|  |  |
| --- | --- |
|  | class X  {      //Class X Members  }    class Y  {      //Class Y Members  }    class Z extends X, Y  {      //Class Z Members  } |

**Answer :**  
In Java, a class can not extend more than one class. Class Z is extending two classes – Class X and Class Y. It is a compile time error in Java

|  |
| --- |
| class A  {      int i = 10;  }    class B extends A  {      int i = 20;  }    public class MainClass  {      public static void main(String[] args)      {          A a = new B();            System.out.println(a.i);      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse2)

**Answer :**  
10

**What will be the output of the below program?**

|  |  |
| --- | --- |
|  | class A  {      {          System.out.println(1);      }  }    class B extends A  {      {          System.out.println(2);      }  }    class C extends B  {      {          System.out.println(3);      }  }    public class MainClass  {      public static void main(String[] args)      { A a = new A()          C c = new C();      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse3)

Instance initializers java mai us waqt execute hota hai jab class ka instance banaya jaata hai

Isko {} sai define kiya hai in the class body without nay method or constructor name

Executed each time an instance of a class is created

Is tareekay sai execute kartay hain jaisay wo classs mai appear hotay hain

Agar koi class kisi class ko inherit kar rahi hai toh instance initializer of superclass execute hogi and then sub class

|  |
| --- |
| class A  {      String s = "Class A";  }    class B extends A  {      String s = "Class B";        {          System.out.println(super.s);      }  }    class C extends B  {      String s = "Class C";        {          System.out.println(super.s);      }  }    public class MainClass  {      public static void main(String[] args)      {          C c = new C();            System.out.println(c.s);      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse5)

**Answer :**  
Class A  
Class B  
Class C

class A

{

    public A()

    {

        System.out.println("Class A Constructor");

    }

}

class B extends A

{

    public B()

    {

        System.out.println("Class B Constructor");

    }

}

class C extends B

{

    public C()

    {

        System.out.println("Class C Constructor");

    }

}

public class MainClass

{

    public static void main(String[] args)

    {

        C c = new C();

    }

}

 **Class A:**

* A class ek simple constructor A() rakhti hai jo "Class A Constructor" print karta hai.

 **Class B extends A:**

* B class A ki subclass hai, isliye jab bhi B class ka object banaya jata hai, to pehle A class ka constructor (A()) execute hota hai, phir B class ka constructor (B()) chala jata hai.
* B class ka constructor "Class B Constructor" print karta hai.

 **Class C extends B:**

* C class B ki subclass hai, isliye jab C class ka object banaya jata hai, to A class ka constructor (A()), phir B class ka constructor (B()), aur phir C class ka constructor (C()) execute hote hain.
* C class ka constructor "Class C Constructor" print karta hai.

 **MainClass:**

* MainClass mein main method hai jismein C class ka ek object c banaya jata hai (new C()).
* Jab C class ka object banaya jata hai, to uske constructors ki hierarchy (A(), B(), C()) follow hoti hai.
* Is tarah se output mein pehle "Class A Constructor" print hota hai, phir "Class B Constructor", aur akhir mein "Class C Constructor".

|  |  |
| --- | --- |
| 2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | class X  {      static void staticMethod()      {          System.out.println("Class X");      }  }    class Y extends X  {      static void staticMethod()      {          System.out.println("Class Y");      }  }    public class MainClass  {      public static void main(String[] args)      {          Y.staticMethod();      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse9)

**Answer :**  
Class Y

**Below code is showing compile time error. Can you suggest the corrections?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | class X  {      public X(int i)      {          System.out.println(1);      }  }    class Y extends X  {      public Y()      {          System.out.println(2);      }  } |

**What is wrong with the below code? Why it is showing compile time error?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class A  {      public A()      {          System.out.println(1);            super();            System.out.println(2);      }  } |

[**View An**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse11)

In Java, when you define a constructor in a class, the first statement in that constructor must be either a call to another constructor of the same class (this(...)) or a call to a constructor of the superclass (super(...)). This call must be the first statement in the constructor body.

