NAME: VISHWAK SENAN

INHERITANCE

- It is one of the key features of Object Oriented Programming.
- It provide a mechanism that allow a class to inherit property of another class.
- The idea behind the inheritance in java is that we can create new classes that are built upon existing classes.
- When a new class extends another class it inherits all non-private members including fields and methods.
- Inheritance in java can be understood in terms of parent and child relationship, also know as super class(parent) and sub class(child) in java language.
- Inheritance defines a **is-a** relationship between a super class and its sub-class.
- When we inherit from an existing class, we can reuse methods and fields of the parent class. Moreover, we can add new methods and fields in our current class also.

SYNTAX

Class Subclass-name extends Superclass-name{

//methods and fields

}

- The extends keyword indicates that we are making new class that derives from an derived class.
- **Sub class/Child Class –** Subclass is a class which inherits the other class.It is also called as derived class,extended class, or child class.
- **Super class/Parent Class –** Superclass is the class from a where a subclass inherits the feature. It is also called a base class or a parent class.

Example

```
class Base{
int I;
void method1(){
System.out.println(“method 1”);
}
}
class Sub extends Base{
float f;
void method2(){
System.out.println(“method 2”);
                        } }
class InheritTest{
        public static void main(String args[]){
                                Sub obj = new Sub();
                                System.out.println(“obj.i”=+obj.i);
                                System.out.println(“obj.f”=+obj.f);
                                Obj.method1();
                                Obj.method2();
                        } }
```

OUTPUT

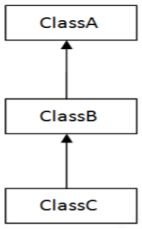
```
Obj.i = 0
Obj.f  =0.0
method 1
method 2
```

TYPES OF INHERITANCES IN JAVA

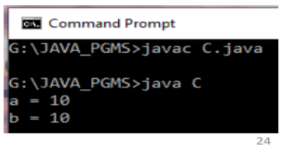
- Single
- Multilevel
- Hierarchical
- Multiple(does not support directly however implemented using secondary inheritance path in the form of interfaces)

```
class A
{
   int a = 10;
   void showA()
   {
      System.out.println("a = "+a);
   }
}
class B extends A
{
   int b = 10;
   void showB()
   {
      System.out.println("b = "+b);
   }
}
public class C extends B
{
   public static void main(String[] args)
   {
   C c = new C();
   c.showA();
   c.showB();
   }
}
```

Multilevel Inheritance Example

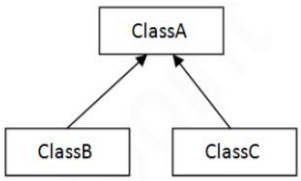


2) Multilevel



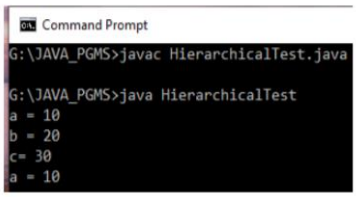

```
class A
{
   int a = 10;
   void showA()
   {
      System.out.println("a = "+a);
   }
}
class B extends A
{
   int b = 20;
   void showB()
   {
      System.out.println("b = "+b);
   }
}
class C extends A
{
   int c = 30;
   void showC()
   {
      System.out.println("c= "+c);
      showA();
   }
}
```

Hierarchical Inheritance Example



3) Hierarchical

```
public class HierarchicalTest
{
   public static void main(String[] args)
   {
      B b = new B();
      b.showA();
      b.showB();
       C c = new C();
      c.showC();
      c.showA();
   }
}
```



CONSTRUCTORS

- In java, constructor of base class with no arguments gets automatically called in derived class constructor.
- If we want to call parameterized constructor of base class, then we can call it using super()
- The point to note is base class constructor call mus be the first line in derived class constructor.

EXAMPLE

```
Class Base{
Base(){
System.out.println("base class constructor is called");
        }
    }
    Class Derived extends Base{
        Derived(){
            System.out.println("Derived class is called");
        }
    }
    public class Tester{
        Public static void main(String args[]){
            Derived d = new Derived();
        }
    }
```

OUTPUT :

```
base class constructor is called
Derived class is called
```

METHOD OVERRIDING

- When we declare the same method in child class (sub class) which is already present in the parent class(super class), then the method of the subclass class overides the method of method the superclass . This is know as **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
- Method overriding performs only if two classes have is-a relationship.it mean class must have inheritance.
- Method overriding is also referred to as runtimre polymorphism because caling method is decided by JVM during run time.
- In overriding,methods of both class must have same name and equal number of parameters.
- We cannot override static method.Because static is bound to class whereas method overriding is associated with object.

POLYMORPHISM

RUNTIME POLYMORPHISM

- It is a process in which a call to an ovrridden method is resolved at runtime rather than compile time.
- In this process , an overridden method is called through the reference variable of superclass
- The determination of the method to be called is based on the object being referred to by the reference variable.
- When parent class reference variable refers to child class object it is know as upcasting.
- In those cases we can create a parent class refrence and assign child classes objects to it.

EXAMPLE

```
class Animal{
void eat(){
System.out.println("Animal eats food");
        }
    }
    Class Dog extends Animal{
    Void eat(){
    System.out.println("dogs drink milk");
        }
    }
    Class BabyDog extends Dog{
    Void move(){
    System.out.println("BabyDogs can move");
        }
    }
    Class RunTime Polymorphism{
    Public static void(String args[]){
     Animal a = new Animal();
    A.eat();
        }
    }
```

OUTPUT

Dogs drink milk

STRING HANDLING

- String is an object that represents sequence of characters. In java, String is represented by String class which is located into java.lang package.
- It is probably most commonly used class in java library.
- In java,every String that we create is actually an object of type string.
- The String is immutable which means it cannot be changed.
- The java.lang.String class implements serializable, comparable and charSequence and interfaces.

CharSequence:

- The charSequnece interface is used to represent the sequence of characters.String,StringBuffer and StringBuilder classes implement it.
- It means,we can create String in java by using three classes
  - String
  - StringBuffer
  - StringBuilder

String:

- String is a sequence of characters but in java, String is an object that represents a sequence of characters. The java.lang.String class is used to create a string object.

String object:

- By String literal
- By new Keyword

String literal:

- Java literal is created by using double quotes(“ ”).
- Each time you create a String literal, the JVM checks the “string constatnt pool” first.
- If the string already exists in the pool,a new string instance is created and placed in the pool.

New Keyword:

- In such case, JVM will create a new string object in normal(non-pool)heap memory, and the literal “welcome” will be placed in the string constant pool.
- The variable s will refer tot the object in a heap(non-pool)