**What is an object in Java?**

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class. An entity that has state and behavior is known as an object

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

**What is a class in Java**

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

**\*\*Real life example of class and object in java:**

1. Class: Human Object: Man, Woman

2. Class: Fruit Object: Apple, Banana, Mango, Guava wtc.

3. Class: Mobile phone Object: iPhone, Samsung, Moto

4. Class: Food Object: Pizza, Burger, Samosa

**\*\*Can we have multiple classes in a single java file?**

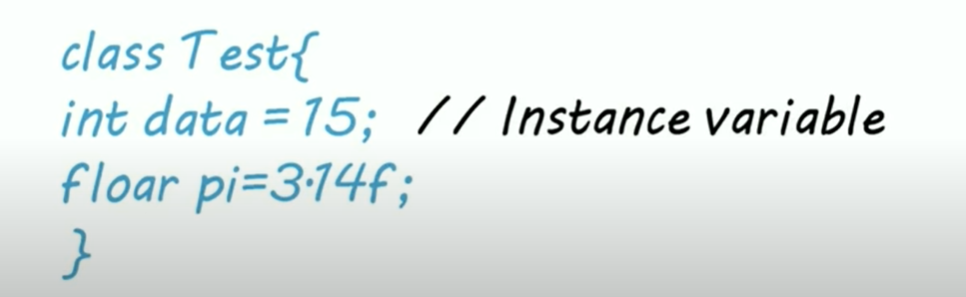
Yes, we can have multiple classes in same java file. But, there is one restriction over here, which is that you can have as many classes in one file but only one public class is allowed. If we try to declare 2 classes as public in the same file, the code will not compile.

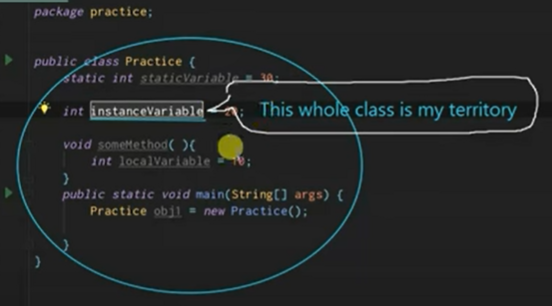
**Instance variable in Java**

A variable which is created inside the class but outside the method is known as an instance variable. Instance variable doesn't get memory at compile time. It gets memory at runtime when an object or instance is created. That is why it is known as an instance variable.

- They always get a value by default.

- Can’t be reinitialized directly within a class, but can be done in a method or constructor.



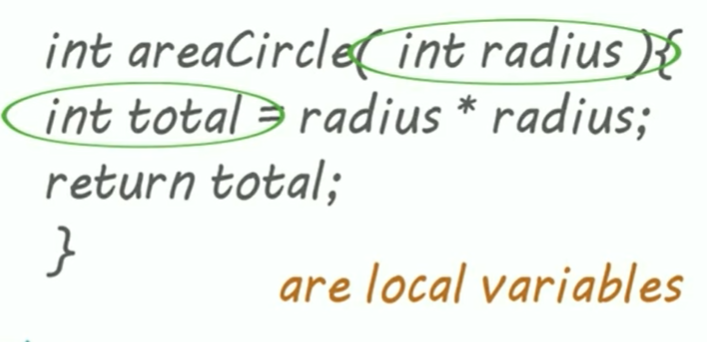
****

**Local variable in Java**

Declared inside method or method parameters

- They don’t get value by default.

- Not accessible outside method.

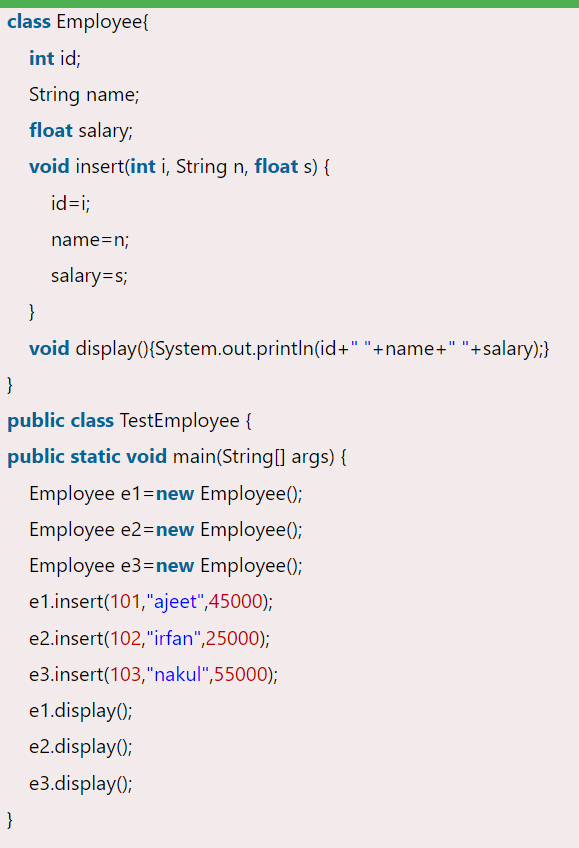


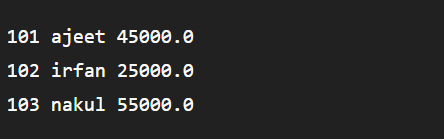
**New keyword in Java**

The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.

The new operator is used in Java to create new objects

Simple code for class and object:



****

**What are constructors?**

- In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

- It is a special type of method which is used to initialize the object.

\*\* It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.

