**Polymorphism**

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

Types of polymorphism

1. Static polymorphism

2. Dynamic polymorphism

1. Static Polymorphism / Compile time Polymorphism

Static polymorphism is a process in which a call to an overloaded method is resolved during compile time.

Example: Method Overloading

2. Dynamic Polymorphism / Runtime Polymorphism

Dynamic polymorphism is a process in which a call to an overridden method is resolved at runtime.

Example: Method Overriding

Method Overloading:

A class having many methods with same name but different in number of parameters, data types of parameters & return type of method.

To overload the method in java

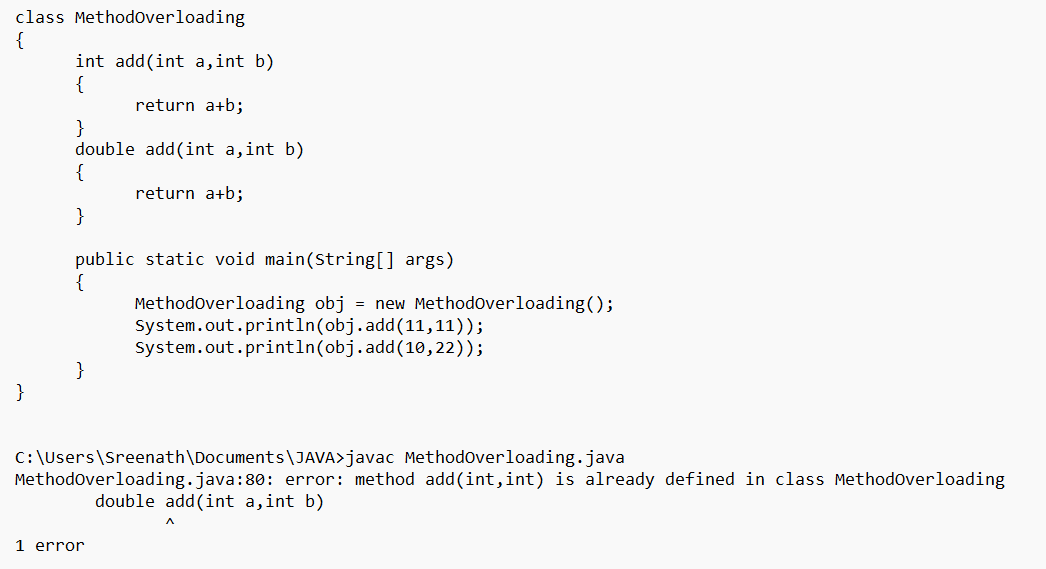
1. By changing number of arguments.

2. By changing the data type of arguments.

3. By changing return type of method.

Why Method Overloading is not possible by changing only the return type of method?

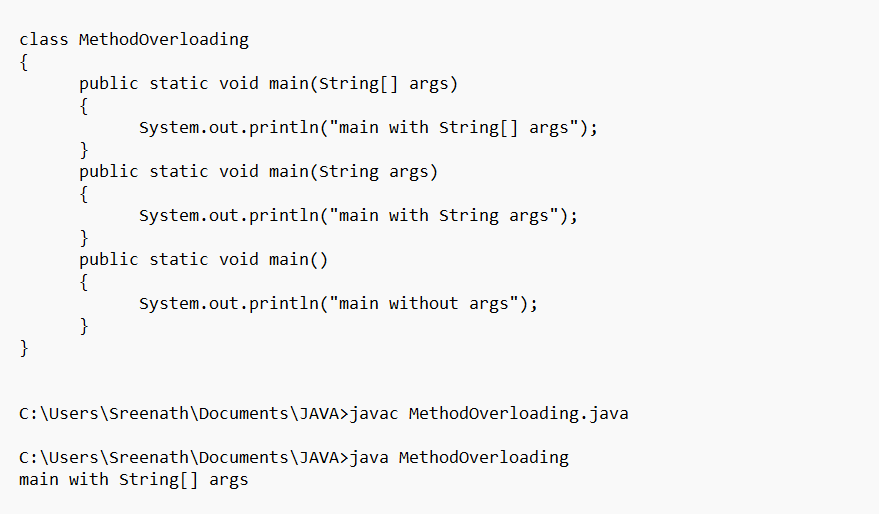
In java, method overloading is not possible by changing only the return type of the method because of ambiguity.

