one class acquires the properties(data members) and functionalities(methods) of another class is called **inheritance**.
- The aim of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from another class.

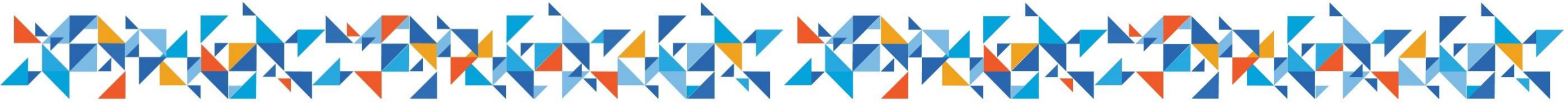
  Child Class:

  The class that extends the features of another class is known as child class, sub class or derived class.



Inheritance

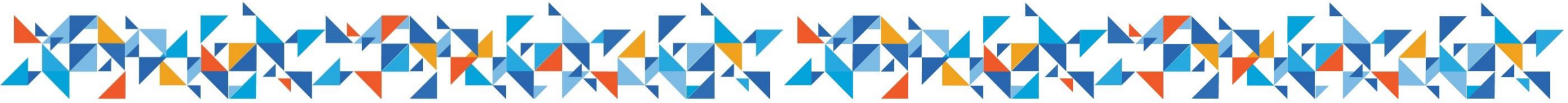
- **Parent Class:**
- The class whose properties and functionalities are used(inherited) by another class is known as **parent class, super class** or **Base class**.
- Inheritance is a process of defining a new class based on an existing class by extending its common data members and methods.
- Inheritance allows us to reuse of code, it improves reusability in your java application.
- **Note: The biggest advantage of Inheritance is that the code which is already present in base class need not be rewritten in the child class.**



Inheritance

- This means that the data members(instance variables) and methods of the parent class can be used in the child class as.
- **Syntax: Inheritance in Java**
- To inherit a class we use **extends** keyword.
- Here class XYZ is child class and class ABC is parent class. The class XYZ is inheriting the properties and methods of ABC class.

```
class XYZ extends ABC
{
}
```



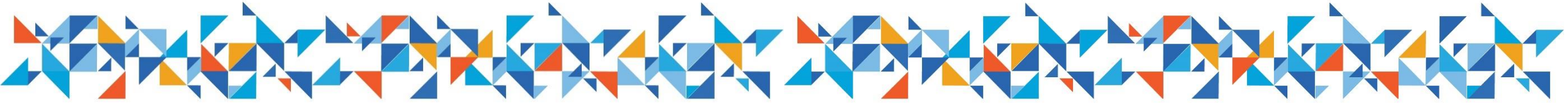
Inheritance Example

```
public class Teacher {
    String designation="Teacher";
    String collegename="Kardan International College";

    void does(){
        System.out.println("Teacher is teaching ");
    }
}

public class Physics_teacher extends Teacher {
    String subject = "Physics";

    public static void main(String[] args) {

        Physics_teacher obj=new Physics_teacher();
        System.out.println(obj.collegename);
        System.out.println(obj.designation);
        obj.does();
        System.out.println(obj.subject);
    }
}
```

