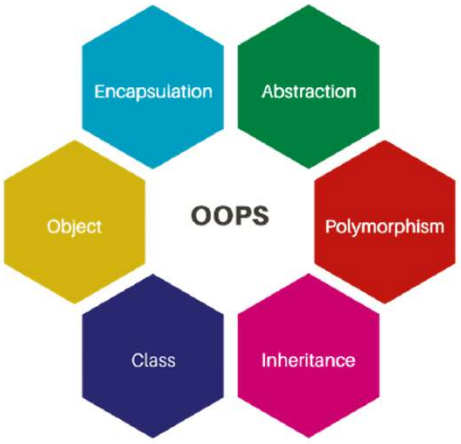


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Java™

Inheritance

ABESIT COLLEGE OF ENGINEERING
CRC-Training

What is Inheritance?

- Inheritance is a mechanism in which one class acquires the property of another class.
- For example, a child inherits the traits of his/her parents.
- With inheritance, we can reuse the fields and methods of the existing class.
- Inheritance facilitates Reusability and is an important concept of OOPs.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

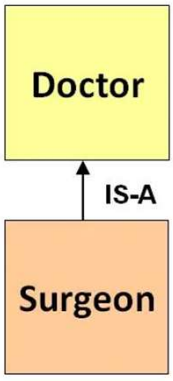
What is Inheritance?

- Inheritance is everywhere in Java.
- It's safe to say that it's almost impossible to write even the tiniest Java program without using inheritance.
- Every class in Java is a subclass of class Object (except, of course, class Object itself).
- In other words, every class you'll ever use or ever write will inherit from class Object.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

What is Inheritance?

- In Java, when an "Is-A" relationship exists between two classes, we use Inheritance.



ABESIT COLLEGE OF ENGINEERING
CRC-Training

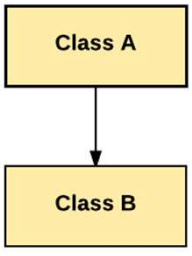
Inheritance in OOPs

- Single Inheritance
- Multiple Inheritance
- Multilevel Inheritance
- Hybrid Inheritance

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Inheritance in OOPs

- Single Inheritance –
  - In Single Inheritance, one class extends another class (one class only).



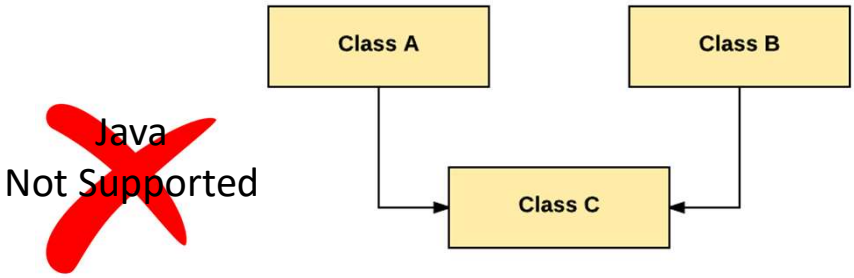

```
class B extends A
{
  //methods and fields
}
```

Class A is a super class and Class B is a Sub-class.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Inheritance in OOPs

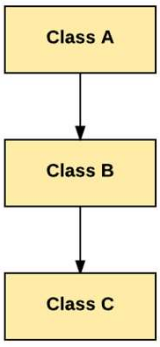
- Multiple Inheritance
  - In Multiple Inheritance, one class extends more than one class.
  - Java does not support multiple inheritances.



ABESIT COLLEGE OF ENGINEERING
CRC-Training

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.

