**1) Abstract class must have only abstract methods. True or false?**

False. Abstract methods can also have concrete methods.

**2) Is it compulsory for a class which is declared as abstract to have at least one abstract method?**

Not necessarily. Abstract class may or may not have abstract methods.

**3) Can we use “abstract” keyword with constructor, Instance Initialization Block and Static Initialization Block?**

No. Constructor, Static Initialization Block, Instance Initialization Block and variables can not be abstract.

**4) Why final and abstract can not be used at a time?**

Because, final and abstract are totally opposite in nature. A final class or method can not be modified further where as abstract class or method must be modified further. “final” keyword is used to denote that a class or method does not need further improvements. “abstract” keyword is used to denote that a class or method needs further improvements.

**5) Can we instantiate a class which does not have even a single abstract methods but declared as abstract?**

No, We can’t instantiate a class once it is declared as abstract even though it does not have abstract methods.

**6) Can we declare abstract methods as private? Justify your answer?**

No. Abstract methods can not be private. If abstract methods are allowed to be private, then they will not be inherited to sub class and will not get enhanced.

**7) We can’t instantiate an abstract class. Then why constructors are allowed in abstract class?**

It is because, we can’t create objects to abstract classes but we can create objects to their sub classes. From sub class constructor, there will be an implicit call to super class constructor. That’s why abstract classes should have constructors. Even if you don’t write constructor for your abstract class, compiler will keep default constructor.

**8) Can we declare abstract methods as static?**

No, abstract methods can not be static.

**9) Can a class contain an abstract class as a member?**

Yes, a class can have abstract class as it’s member.

**10) Abstract classes can be nested. True or false?**

True. Abstract classes can be nested i.e an abstract class can have another abstract class as it’s member.

**11) Can we declare abstract methods as synchronized?**

No, abstract methods can not be declared as synchronized. But methods which override abstract methods can be declared as synchronized.

**12) Can we declare local inner class as abstract?**

Yes. Local inner class can be abstract.

**13) Can abstract method declaration include throws clause?**

Yes. Abstract methods can be declared with throws clause.

**1) Can abstract class have constructors in Java?**

Yes, an abstract class can declare and define a constructor in Java. Since you can not create an instance of an abstract class, a constructor can only be called during [constructor chaining](http://javarevisited.blogspot.com/2012/12/constructor-chaining-in-java-calling-another-constructor.html), i.e. when you create an instance of the concrete implementation class.   
  
Now some interviewer, ask what is the purpose of a constructor, if you can not instantiate abstract class? Well, it can still be used to initialize common variables, which are declared inside an abstract class, and used by the various implementation.   
  
Also even if you don’t provide any constructor, the compiler will add [default no-argument constructor](http://javarevisited.blogspot.com/2012/12/what-is-constructor-in-java-example-chainning-overloading.html) in an abstract class, without that your subclass will not compile, since the first statement in any constructor implicitly calls super(), default superclass constructor in Java.

**2) Can abstract class implements interface in Java? do they require to implement all methods?**

Yes, an abstract class can implement an interface by using the implements keyword. Since they are abstract, they don’t need to implement all methods. It’s good practice to provide an abstract base class, along with an interface to declare Type. One example of this is java.util.List interface and corresponding java.util.AbstractList abstract class.   
  
Since AbstractList implements all common methods,  concrete implementations like [LinkedList](http://javarevisited.blogspot.com/2012/02/difference-between-linkedlist-vs.html) and [ArrayList](http://javarevisited.blogspot.com/2012/03/how-to-loop-arraylist-in-java-code.html) are free from the burden of implementing all methods, had they implemented List interface directly.   
  
It’s best of both worlds, you can get the advantage of interface for declaring type, and flexibility of abstract class to implement common behavior in one place. Effective Java has a nice chapter on how to use interface and abstract class in Java, which is worth reading.

**3) Can an abstract class be final in Java?**

No, an abstract class can not be final in Java. Making them final will stop the abstract class from being extended, which is the only way to use an abstract class. They are also opposite of each other, abstract keyword enforces to extend a class, for using it, on the other hand, [final keyword](http://javarevisited.blogspot.com/2011/12/final-variable-method-class-java.html) prevents a class from being extended. In real world also, abstract signifies incompleteness, while final is used to demonstrate completeness. Bottom line is, you can not make your class abstract and final in Java, at same time, it’s a compile time error.