**Can you find out the error in the below code?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public class A  {      public A()      {          super();            this(10);      }        public A(int i)      {          System.out.println(i);      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse15)

**Answer :**  
A constructor can have either super() or this() but not both.

**What value the fields ‘i’ and ‘j’ will hold when you instantiate ‘ClassTwo’ in the below code?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | class ClassOne  {      static int i = 111;        int j = 222;        {          i = i++ - ++j;      }  }    class ClassTwo extends ClassOne  {      {          j = i-- + --j;      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse38)

**Answer :**  
i = -113  
j = 110

**39) When you instantiate a sub class, super class constructor will be also executed. True or False?**

[**View Answer**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse39)

**Answer :**  
True

**40) Does the below code written correctly? If yes, what will be the output?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | class One  {      int x = 2121;  }    class Two  {      int x = 1212;        {          System.out.println(x);      }  }    public class MainClass  {      public static void main(String[] args)      {          Two two = new Two();      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/#collapse40)

**Answer :**  
Yes, above code is written correctly. Output will be 1212.

**) Below class ABC doesn’t have even a single abstract method, but it has been declared as abstract. Is it correct?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | abstract class ABC  {      void firstMethod()      {          System.out.println("First Method");      }        void secondMethod()      {          System.out.println("Second Method");      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/#collapse1)

**Answer :**  
Yes, it is correct. abstract classes may or may not have abstract methods.

|  |
| --- |
| abstract class AbstractClass  {      abstract void abstractMethod()      {          System.out.println("First Method");      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/#collapse2)

**Answer :**  
Because, abstract methods must not have a body.

**Below code snippet is showing compilation error? Can you suggest the corrections?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | abstract class A  {      abstract int add(int a, int b);  }    class B extends A  {    } |

The compilation error in the provided code snippet occurs because class B extends abstract class A but does not provide an implementation for the abstract method add(int a, int b) that is defined in A.

To correct this error, class B must provide an implementation for the abstract method add(int a, int b). Here’s how you can modify the code:

java

Copy code

abstract class A {

abstract int add(int a, int b);

}

class B extends A {

@Override

int add(int a, int b) {

return a + b; // Implementing the abstract method

}

}

**Is the following program written correctly? If yes, what value “result” variable will hold if you run the program?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | abstract class Calculate  {      abstract int add(int a, int b);  }    public class MainClass  {      public static void main(String[] args)      {          int result = new Calculate()          {              @Override              int add(int a, int b)              {                  return a+b;              }          }.add(11010, 022011);      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/#collapse5)

**Answer :**  
Yes, program is written correctly. result = 20235.

**7) Can you identify the error in the below code?**

|  |  |
| --- | --- |
| 1  2  3  4 | abstract class AbstractClass  {      private abstract int abstractMethod();  } |

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/#collapse7)

**Answer :**  
abstract methods can’t be private.

**8) Can we declare protected methods in an interface?**

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/#collapse8)

**Answer :**  
No. only public methods are allowed in an interface.

No. abstract methods can’t be static.

**Is the below program written correctly? If yes, what will be the output?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | abstract class A  {      {          System.out.println("AAA");      }  }    abstract class B extends A  {      {          System.out.println("BBB");      }  }    class C extends B  {      {          System.out.println("CCC");      }  }    public class MainClass  {      public static void main(String[] args)      {          C c = new C();      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/2/#collapse12)

**Answer :**  
Yes, program is written correctly. Output will be,  
AAA  
BBB  
CCC

This sequence of output occurs because the instance initializer blocks are executed in the order of inheritance (from superclass to subclass) when an object of class C is instantiate

**s the below code written correctly?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | class A  {      void methodOfA()      {          abstract class B          {            }      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/2/#collapse20)

**Answer :**  
Yes, code is written correctly. Local inner classes can be abstra

**Abstract methods can be declared as final. True or False?**

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/2/#collapse17)

**Answer :**  
False. Abstract methods can’t be final.

**Can you identify the error in the below code?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | class X  {      public X()      {          System.out.println("Constructor One");      }        abstract X(int i)      {          System.out.println("Constructor Two");      }  } |

[**View Answer**](https://javaconceptoftheday.com/java-practice-coding-questions-on-abstract-classes/2/#collapse16)

**Answer :**  
Constructors can’t be abstract.