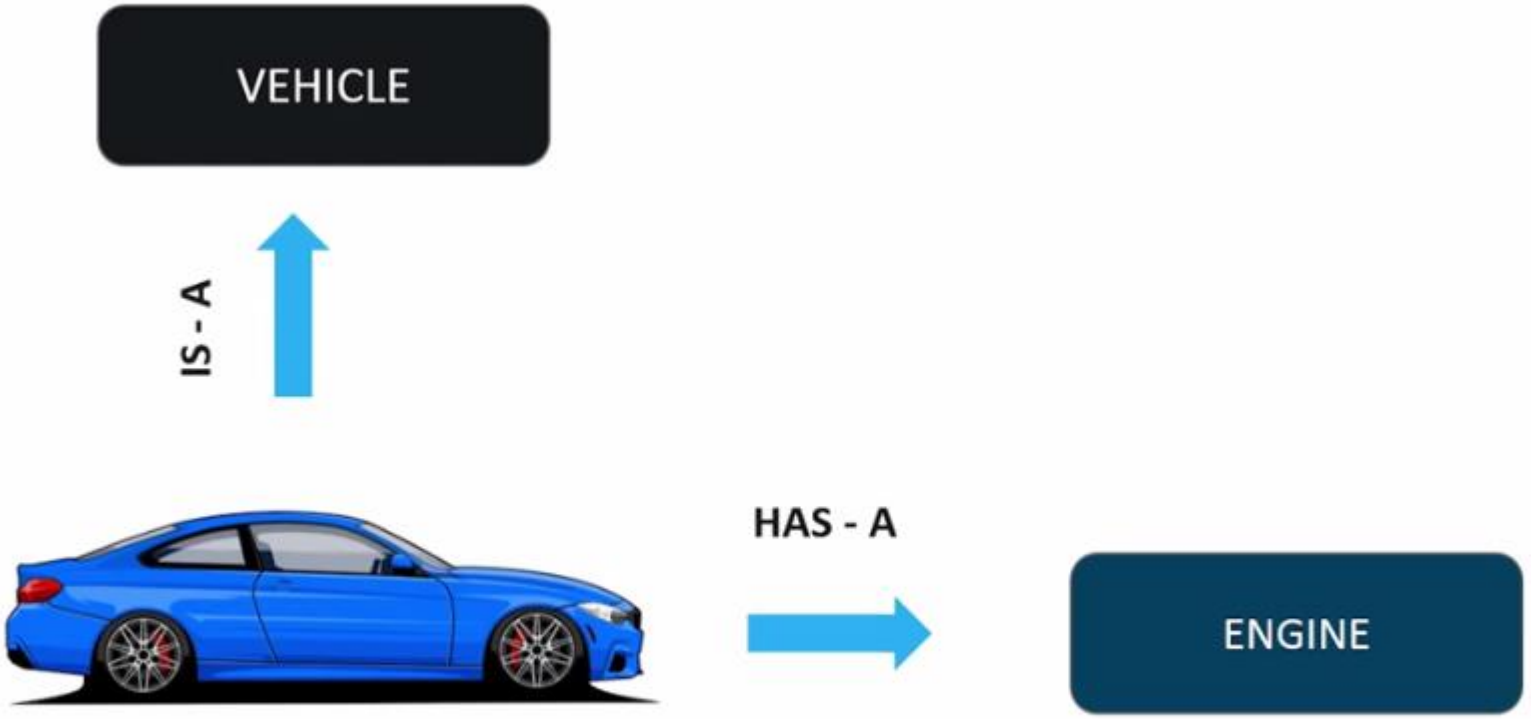
Based on the above example we can say that PhysicsTeacher IS-A Teacher. This means that a child class has IS-A relationship with the parent class. This inheritance is known as IS-A relationship between child and parent class

Inheritance Example

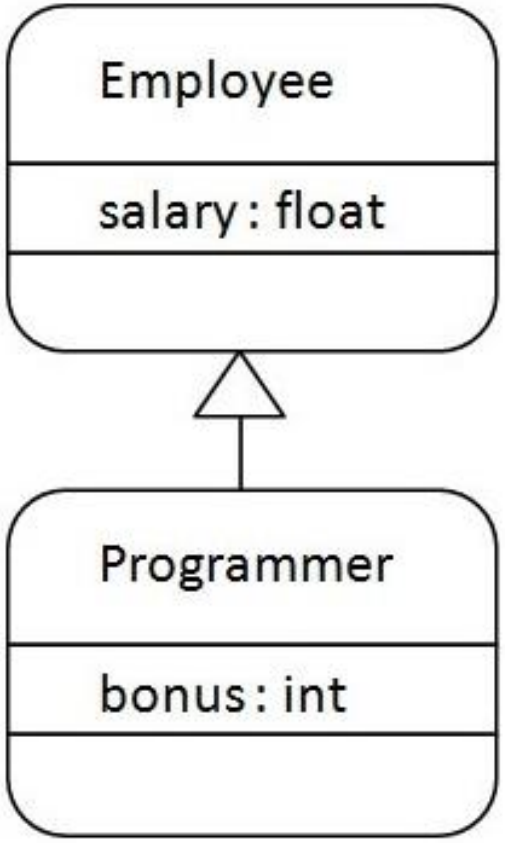
- In this example, we have a base class **Teacher** and a sub class **PhysicsTeacher**. Since class PhysicsTeacher extends the designation and college properties and **work**() method from base class, we do not need to declare these properties and method in sub class.
- Here we have collegeName, designation and work() method which are common to all the teachers so we have declared them in the base class, this way the child classes like MathTeacher, MusicTeacher and PhysicsTeacher do not need to write this code and can be used directly from base class.