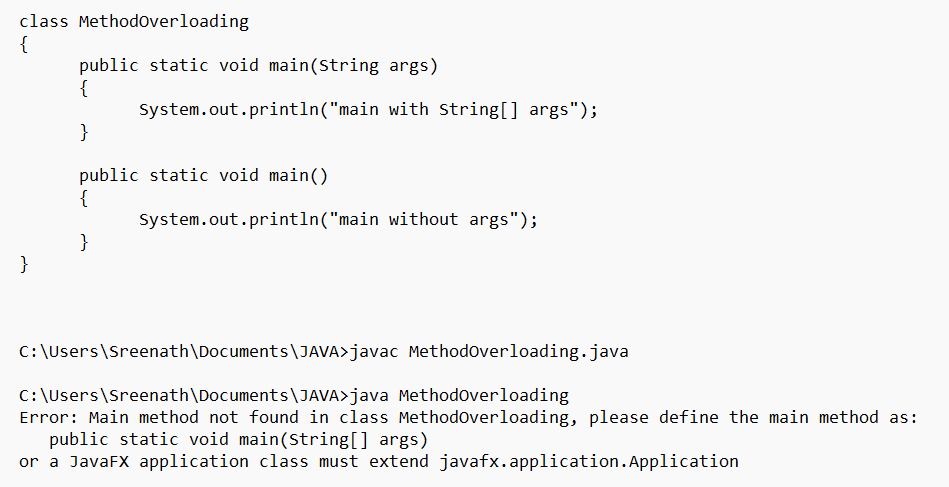


Can we overload java main() method?

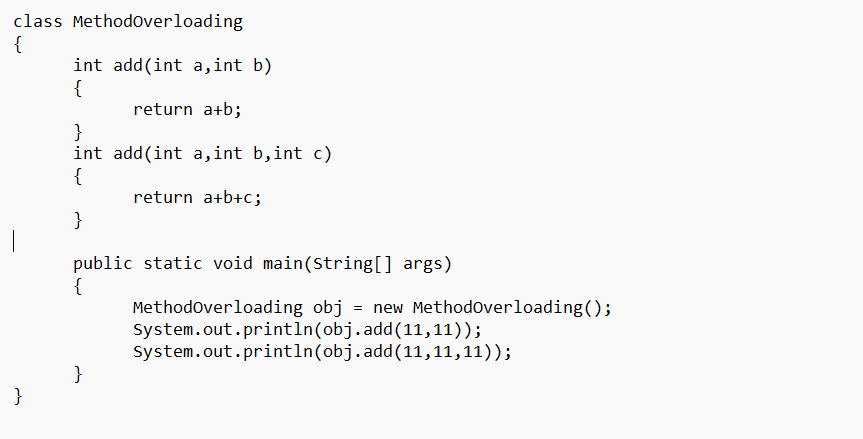
Yes, you can have any number of main methods in a class by method overloading.

But JVM calls main() method which receives string array as argument only.

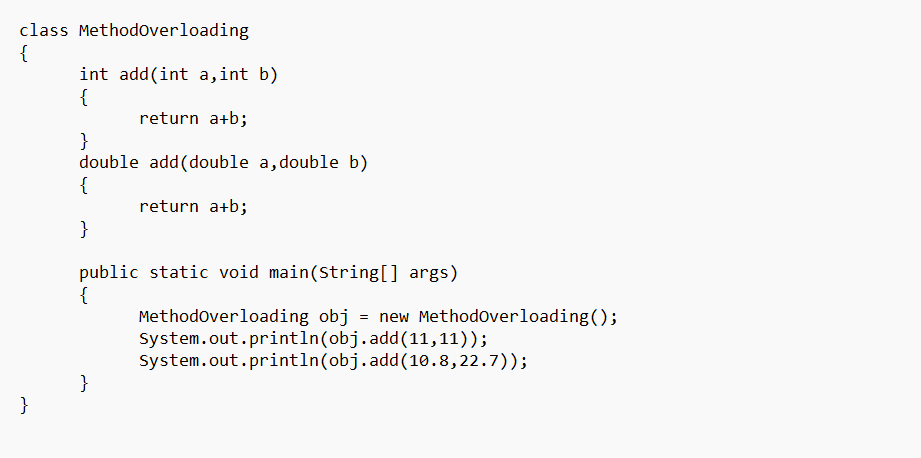




By changing number of arguments



By changing the data type of arguments



Method Overriding

If subclass (child class) has the same method as declared in the parent class.

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).

method must have the same return type.

FINAL METHODS cannot be OVERRIDDEN.

