OBJECT ORIENTED PROGRAMMING USING JAVA

OUTLINE

- INHERITANCE IN JAVA

INHERITANCE IN JAVA

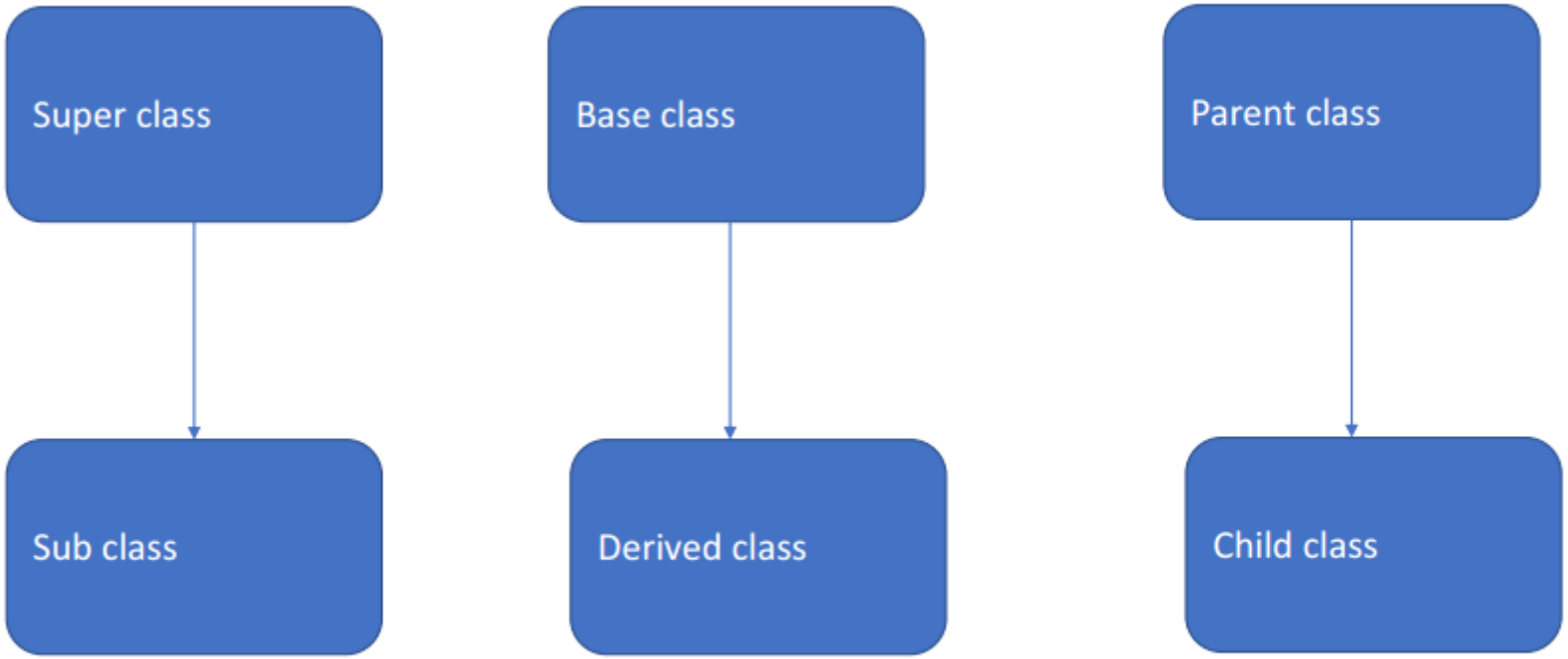
- Inheritance in Java is a mechanism in which one object acquires all the`properties and behaviours of a **parent object**. It is an important part of OOPs (Object Oriented programming system).

- The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.
- When you inherit from an existing class, you can reuse methods and fields of the parent class.
- Moreover, you can add new methods and fields in your current class also.

TERMS USED IN INHERITANCE

- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

TERMINOLOGY



THE SYNTAX OF JAVA INHERITANCE

```
class Subclass-name extends Superclass-name
  {
  //methods and fields
  }
```

- The **extends** keyword indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality .

EXAMPLE:

```
class p
{
public void m1()
{
System.out.println("parent");
}
}

class c extends p
{
public void m2()
{
System.out.println("child");
}
}
```

Note: Whatever member/method parent has, it is available for the child.

