Introduction to Java

By Sanjana Bandara

Content

- Object Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation
- OOPs Misc

Inheritance

01. Types of inheritance in Java

Types

- Single
- Multi level
- Hierarchical

- Multiple inheritance is not supported in java by classes

Sinale inheritance

- When a class inherits another class, it is known as a Single inheritance
- Example : Dog class inherits Animal class

- (practicle)

Multi level inheritance

- When there is a chain of inheritance, it is known as multi level inheritance
- Example :Puppy class inherites the dog class which again inherits the animal class

- (practicle)

Hierarchical inheritance

- When two or more classes inherits a single class, it is known as hierarchical inheritance.
- Example : dog and cat classes inherits animal class
- (practicle)

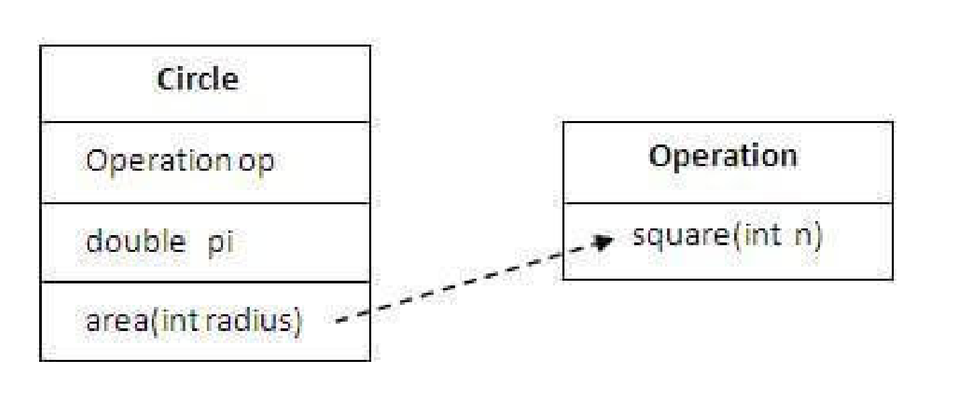
Why multiple inheritance not supported in Java?

- To reduce complexity and simplify language
- Example: A and B class inherits class C

02. Aaareaation in Java

Aaareaation in Java

- If a class has an entity reference,
- it is known as Aggregation. Aggregation represents HAS-A relationship.



Practicles

Polymorphism

01. Method Overloading in Java

Method Overloadina in Java

- Method Overloading: changing no. of arguments

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

Method Overloadina in Java

Method Overloading: changing data type of arguments

```
class Adder{
static int add(int a, int b){return a+b;}
static double add(double a, double b){return a+b;
}
}
class TestOverloading2{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6));
}}
```

02. Method Overridina in Java

Method Overridina in Java

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.

- Usage of Java Method Overriding
  - Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
  - Method overriding is used for runtime polymorphism
- Rules for Java Method Overriding
  - The method must have the same name as in the parent class
  - The method must have the same parameter as in the parent class.
  - There must be an IS-A relationship (inheritance).

Practicles

03. super keyword

super keyword in Java

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.

- 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.

super keyword in Java

1) super is used to refer immediate parent class instance variable.

```
class Animal{
String color="white";  }
class Dog extends Animal{
String color="black";
void printColor(){
System.out.println(color);//prints color of Dog class
System.out.println(super.color);//prints color of Animal class
}}
class TestSuper1{
public static void main(String args[]){
Dog d=new Dog();
d.printColor();
}}
```

super keyword in Java

2) super can be used to invoke parent class method

```
class Animal{
void eat(){System.out.println("eating...");}  }
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
void bark(){System.out.println("barking...");}
void work(){
super.eat();
bark();
}}
class TestSuper2{
public static void main(String args[]){
Dog d=new Dog();
d.work();
}}
```

super keyword in Java

3) super is used to invoke parent class constructor.

```
class Animal{
Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
Dog(){
super();
System.out.println("dog is created");
}
}
class TestSuper3{
public static void main(String args[]){
Dog d=new Dog();
}}
```

04. final keyword

final keyword in Java

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

final keyword in Java

1) Java final variable

If you make any variable as final, you cannot change the value of final variable(It will be constant).

```
class Bike9{
 final int speedlimit=90;//final variable
 void run(){
  speedlimit=400;
 }
 public static void main(String args[]){
 Bike9 obj=new  Bike9();
 obj.run();
 }
}//end of class
```

final keyword in Java

02) Java final method

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

```
class Bike{
  final void run(){System.out.println("running");}
}

class Honda extends Bike{
   void run(){System.out.println("running safely with 100kmph");}

   public static void main(String args[]){
   Honda honda= new Honda();
   honda.run();
   }
}
```

final keyword in Java

03) Java final class
If you make any method as final, you cannot override it.

```
final class Bike{}

class Honda1 extends Bike{
  void run(){System.out.println("running safely with 100kmph");}

  public static void main(String args[]){
  Honda1 honda= new Honda1();
  honda.run();
  }
}
```