CONSTRUCTORS cannot be OVERRIDDEN.

CAN WE OVERRIDE STATIC METHOD ?

No, a static method cannot be overridden. It can be proved by runtime polymorphism.

WHY CAN WE NOT OVERRIDE STATIC METHOD ?

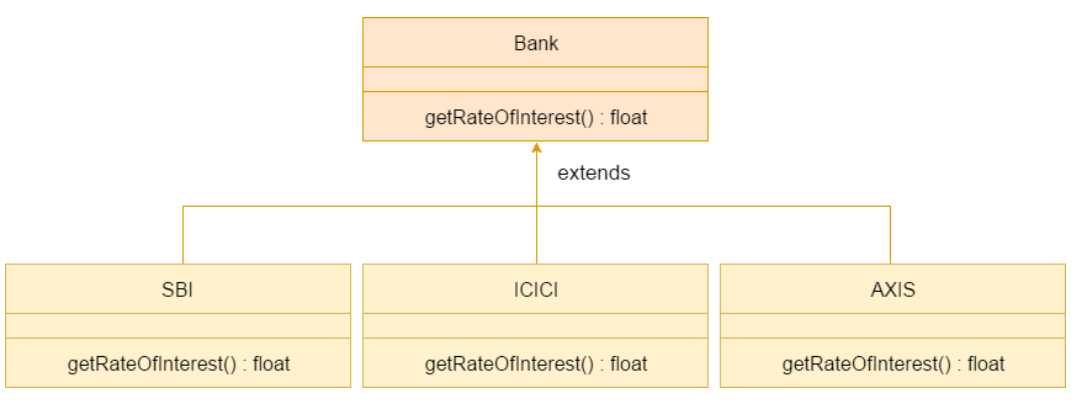
It is because the static method is bound with class whereas instance method is bound with an object.

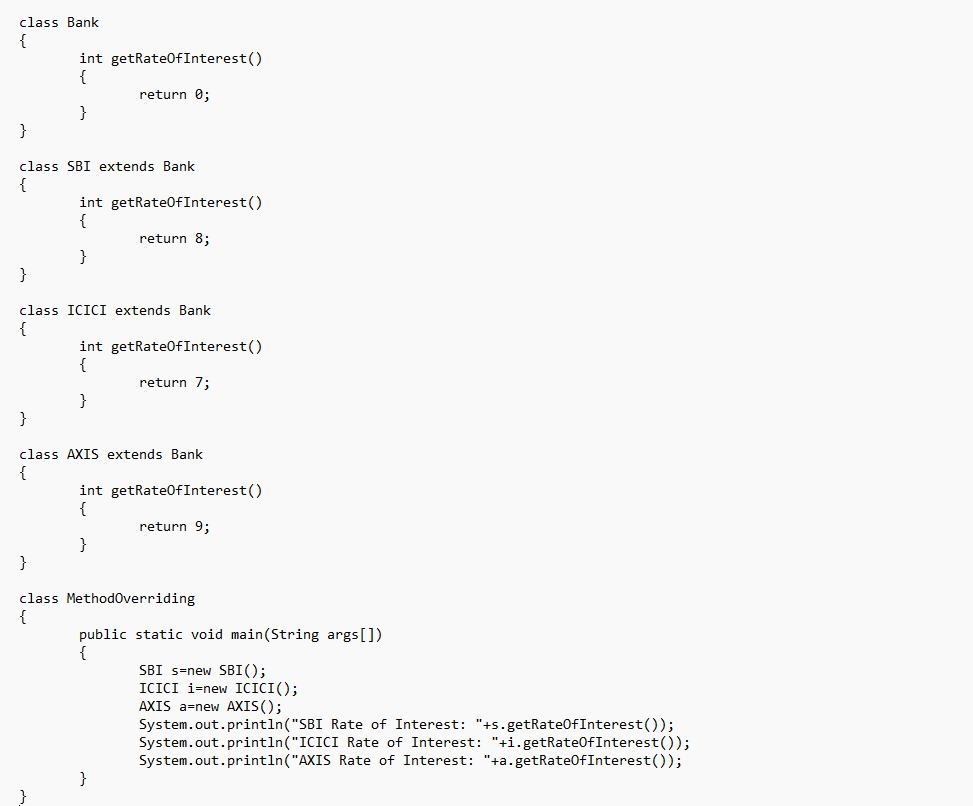
Static belongs to the class area, and an instance belongs to the heap area.