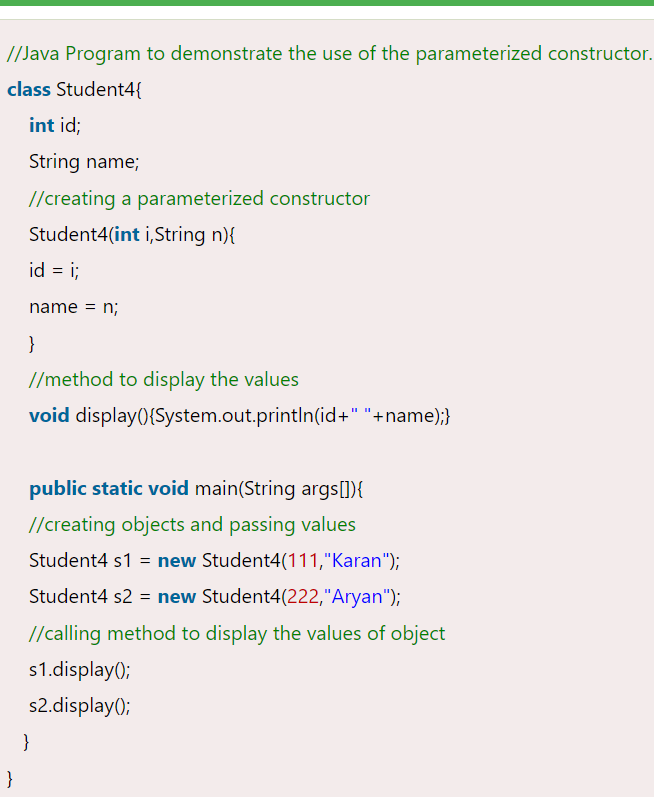
Rules-

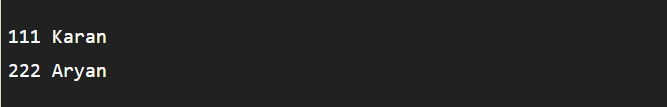
1. Constructor name must be the same as its class name

2. A Constructor must have no explicit return type

3. A Java constructor cannot be abstract, static, final, and synchronized

\*\*In Java, a constructor is just like a method but without return type. It can also be overloaded like Java methods.





**Static Keyword in Java**

We can apply static keyword with variables, methods, blocks and nested classes. The static keyword belongs to the class rather than an instance of the class.

\***Static Variable**

- The static variable gets memory only once in the class area at the time of class loading.

- The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.

\***Static Method**

- A static method belongs to the class rather than the object of a class.

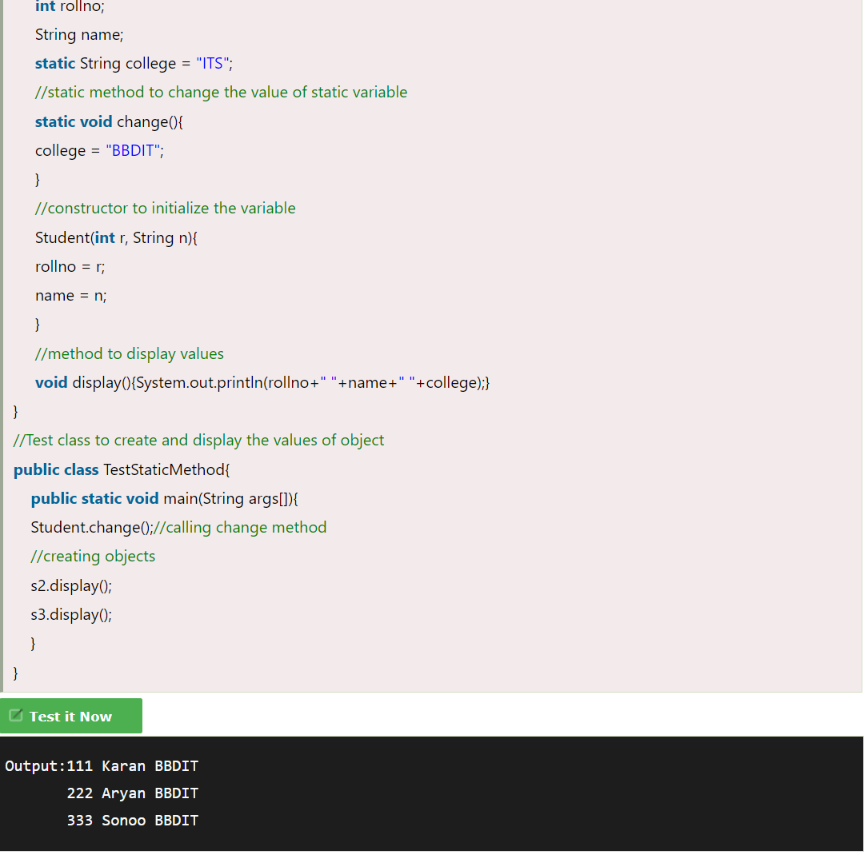
- A static method can be invoked without the need for creating an instance of a class.

- A static method can access static data member and can change the value of it.

\*\*There are two main restrictions for the static method. They are:

1. The static method can not use non static data member or call non-static method directly.

2. this and super cannot be used in static context.



**What are the access modifiers?**

There are four types of Java access modifiers:

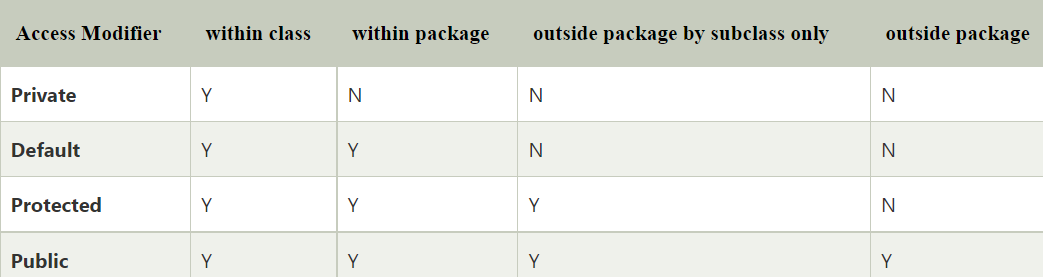
-Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

-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.