IS-A and Has-A relationship



Java Inheritance Example



- As displayed in the above figure, Programmer is the subclass and Employee is the superclass.
- The relationship between the two classes is **Programmer IS-A Employee**.
- It means that Programmer is a type of Employee.

```
public class Employee{
 float salary=40000.0f;
}

class Programmer extends Employee{
 int bonus=10000;

 public static void main(String args[]){
    Programmer p=new Programmer();
    System.out.println("Programmer salary is:"+p.salary);
    System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

Types of Inheritance in Java

- There are 5 types of inheritance in Java programming.

➢ The first 3 types of inheritance are supported by Java and the last 2 types are not supported by Java programming language.

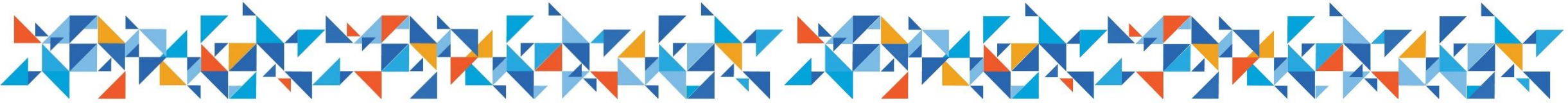
1. Single Inheritance

2. Multilevel Inheritance

3. Hierarchical Inheritance

4. Multiple inheritance

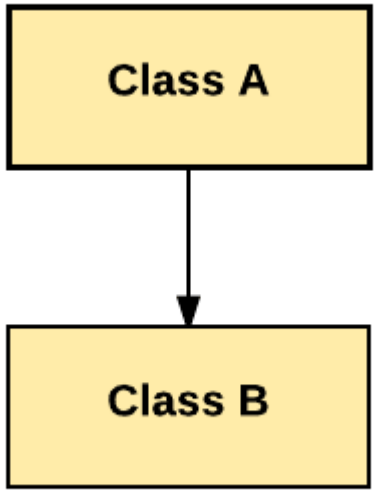
5. Hybrid inheritance



Single Inheritance

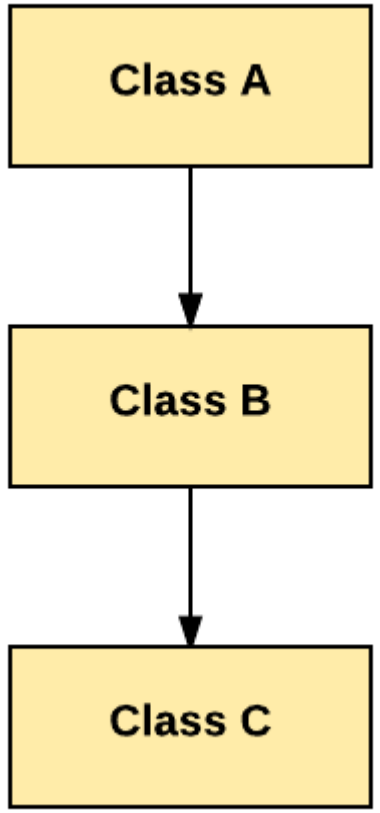
In Single Inheritance one class extends another class (one class only).

In above diagram, Class B extends only Class A. Class A is a super class and Class B is a Sub-class.



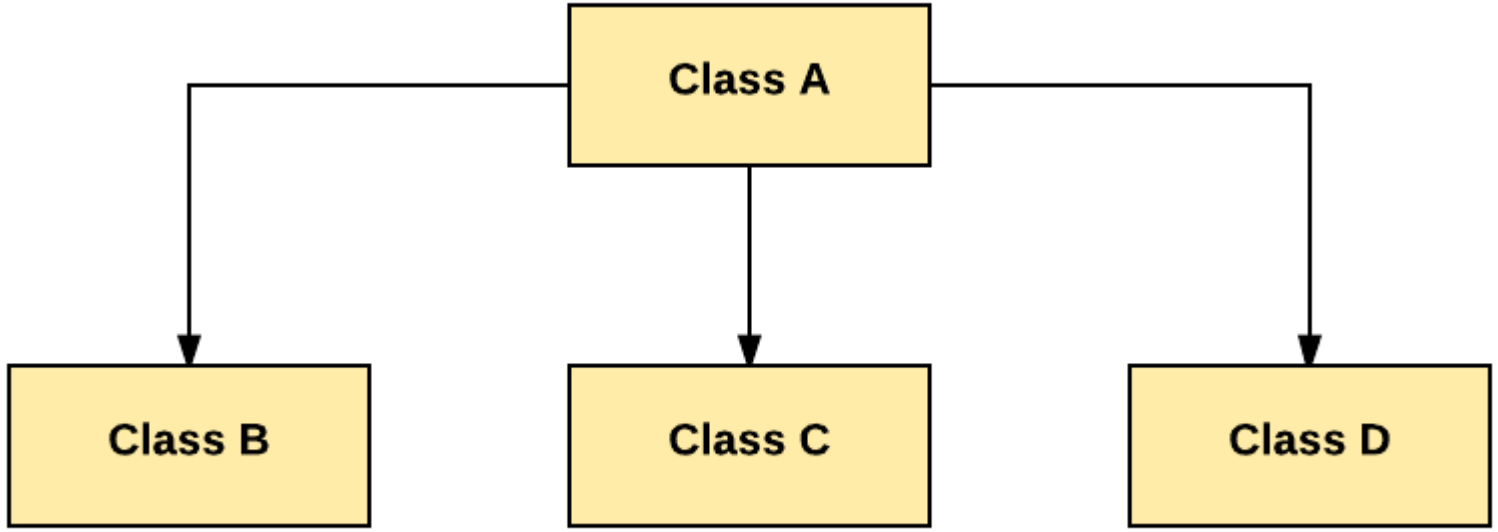
Multilevel Inheritance

- In Multilevel Inheritance, one class can inherit from a derived class. Hence, the derived class becomes the base class for the new class.
- As per shown in diagram Class C is subclass of B and B is a of subclass Class A.
-



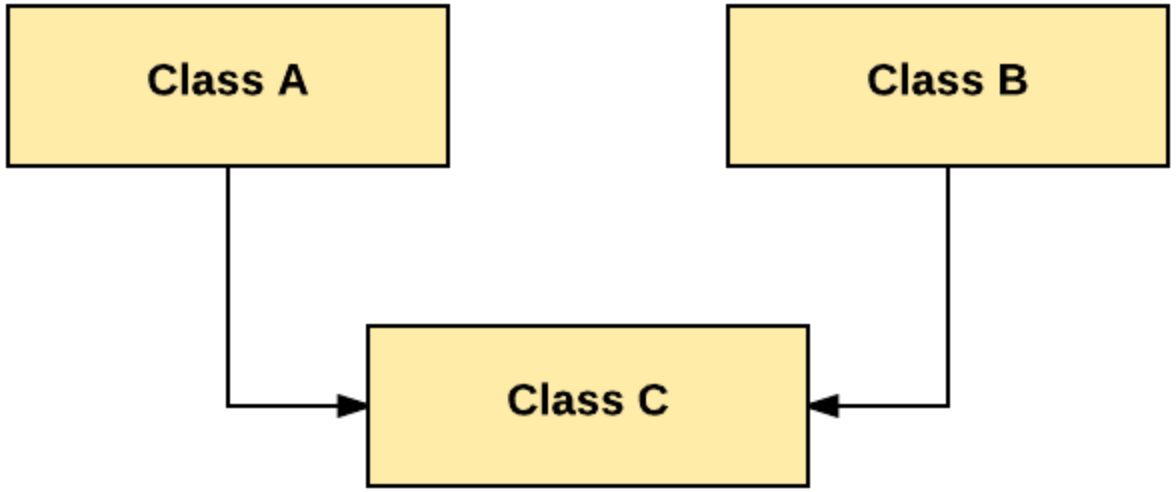
Hierarchical Inheritance