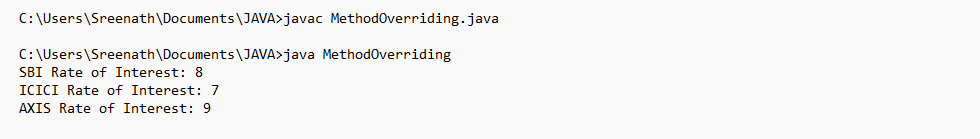
CAN WE OVERRIDE JAVA MAIN METHOD ?

No, because the main is a static method.

A REAL EXAMPLE OF JAVA METHOD OVERRIDING







Runtime Polymorphism / Dynamic Polymorphism / Dynamic method Dispatch

Runtime polymorphism / Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

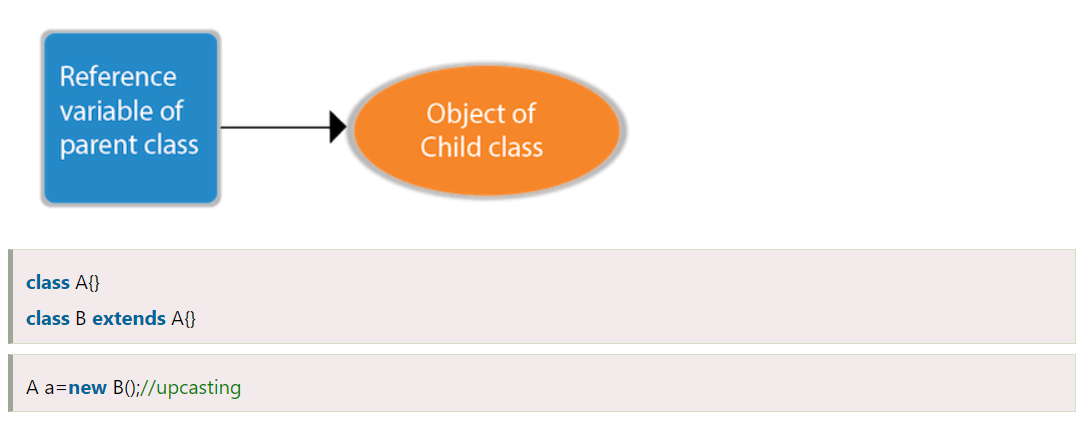
An overridden method is called through the reference variable of a superclass.

A super class reference variable can hold a reference of a subclass object.

UPCASTING

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

Example:



For upcasting, we can use the reference variable of class type or an interface type.

Example:

interface I{}

class A{}

class B extends A implements I{}

Here, the relationship of B class would be:

B IS-A A

B IS-A I

B IS-A Object

Since OBJECT class is the root class of all classes in JAVA, so we can write B IS-A Object.

EXAMPLE OF JAVA RUNTIME POLYMORPHISM

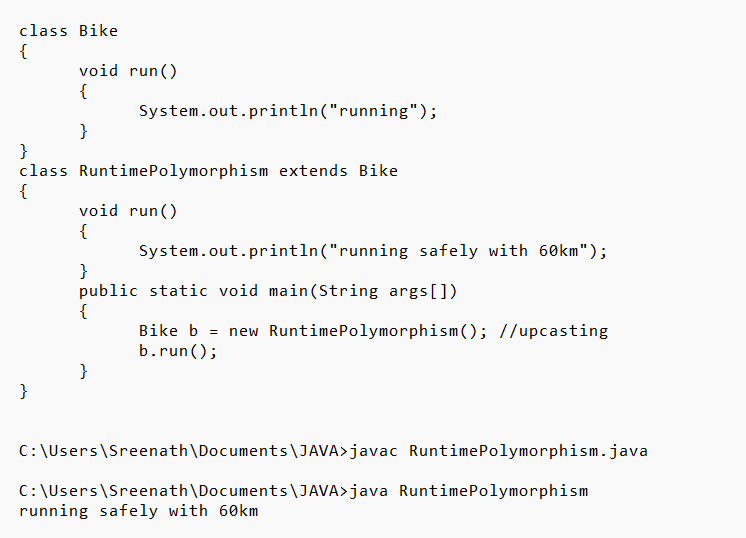
In this example, we are creating two classes Bike and Splendor.

Splendor class extends Bike class and overrides its run() method.

We are calling the run method by the reference variable of Parent class.

Since it refers to the subclass object and subclass overrides the method of Parent class, the subclass method is invoked at runtime.