PREDICT THE OUTPUT: CASE 1

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}


public class Main
{
    public static void main(String[] args) {
        P p= new P();
        p.m1();

    }
}
```

PREDICT THE OUTPUT: CASE 1

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}


public class Main
{
    public static void main(String[] args) {
        P p= new P();
        p.m1();

    }
}
```

Output:
parent

PREDICT THE OUTPUT: CASE 2

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}

public class Main
{
    public static void main(String[] args) {
        P p= new P();
        p.m1();
        p.m2();

    }
}
```

PREDICT THE OUTPUT: CASE 2

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}

public class Main
{
    public static void main(String[] args) {
        P p= new P();
        p.m1();
        p.m2();

    }
}
```

Output:

```
Main.java:30: error: cannot find symbol
            p.m2();
             ^
  symbol:   method m2()
  location: variable p of type P
1 error
```

PREDICT THE OUTPUT: CASE 3

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}

public class Main
{
    public static void main(String[] args) {
        C c=new C();
        c.m1();
        c.m2();
    }
}
```

PREDICT THE OUTPUT: CASE 3

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}

public class Main
{
    public static void main(String[] args) {
        C c=new C();
        c.m1();
        c.m2();
    }
}
```

Output

```
parent
child
```

PREDICT THE OUTPUT: CASE 4

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}

public class Main
{
    public static void main(String[] args) {
       P p=new C(); /// parent reference child object
       p.m1();
       p.m2();

    }
}
```

PREDICT THE OUTPUT: CASE 4

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}

public class Main
{
    public static void main(String[] args) {
       P p=new C(); /// parent reference child object
       p.m1();
       p.m2();

    }
}
```

Output:

```
Main.java:30: error: cannot find symbol
        p.m2();
         ^
  symbol:   method m2()
  location: variable p of type P
1 error
```

Note:

- **Parent reference can be used to hold child object, but by this call we cannot call child specific methods. We can only call parent method.**
- **Its advantage is in the polymorphism.**

PREDICT THE OUTPUT: CASE 5

```
class P
{
public void m1()
{
System.out.println("parent");
}
}

class C extends P
{
public void m2()
{
System.out.println("child");
}
}

public class Main
{
    public static void main(String[] args) {
       C c=new P();  /// Child reference parent object
       c.m1();
       c.m2();

    }
}
```

Output:

```
Main.java:28: error: incompatible types: P cannot be converted to C
           C c=new P();  /// Child reference parent object
                 ^
1 error
```

Note: Child reference can not hold parent object

WHY USE INHERITANCE IN JAVA?

- For Code Reusability.
- For Method Overriding (so runtime polymorphism can be achieved)

ADVANTAGE OF INHERITANCE: EXAMPLE

- I want to develop loan module for housing loan, vehicle loan, home loan.
- Case 1: (without inheritance)

```
Class H_loan
{
300 methods
}
```

```
Class V_loan
{
300 methods
}
```

```
Class  P_loan
{
300 methods
}
```

Hence, for this example it required 900 method, to develop entire loan module

- **Lets assume on an average 1 method requires 5 minutes**
- **Therefore to develop it requires 900*5= 4500 minutes= 75 hours**
- **Redundancy is high**

ADVANTAGE OF INHERITANCE: EXAMPLE

- I want to develop loan module for housing loan, vehicle loan, home loan.
- Case 1: (with inheritance)

```
Class Loan
{
200 methods
}
```

```
Class H_loan extends Loan
{
100 methods
}
```

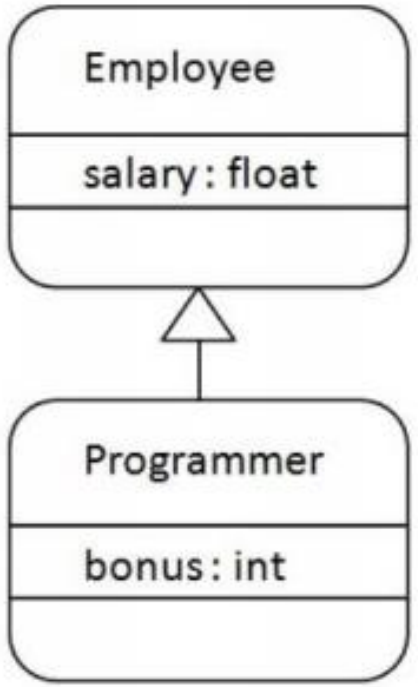
```
Class V_loan extends Loan
{
100 methods
}
```

```
Class  P_loan extends Loan
{
100 methods
}
```

- All of them are loan, hence First find out the common methods , lets assume 200 methods are common. Therefore create a Loan class and then preform inheritance.
- Therefore to develop it requires 500*5= 2500 minutes= 41.67 hours
- Less code less development time (Code reusability, less development time).