\*\*If you are overriding any method, overridden method (i.e. declared in subclass) must not be more restrictive.

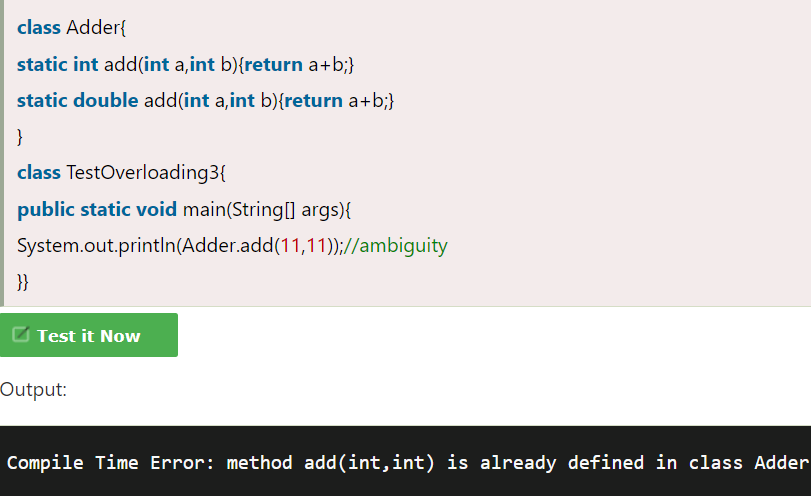


**What is Method overloading ?**

If a class has multiple methods having same name but different in parameters (return type doesn’t matter in Java), it is known as Method Overloading.

If we have to perform only one operation, having same name of the methods increases the readability of the program.

\*\* In java, method overloading is not possible by changing the return type of the method only because of ambiguity.

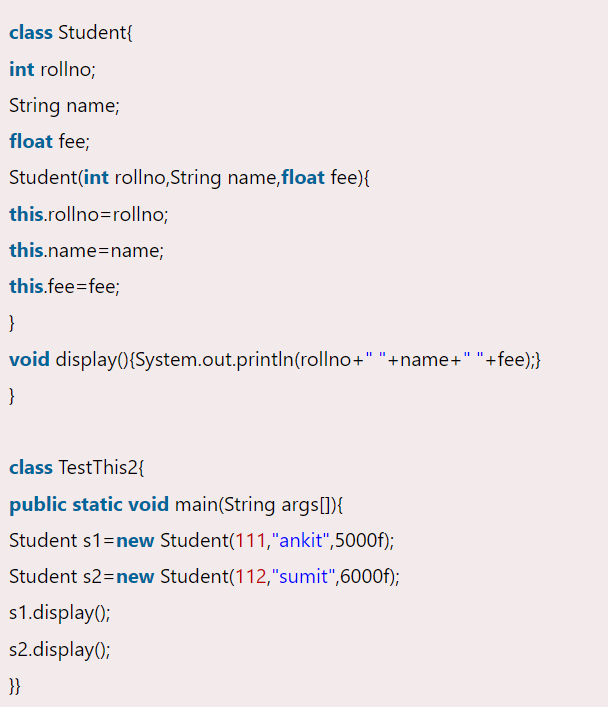
****

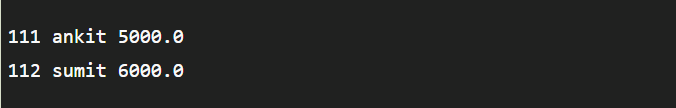
**this keyword**

in simple terms ‘this’ keyword is used to refer to any member of the current object

\*\*there are many uses of this, one of which is:

- to refer current class instance variable





**Inheritance**

- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors 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.

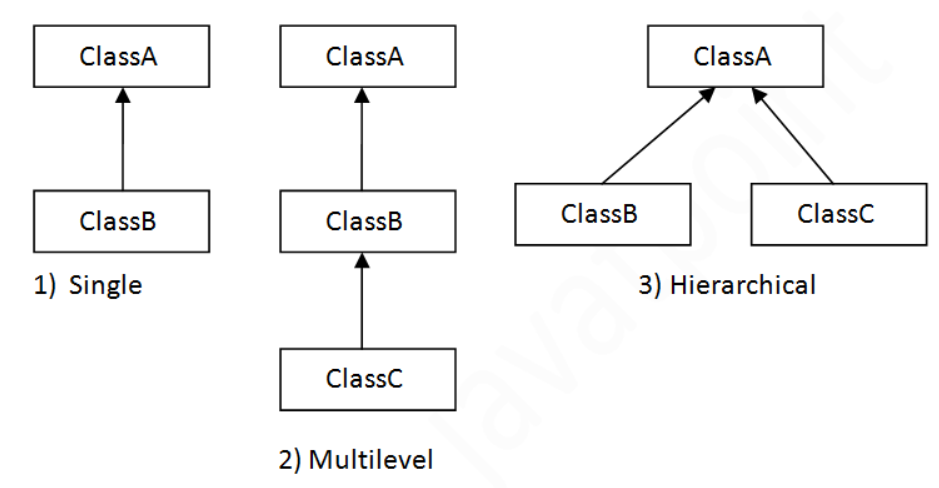
- Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