Since method invocation is determined by the JVM not compiler, it is known as runtime polymorphism.



RUNTIME POLYMORPHISM CAN NOT BE ACHIEVED BY DATA MEMBER

A method is overridden, not the data members, so runtime polymorphism can't be achieved by data members.

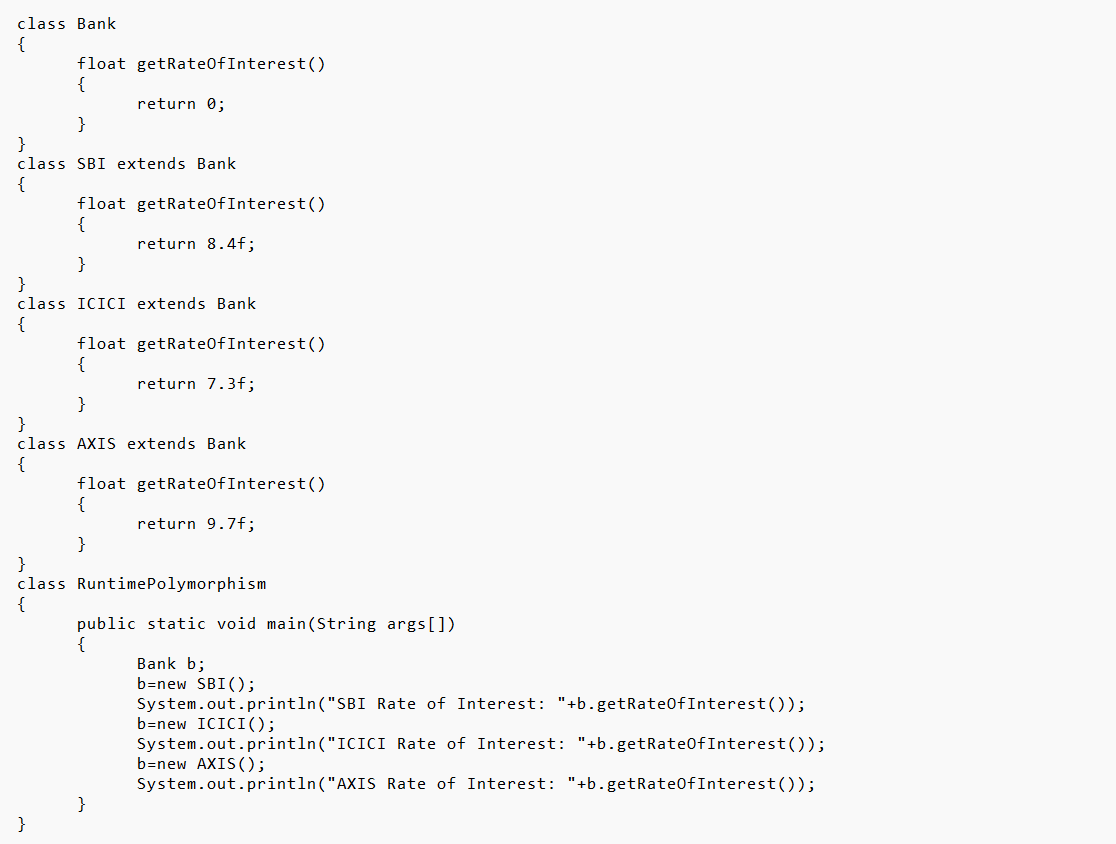
In the example given below, both the classes have a data member speedlimit.

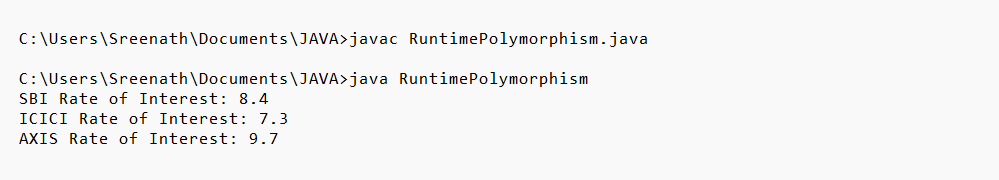
We are accessing the data member by the reference variable of Parent class which refers to the subclass object.

Since we are accessing the data member which is not overridden, hence it will access the data member of the Parent class always.



JAVA RUNTIME POLYMORPHISM : Bank





JAVA RUNTIME POLYMORPHISM WITH MULTILEVEL INHERITANCE