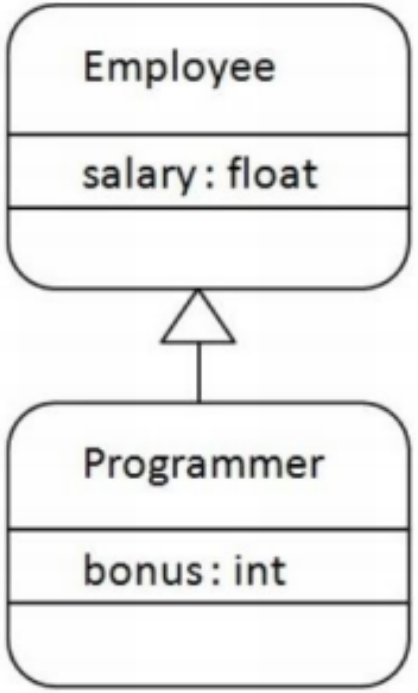
JAVA INHERITANCE EXAMPLE

- Inheritance represents the IS-A relationship which is also known as a parent-child relationship.
- As displayed in the figure, Programmer is the subclass and Employee is the superclass. The relationship between the two classes is Programmer ISA Employee. It means that Programmer is a type of Employee.



JAVA INHERITANCE EXAMPLE

```
class Employee
{
 float salary=40000;
}
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);
}
}
```



o/p:
Programmer salary is 40000.00
Bonus of Programmer is 10000

IDEA OF INHERITANCE

- Inheritance in java is a mechanism in which child class object acquires all the properties and behaviours of parent class object.
- The idea behind inheritance in java is that you can create new classes that are built upon existing classes.
- When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.

PREDICT THE OUTPUT:

```
class addition{  //super class
        int a=10;
        int b=5;
       }   //endof super class

class Inheritance extends addition {        //sub class
         int c;
        void add()
            {c=a+b;
            System.out.println("the sum ="+c);}}

public class Main
{
    public static void main(String[] args) {
      Inheritance x= new Inheritance();     //sub class object
    x.add();
    }
}
```

PREDICT THE OUTPUT:

```
class addition{  //super class
        int a=10;
        int b=5;
       }   //endof super class

class Inheritance extends addition {        //sub class
         int c;
        void add()
            {c=a+b;
            System.out.println("the sum ="+c);}}

public class Main
{
    public static void main(String[] args) {
      Inheritance x= new Inheritance();     //sub class object
    x.add();
    }
}
```

Output:

the sum is =15

SAME NAME VARIABLE OR METHOD

```
class eval{  //super class
        int a=10;
        int b=5;
        int c;
        void cal()
        {
            c=a-b;
           System.out.println("parent output is ="+c);

        }
       }   //endof super class

class Inheritance extends eval {        //sub class
         int a=100,c;
        void cal()
            {c=a+b;
            System.out.println("child output is ="+c);}}

public class Main
{
    public static void main(String[] args) {
      Inheritance obj2= new Inheritance();     //sub class object
      eval obj1=new eval();
         obj1.cal();
        obj2.cal();
    }
}
```

SAME NAME VARIABLE OR METHOD

```
class eval{  //super class
        int a=10;
        int b=5;
        int c;
        void cal()
        {
            c=a-b;
           System.out.println("parent output is ="+c);

        }
       }   //endof super class

class Inheritance extends eval {        //sub class
         int a=100,c;
        void cal()
            {c=a+b;
            System.out.println("child output is ="+c);}}

public class Main
{
    public static void main(String[] args) {
      Inheritance obj2= new Inheritance();     //sub class object
      eval obj1=new eval();
         obj1.cal();
        obj2.cal();
    }
}
```

Output:

```
parent output is =5
child output is =105
```

Note: The derived class cannot inherit a member of the base class if the derived class declares another member with the same name.

THANK YOU
?