- In Hierarchical Inheritance, one class is inherited by many sub classes.
- As per example, Class B, C, and D inherit the same class A.



Multiple Inheritance

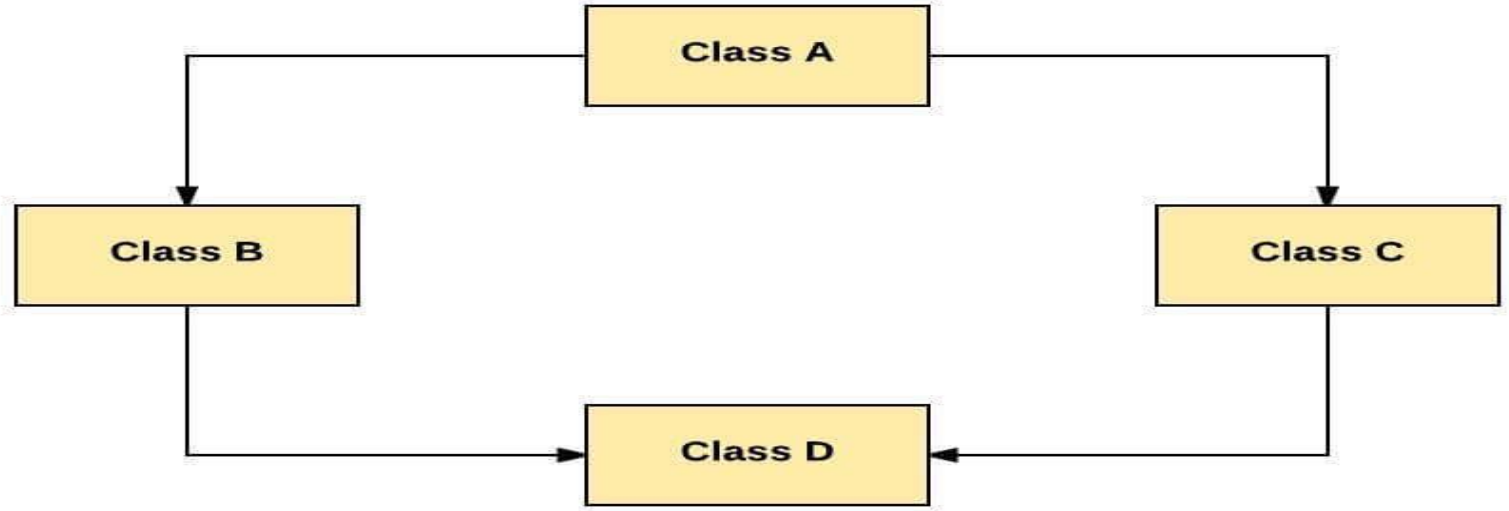
- In Multiple Inheritance, one class extending more than one class. Java does not support multiple inheritance.



- As per above diagram, Class C extends Class A and Class B both.

Hybrid Inheritance

- Hybrid inheritance is a combination of Single and Multiple inheritance
- As per above example, all the public and protected members of Class A are inherited into Class D, first via Class B and secondly via Class C.
- **Note:** Java doesn't support hybrid/Multiple inheritance.



Single Inheritance Example

In the example given below, Dog class inherits the Animal class, so there is the single inheritance.