Why use inheritance in java

1. For Method Overriding (so runtime polymorphism can be achieved).

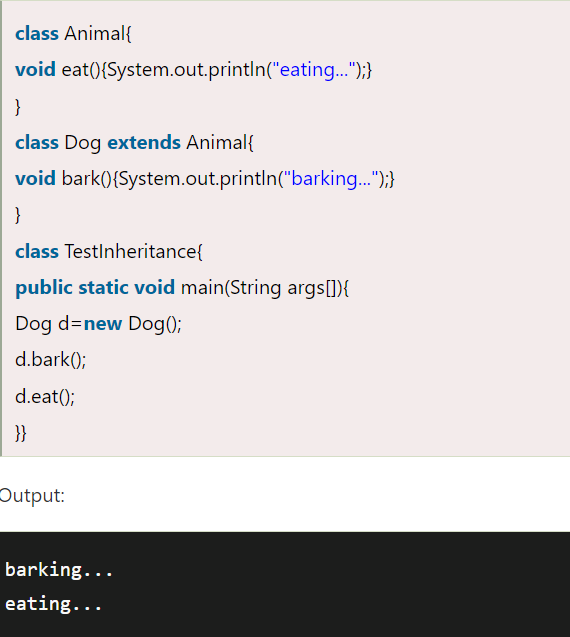
2. For Code Reusability.

**\*\*Types of inheritance -**

****

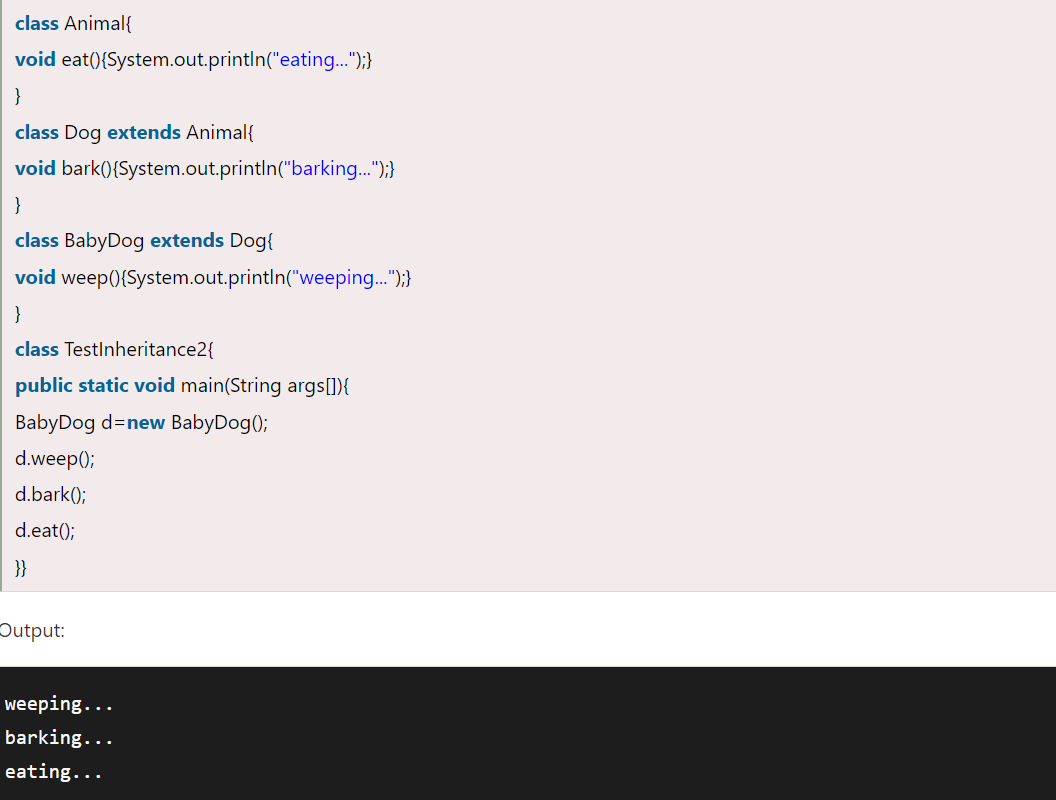
1. Single inheritance

When a class inherits another class, it is known as a single inheritance. In the example given below, Dog class inherits the Animal class, so there is the single inheritance.