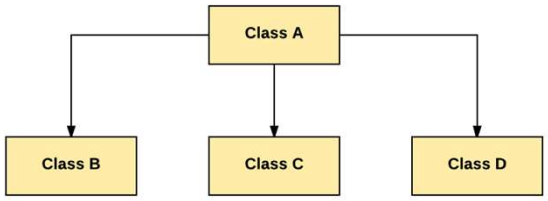



```
class A{
//methods and fields
}
class B extends A
{
  //methods and fields
}
class C extends B
{
  //methods and fields
}
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Hierarchical Inheritance

- In Hierarchical Inheritance, one class is inherited by many sub classes.

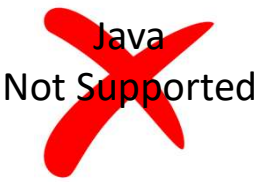


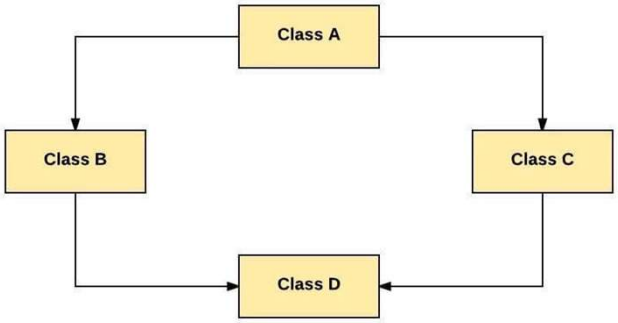

```
class A{
//methods and fields
}
class B extends A
{
  //methods and fields
}
class C extends A
{
  //methods and fields
}
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Hybrid Inheritance

- It is a mix of two or more of the above types of inheritance.
- Since java doesn't support multiple inheritances with classes, hybrid inheritance is also not possible with classes.





ABESIT COLLEGE OF ENGINEERING
CRC-Training

Super Keyword in Java

- The super keyword in Java is a reference variable which is used to refer immediate parent class object.
- Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Usage of Java super Keyword

- super can be used to refer immediate parent class instance variable.
- super can be used to invoke immediate parent class method.
- super() can be used to invoke immediate parent class constructor.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

super is used to refer immediate parent class instance variable.

- It is used if parent class and child class have same fields.

```
1  class A{
2      int i=12;
3  }
4  class B extends A{
5      int i=10;
6      void display(){
7          System.out.println(i);
8          System.out.println(super.i);
9      }
10 }
11 public class Main
12 {
13     public static void main(String[] args) {
14         B obj = new B();
15         obj.display();
16     }
17 }
```

input

```
10
12
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

super can be used to invoke parent class method

It should be used if subclass contains the same method as parent class.

```
1  class A{
2      void display(){
3          System.out.println("In Class A");
4      }
5  }
6  class B extends A{
7      void display(){
8          System.out.println("In Class B");
9          super.display();
10     }
11 }
12 public class Main
13 {
14     public static void main(String[] args) {
15         B obj = new B();
16         obj.display();
17     }
18 }
```

```
In Class B
In Class A
```

super is used to invoke parent class constructor

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Main.java

```
1  class person{
2      String name,address;
3      person(){}
4      person(String New_name, String addr){
5          name=New_name;
6          address=addr;
7      }}
8  class employee extends person{
9      int empcode;
10     employee(int ecode, String Name, String address){
11         super(Name,address);
12         empcode=ecode;
13     }
14     void display(){
15         System.out.println(empcode + " " + name + " " + address);
16     }}
17 public class Main
18 {
19     public static void main(String[] args) {
20     employee obj1 = new employee(123,"Gopal","Abesit");
21     obj1.display();
22     }
23 }
```

```
123 Gopal Abesit
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

When Constructors Are Executed – Multi-Level Inheritance

Main.java

```
1  class A{
2      A(){ System.out.println("I am GrandParent");}
3  }
4  class B extends A{
5      B(){ System.out.println("I am Parent");}
6  }
7
8  class C extends B{
9      C(){ System.out.println("I am child");}
10 }
11 public class Main
12 {
13     public static void main(String[] args) {
14     C obj = new C();
15     }
16 }
```

```
I am GrandParent
I am Parent
I am child
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Access Modifiers in Java

- There are two types of modifiers in Java: **access modifiers** and **non-access modifiers**.
- There are two types of modifiers in Java: **access modifiers** and **non-access modifiers**.

There are four types of Java access modifiers:

- **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
- **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
- **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
- **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Understanding Java Access Modifiers

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Private

The private access modifier is not accessible outside the class.

It is accessible only inside the class

Main.java

```
1  class A{
2      private int a=10;
3  }
4
5  public class Main
6  {
7      public static void main(String[] args) {
8      A obj = new A();
9      System.out.println(obj.a);
10     }
11 }
```

input

Compilation failed due to following error(s).