```
public class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

Multilevel Inheritance Example

- In the example given below, BabyDog class inherits the Dog class which again inherits the Animal class, so there is a multilevel inheritance.

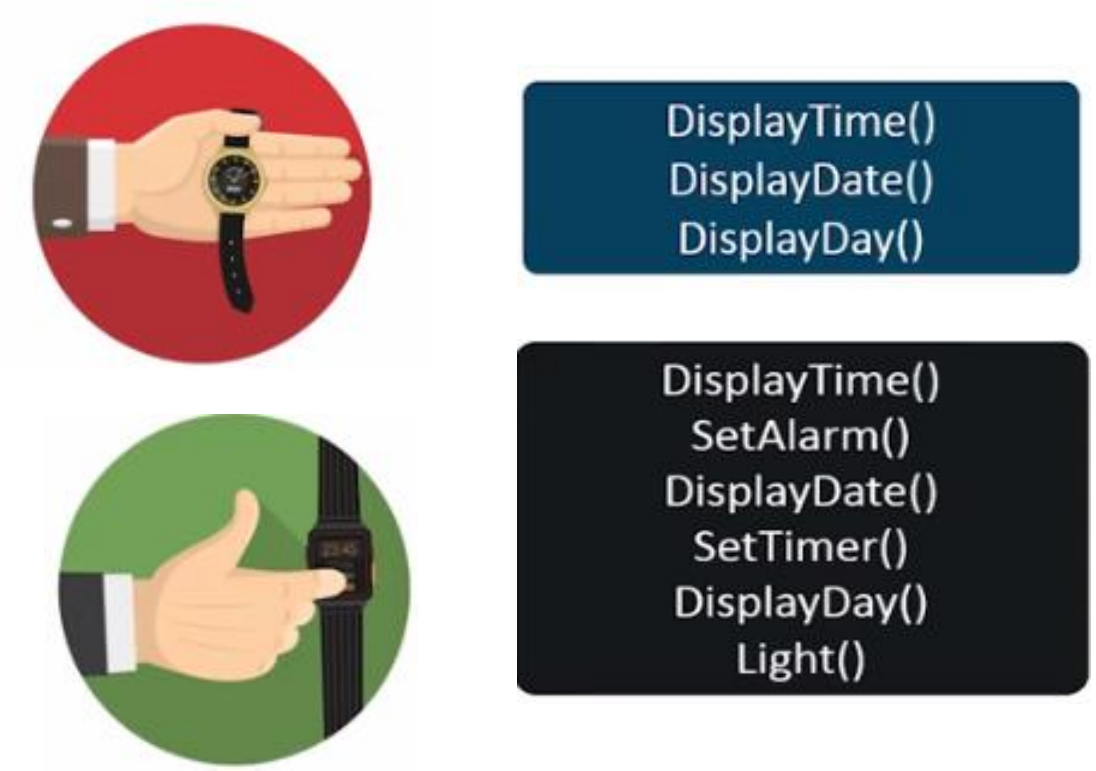
```
public class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

Hierarchical Inheritance Example

- In the example given below, Dog and Cat classes inherits the Animal class, so there is hierarchical inheritance.

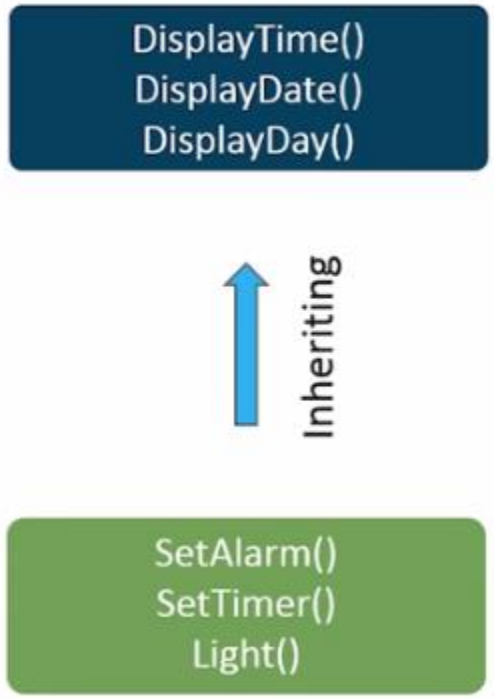
```
public class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

- In inheritance, the child class is called subclass and the parent class is called super class.
- “Watch” is parent class
- “Digital watch” is child class



Contd.

- With inheritance, you can save time during program development by basing new classes on existing proven and debugged high-quality software.



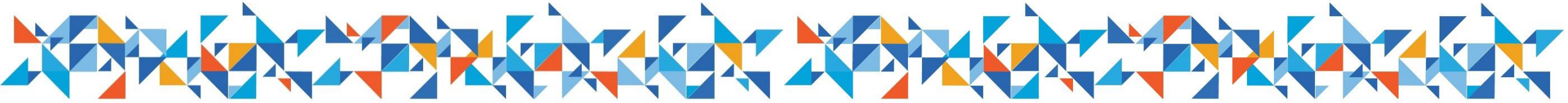
Extends

- **extends** is the keyword used to inherit the properties of a class.