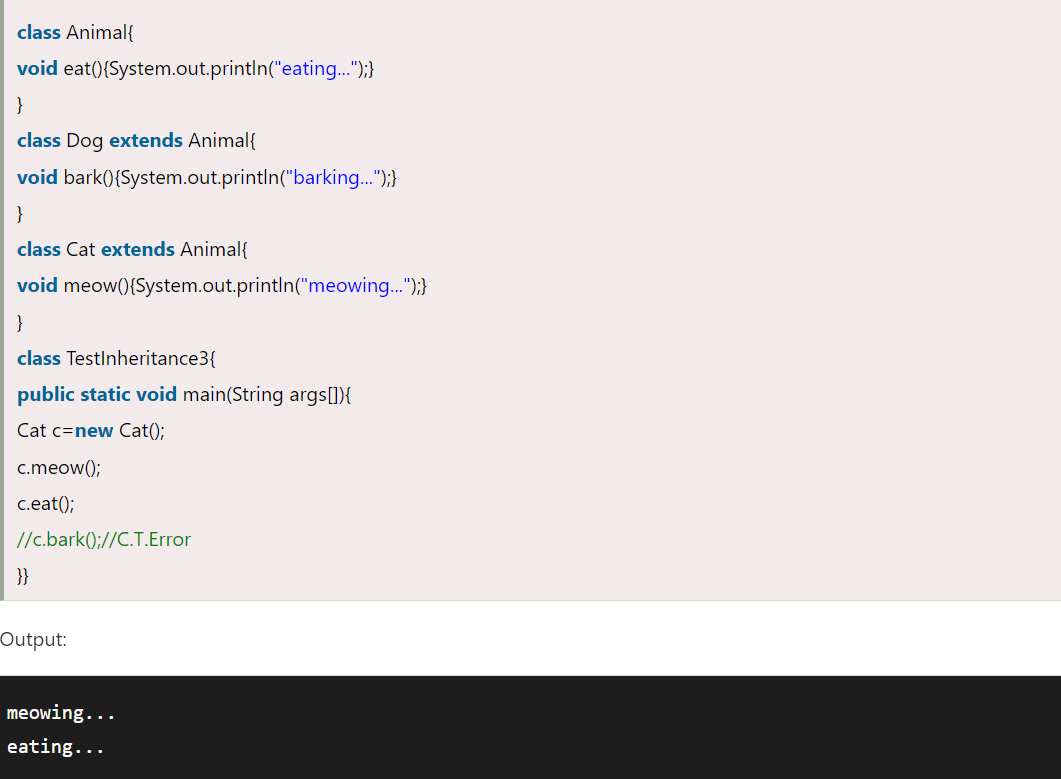
2. Multilevel inheritance

When there is a chain of inheritance, it is known as multilevel inheritance. As you can see in the example given below, BabyDog class inherits the Dog class which again inherits the Animal class, so there is a multilevel inheritance.



3. Hierarchical inheritance

When two or more classes inherits a single class, it is known as hierarchical inheritance. In the example given below, Dog and Cat classes inherits the Animal class, so there is hierarchical inheritance.



\*\*To reduce the complexity and simplify the language, multiple inheritance is not supported in java, though they can be implemented via interfaces

**What is Method Overriding?**

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

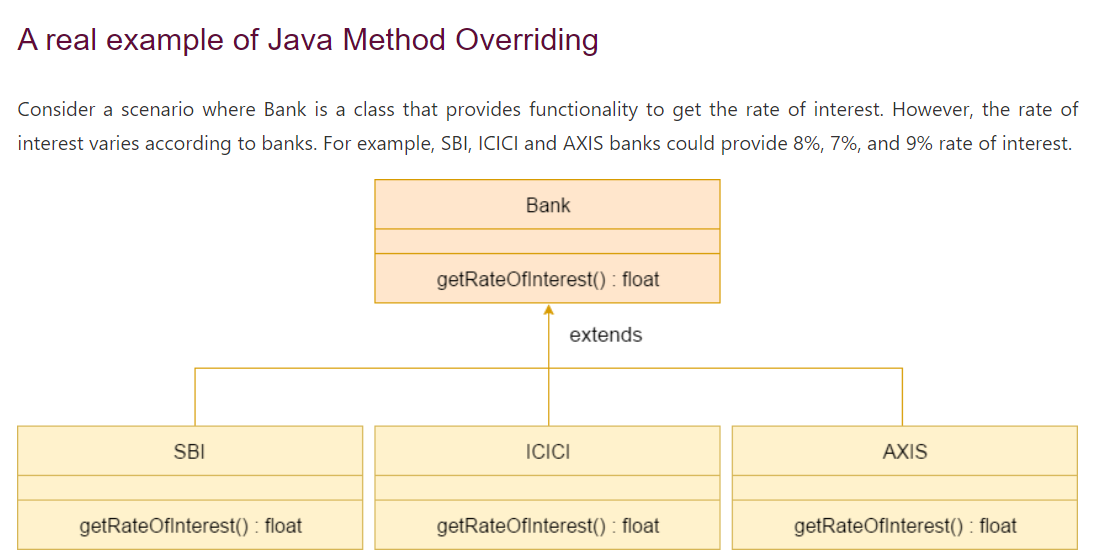
\*\*Rules for Java Method Overriding:

1. The method must have the same name as in the parent class

2. The method must have the same parameter as in the parent class.

3. There must be an IS-A relationship (inheritance)

\*\*JVM gives preferance to object type



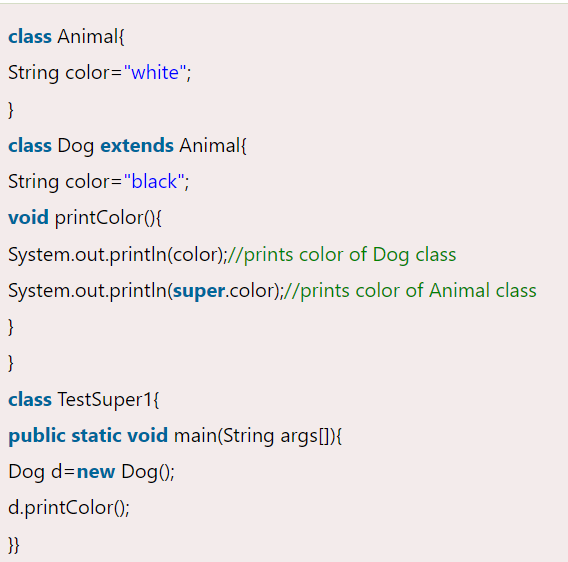
**Super keyword**

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

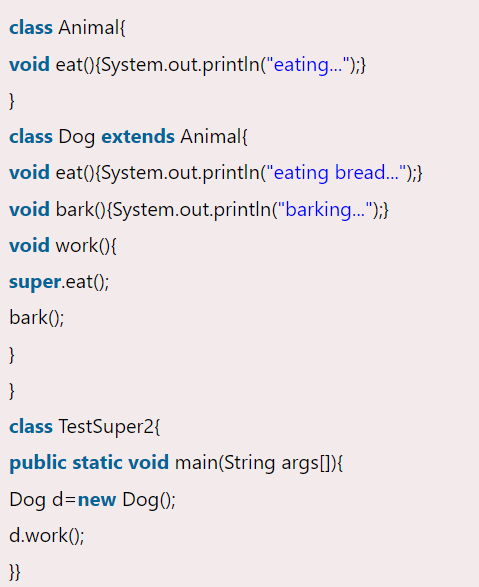
- Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