```
Main.java:9: error: a has private access in A
        System.out.println(obj.a);
                           ^
1 error
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Default

- If you don't use any modifier, it is treated as default by default.
- The default modifier is accessible only within package.
- It cannot be accessed from outside the package.

- **We will discussed in Package**

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Protected

The **protected access modifier** is accessible within package and outside the package but through inheritance only.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Class Level Access Modifier

Important Note –

- Protected can't be applied on the class.
- Private can't be applied on the class.

Main.java

```
1  private class A{
2      int pr =20;
3  }
4  public class Main
5  {
6      public static void main(String[] args) {
7      A obj = new A();
8      System.out.println(obj.pr);
9      }
10 }
```

input

Compilation failed due to following error(s).

```
Main.java:1: error: modifier private not allowed here
private class A{
        ^
1 error
```

Main.java

```
1  protected class A{
2      int pr =20;
3  }
4  public class Main
5  {
6      public static void main(String[] args) {
7      A obj = new A();
8      System.out.println(obj.pr);
9      }
10 }
```

input

Compilation failed due to following error(s).

```
Main.java:1: error: modifier protected not allowed here
protected class A{
          ^
1 error
```

Difference Between this and super Keyword in Java

ABESIT COLLEGE OF ENGINEERING
CRC-Training

this keyword in Java	super keyword in Java
this is an implicit reference variable keyword used to represent the current class.	super is an implicit reference variable keyword used to represent the immediate parent class.
this is to invoke methods of the current class.	super is used to invoke methods of the immediate parent class.
this is used to invoke a constructor of the current class.	super is used to invoke a constructor of the immediate parent class.
this refers to the instance and static variables of the current class.	super refers to the instance and static variables of the immediate parent class.

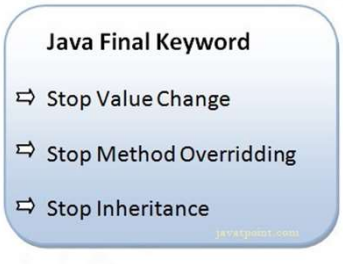
ABESIT COLLEGE OF ENGINEERING
CRC-Training

Final Keyword

The final keyword in java is used to restrict the user.

The java final keyword can be used in many context. Final can be:

- variable
- method
- class



ABESIT COLLEGE OF ENGINEERING
CRC-Training

Java final variable

- If you make any variable as final, you cannot change the value of final variable(It will be constant).
- A final variable that have no value it is called blank final variable or uninitialized final variable.
- It can be initialized in the constructor only.

Main.java

```
1  class A{
2      final int a=10;
3  }
4  public class Main {
5          public static void main(String args[]){
6              A obj = new A();
7              obj.a=20;
8          }
```

input

Compilation failed due to following error(s).

```
Main.java:7: error: cannot assign a value to final variable a
            obj.a=20;
                ^
```

Main.java

```
1  class A{
2      final int a;
3  }
4  public class Main {
5          public static void main(String args[]){
6              A obj = new A();
7              obj.a=20;
8          }
```

input

Compilation failed due to following error(s).

```
Main.java:2: error: variable a not initialized in the default constructor
    final int a;
              ^
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Java final variable

Main.java

```
1  class A{
2      final int a;
3      A(){
4          a=20;
5      }
6  }
7  public class Main {
8          public static void main(String args[]){
9              A obj = new A();
10             System.out.println(obj.a);
11         }
12 }
```

20

Blank final variable or uninitialized final variable.
It can be initialized in the constructor only.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Java final method

If you make any method as final, you cannot override it.

Main.java

```
1  class A{
2      final void display(){
3          System.out.println("In class A");
4      }
5  }
6  class B extends A{
7      void display(){
8          System.out.println("In class B");
9      }
10 }
11 public class Main {
12         public static void main(String args[]){
13             B obj = new B();
14             obj.display();
15         }
16 }
```

input

Compilation failed due to following error(s).