- **Following is the syntax of extends keyword:**

- **public class** Subclass **extends** Superclass

```
{
  //methods and fields
}
```



```java
public class Watch{
    protected int second,minute,hour;

    protected void setTime(int h, int min, int sec){
        this.hour=h;
        this.minute=min;
        this.second=sec;
    }
    protected void showTime(){
        System.out.println("Time is: "+hour+" : "+minute+" : "+second);

    }
}
class DigitalWatch extends Watch{
    protected int day;
    protected String month;
}
class Main_{
    public static void main(String[] args) {
        DigitalWatch obj=new DigitalWatch();
        obj.setTime(12,15,33);
        obj.showTime();

        obj.day=12;
        System.out.println("Day is: "+obj.day);
        obj.month="Auhust";
        System.out.println("Month is : "+obj.month);
    }
```

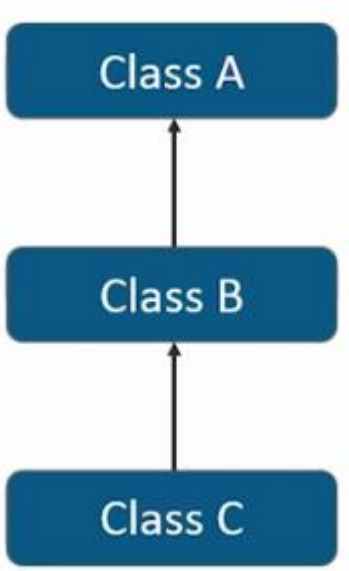
Multilevel inheritance

- Multilevel inheritance happens when a class is getting inherited from another class which is already having a superclass.

```
2 public class Watch {
```

```
2 public class DigitalWatch extends Watch{
```

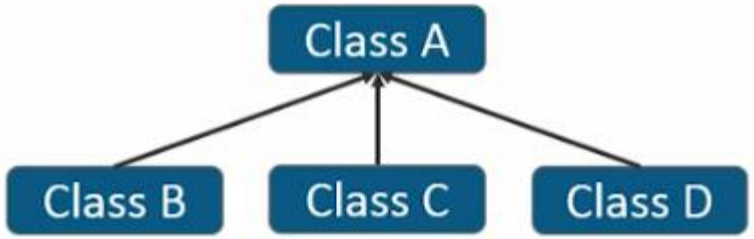
```
2 public class AnalogDigital extends DigitalWatch {
```



Hierarchical Inheritance

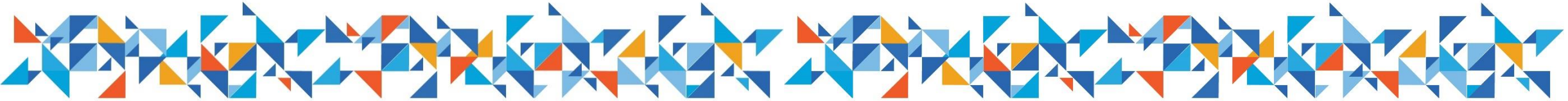
- Hierarchical inheritance happens when multiple classes getting inherited from a single class.

```
2 public class Watch {
```



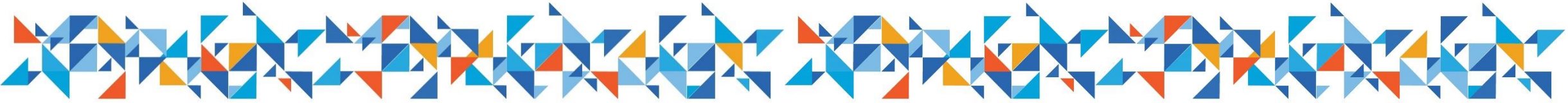

```
public class Rulex extends Watch{
```

```
public class Quartz extends Watch{
```



Inheritance and Method Overriding

- When we declare the same method in child class which is already present in the parent class this is called method overriding.
- In this case when we call the method from child class object, the child class version of the method is called.
- However we can call the parent class method using super keyword as shown in the example below:



```
public class Vehicle{
    void run(){
        System.out.println("Vehicle is runnning");
    }
}
class Bike extends Vehicle{
    void run(){
        System.out.println("Bike can run faster!");
                    super.run();

    }
}
class Check{
    public static void main(String[] args) {
        Bike obj=new Bike();
        obj.run();
    }
}
```

پوهنتون کاردان
KARDAN UNIVERSITY

Thank You…!