\*\*Uses-

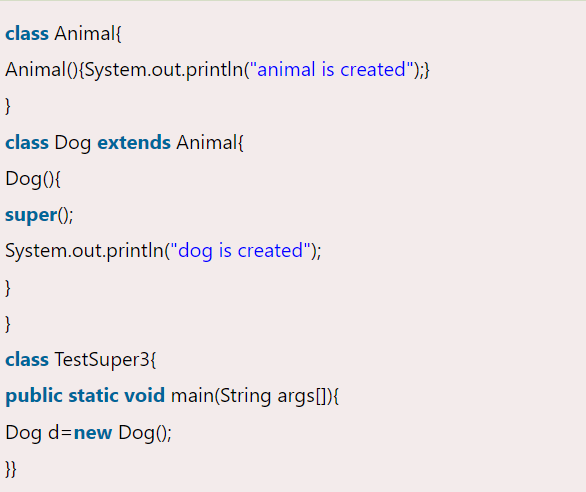
1. super can be used to refer immediate parent class instance variable.



2. super can be used to invoke immediate parent class method.



3. super() can be used to invoke immediate parent class constructor.

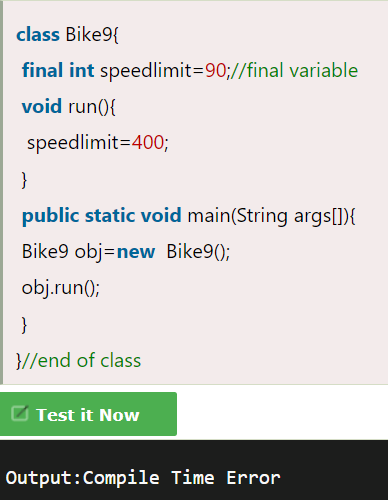


**Final keyword**

The final keyword in java is used to restrict the user. Final can be:

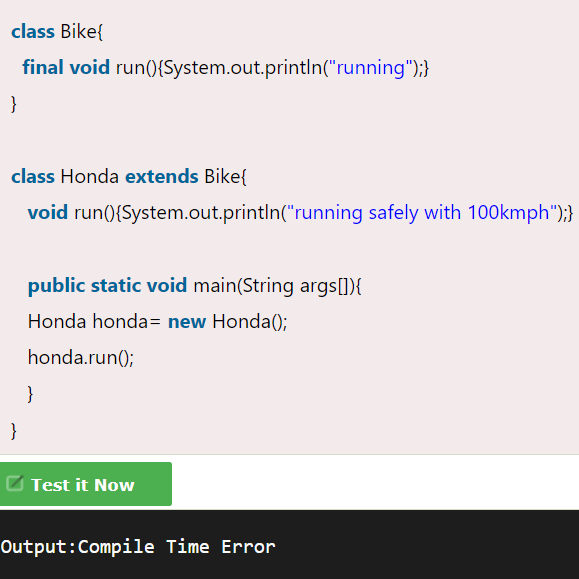
1. variable

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



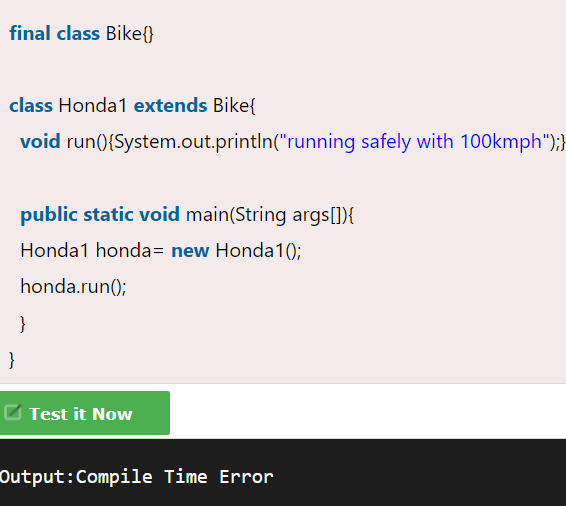
2. method

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



3. class

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

****

**Blank or Uninitialized final variable?**

- A final variable that is not initialized at the time of declaration is known as blank final variable.

- It can be initialized only in constructor.

**What is final parameter?**

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

**Polymorphism**

Polymorphism in Java is a concept by which we can perform a single action in different ways