```
Main.java:7: error: display() in B cannot override display() in A
    void display(){
         ^
  overridden method is final
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Java final class

If you make any class as final, you cannot extend it.

Main.java

```
1  final class A{
2  }
3  class B extends A{
4  }
5  public class Main {
6          public static void main(String args[]){
7              B obj = new B();
8          }
9  }
```

input

Compilation failed due to following error(s).

```
Main.java:3: error: cannot inherit from final A
class B extends A{
                ^
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Is final method inherited?

- Yes, final method is inherited but you cannot override it.

Main.java

```
1  class A{
2      final void display(){
3          System.out.println("I am in A");
4      }
5  }
6  class B extends A{
7  }
8  public class Main {
9          public static void main(String args[]){
10             B obj = new B();
11             obj.display();
12         }
13 }
```

```
I am in A
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Can we declare a constructor final?

No

Main.java

```
1  class A{
2      final A(){
3          System.out.println("I am in A");
4      }
5  }
6  public class Main {
7          public static void main(String args[]){
8              A obj = new A();
9          }
10 }
```

input

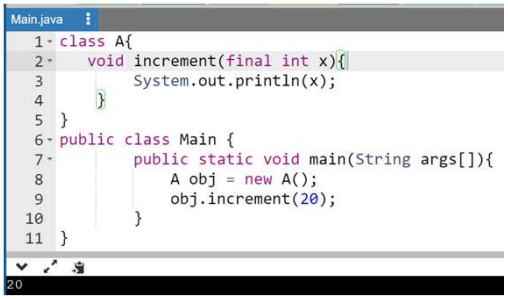
Compilation failed due to following error(s).

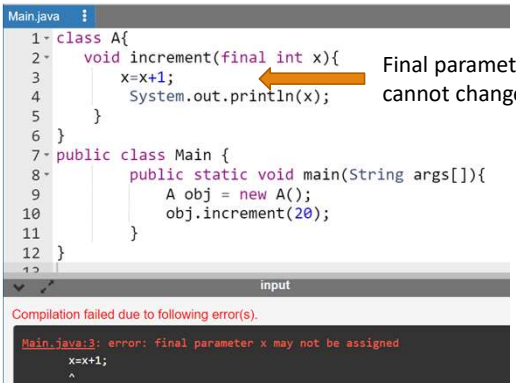
```
Main.java:2: error: modifier final not allowed here
    final A(){
          ^
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

What is final parameter?

If you declare any parameter as final, you cannot change the value of it.





Final parameter cannot changed

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Can we initialize blank final variable?

- Yes, but only in constructor.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

1) What will be the output of the following code?

```
class A
{
}
class B extends A
{
}
class C extends A, B
{
}
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

2) What is the output:

```
class A
{
A()
{
System.out.println("A Class Default Constructor Call ");
}
}
class B
{
B()
{
System.out.println("B class Default Constructor Call ");
}
}
class Test
{
        public static void main(String args[])
        {
                B obj=new B();
        }
}
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

3) What will be the output of the following code?

```
class A
{
A(int a)
{
System.out.println("A class Parameterized Constructor Call ");
}
}
class B extends A
{
B()
{
System.out.println("B Class Default Constructor Call ");
}
}
```

```
class Test
{
public static void main(String args[])
{       B obj=new B();
}
}
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

4) What will be the output of the following code?

```
class A
{
void showA()
{
    System.out.println("A show ");
}
}
final class B extends A
{
void showB()
{
    System.out.println("B Show ");
}
}
```

```
class Test
{
        public static void main(String args[])
        {
                B obj=new B();
                obj.showA();
                obj.showB();
}
}
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

5) What will be the output of the following code?

```
class A
{
private int a=10;
public int b=20;
int c=30;
protected int d=40;
}
class B extends A
{
void add()
{
     System.out.println(a+b+c+d);
}
}
public static void main(String args[])
{
        B obj=new B();
        obj.add();
}
```