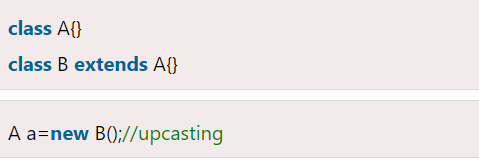
Two types-

Compile time poly

Runtime poly

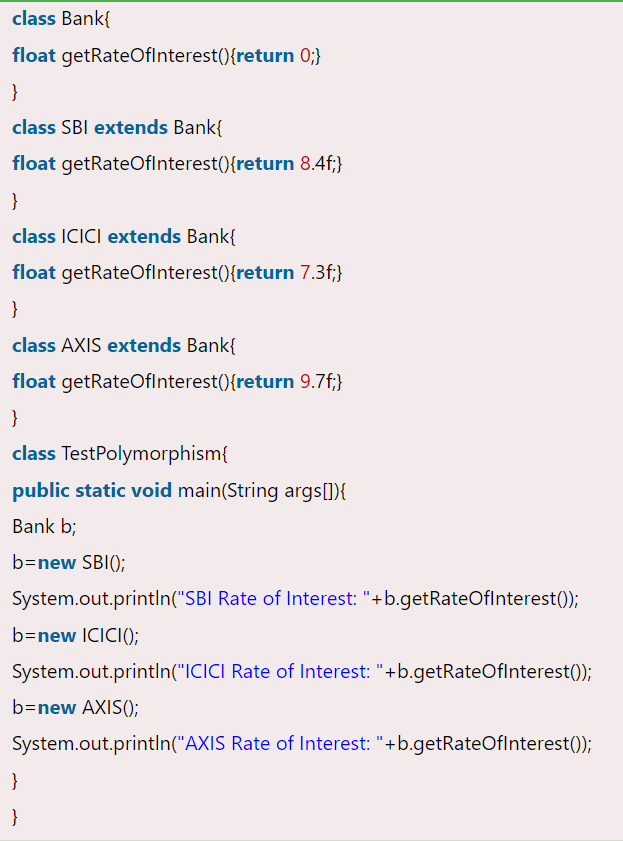
**\*\*Upcasting**

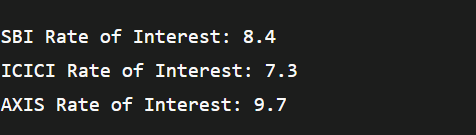
If the reference variable of Parent class refers to the object of Child class, it is known as upcasting. For example:



**Runtime Polymorphism**

A process, in which overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the **object being referred to by the reference variable.** Since it is resolved at runtime rather than compile-time, it called Runtime polymorphism.





**Static Binding and Dynamic Binding**

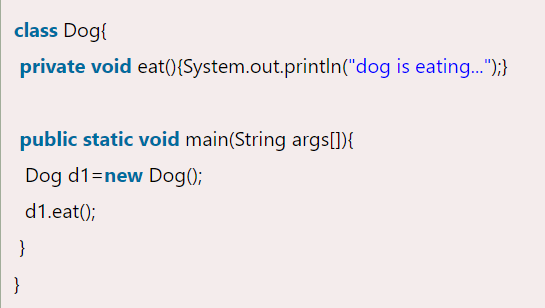
- Connecting a method call to the method body is known as binding.

- There are two types of binding

1. Static Binding (also known as Early Binding).

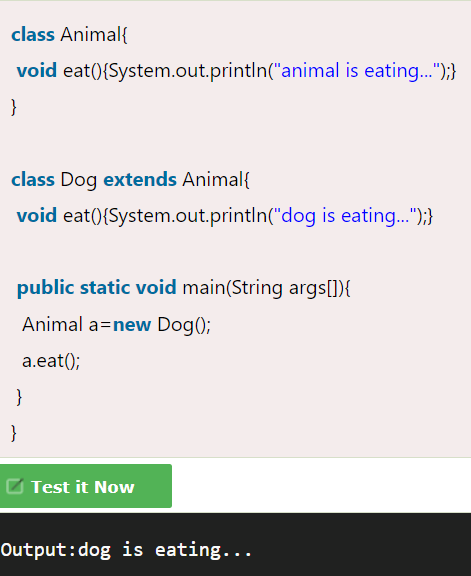
- When type of the object is determined at compiled time(by the compiler), it is known as static binding.

- If there is any private, final or static method in a class, there is static binding.



2. Dynamic Binding (also known as Late Binding).

- When type of the object is determined at run-time, it is known as dynamic binding.



**Abstraction**

Process of hiding the implementation details and showing only functionality to the user.

**Abstract class**

It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

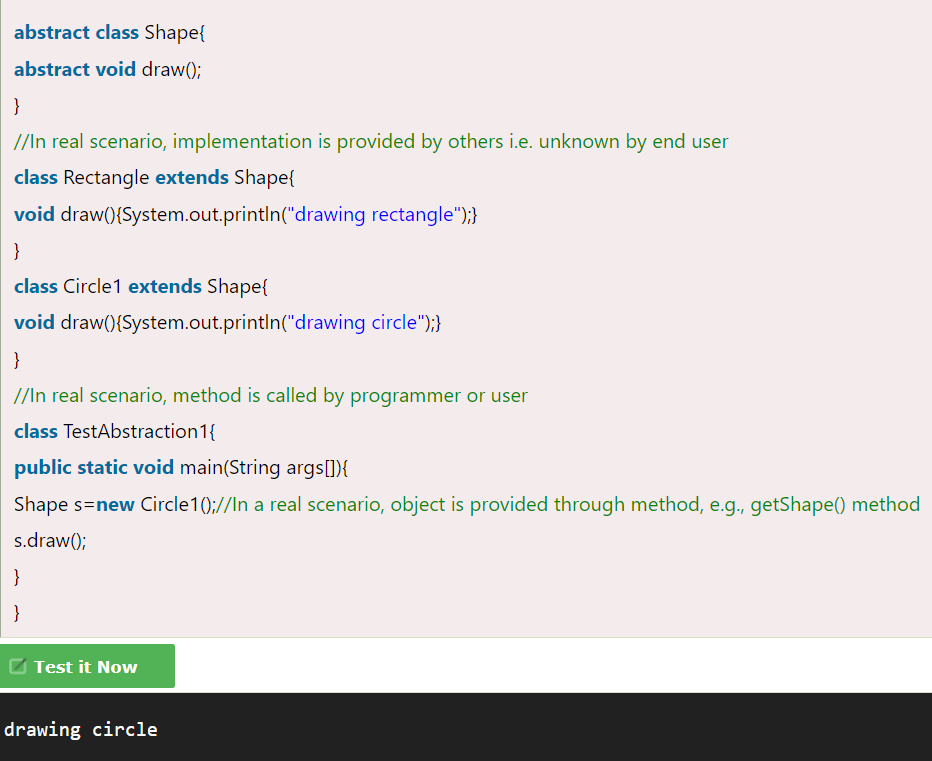
**Abstract method**

A method which is declared as abstract and does not have implementation is known as an abstract method.

\*\* Need of abstract class and method

1. An abstract class can be used as a type of template for other classes. The abstract class will hold common functionality for all classes that extend it.

2. Without abstract classes, you would have to provide dummy implementations of the methods you intend to override ... which could be done, but then there'd be the risk of forgetting to implement one of them. Having some methods remain entirely abstract ensures that the real implementations have to fill in the gaps, or continue to be abstract themselves and force their descendents to do so.

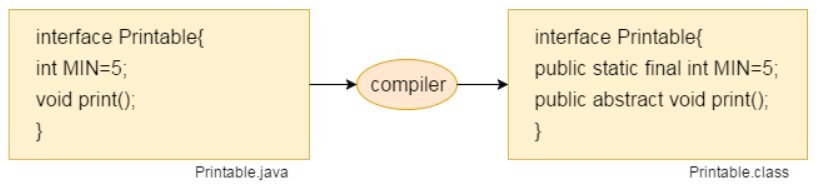


**Interface**

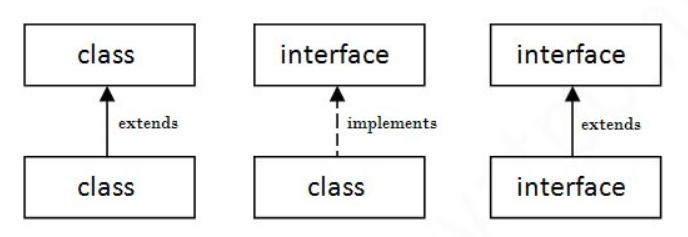
- An interface in Java is a blueprint of a class. It has static constants and abstract methods.

- Java Interface also represents the IS-A relationship.

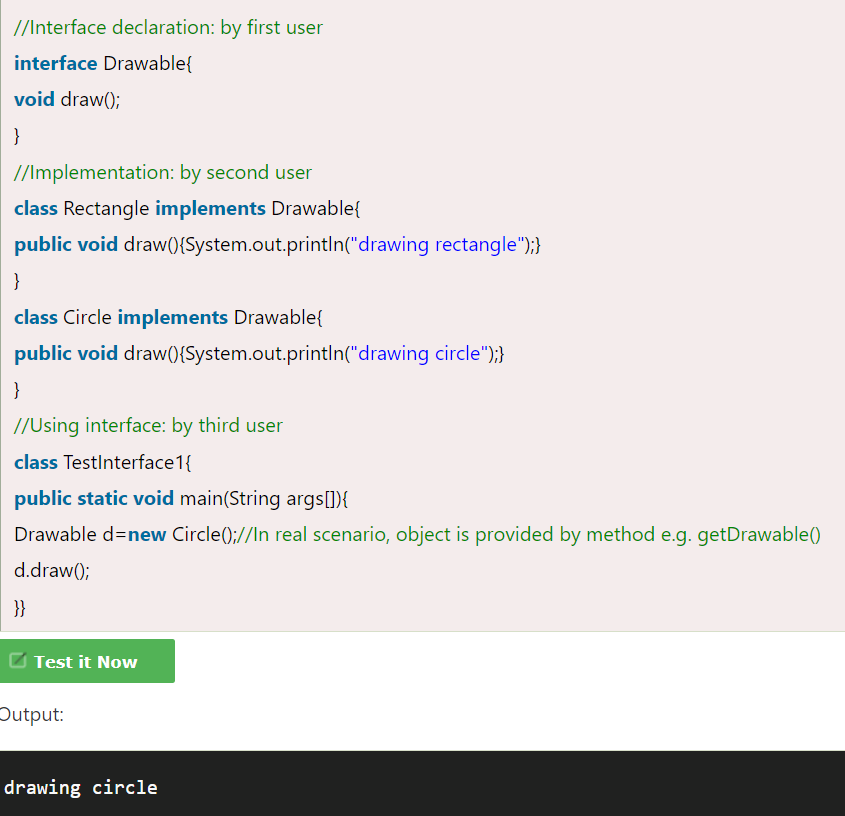
- It cannot be instantiated just like the abstract class

\*\*The Java compiler adds public and abstract keywords before the interface method. Moreover, it adds public, static and final keywords before data members.

\*\* A class extends another class, an interface extends another interface, **but a class implements an interface.**

****

**Interface code :**

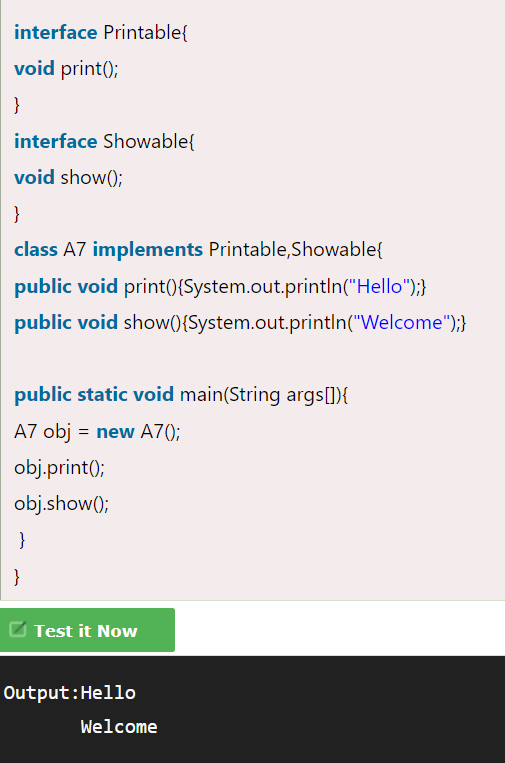


**;**

**Multiple inheritance in Java by interface**

If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.





**Encapsulation**

- Encapsulation in Java is a process of wrapping code and data together into a single unit

- We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

\*\*Advatage

1. It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.

2. It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.