**4) Can abstract class have static methods in Java?**

Yes, an abstract class can declare and define [static methods](http://javarevisited.blogspot.com/2011/11/static-keyword-method-variable-java.html), nothing prevents from doing that. But, you must follow guidelines for making a method static in Java, as it’s not welcomed in a object oriented design, because [static methods can not be overridden in Java](http://javarevisited.blogspot.com/2013/03/can-we-overload-and-override-static-method-java.html). It’s very rare, you see static methods inside abstract class, but as I said, if you have very good reason of doing it, then nothing stops you.

**5) Can you create an instance of abstract class?**

No, you can not create instance of abstract class in Java, they are incomplete. Even though, if your abstract class don’t contain any abstract method, you can not create instance of it. By making a class abstract,  you told compiler that, it’s incomplete and should not be instantiated. Java compiler will throw error, when a code tries to instantiate abstract class.

**6) Is it necessary for an abstract class to have an abstract method?**

No, It’s not mandatory for an abstract class to have any abstract method. You can make a class abstract in Java, by just using abstract keyword in class declaration. Compiler will enforce all structural restriction, applied to abstract class, e.g. now allowing to create any instance. By the way, it’s debatable whether you should have abstract method inside abstract class or interface. In my opinion, abstract class should have abstract methods, because that’s the first thing programmer assumes, when he see that class. That would also go nicely along principle of least surprise.

**7) Difference between abstract class and interface in Java?**

This is the most important and one of the classic Java Interview question. I don’t know, how many times I have seen this question at all most all levels of Java interviews. One reason, which makes this question interesting is the ability to produce example. It’s easy to answers questions on core OOPS concepts like [Abstraction](http://javarevisited.blogspot.com/2010/10/abstraction-in-java.html), [Encapsulation](http://javarevisited.blogspot.com/2012/03/what-is-encapsulation-in-java-and-oops.html), [Polymorphism](http://javarevisited.blogspot.com/2011/08/what-is-polymorphism-in-java-example.html) and [Inheritance](http://javarevisited.blogspot.com/2012/10/what-is-inheritance-in-java-and-oops-programming.html), but when it comes to subtle points like this, candidate more often fumbled. You can see this post for all syntactical difference between abstract class and interface, but it deserve a post on it’s own.

**8) When do you favor abstract class over interface?**

This is the follow-up of previous interview questions on abstract class and interface. If you know syntactical difference, you can answer this question quite easily, as they are the one, which drives the decision. Since it’s almost impossible to add a new method on a published interface, it’s better to use abstract class, when evolution is concern. Abstract class in Java evolves better than interface. Similarly, if you have too many methods inside interface, you are creating pain for all it’s implementation, consider providing an abstract class for default implementation. This is the pattern followed in Java collection package, you can see AbstractList provides default implementation for List interface.

**9) What is abstract method in Java?**

An abstract method is a method without body. You just declare method, without defining it and use abstract keyword in method declaration.  All method declared inside [Java Interface](http://javarevisited.blogspot.com/2012/04/10-points-on-interface-in-java-with.html) are by default abstract. Here is an example of abstract method in Java

                public void abstract printVersion();

Now, In order to implement this method, you need to extend abstract class and [override](http://javarevisited.blogspot.com/2011/12/method-overloading-vs-method-overriding.html) this method.

**10) Can abstract class contains main method in Java ?**

Yes, abstract class can contain [main method](http://javarevisited.blogspot.sg/2011/12/main-public-static-java-void-method-why.html), it just another static method and you can execute Abstract class with main method, until you don’t create any instance.

Read more: <https://javarevisited.blogspot.com/2013/04/10-abstract-class-and-interface-interview-question-java-answers.html#ixzz6uZxH0Nsa>

**1. What is Abstraction in Java?**

Ans: [Abstraction in Java](https://www.scientecheasy.com/2020/05/java-abstraction.html/) is a technique by which we can hide the data that is not required to users. It hides all unwanted data so that users can work only with the required data.

**2. How to achieve or implement Abstraction in Java?**

Ans: There are two ways to implement abstraction in java. They are as follows:

a) Abstract class (0 to 100%)  
b) Interface (100%)

**3. What is Abstract class in Java? How to define it?**

Ans: An abstract class in java is a class that is declared with an abstract keyword.

Example of Abstract class:



abstract class Test {

abstract void show();

}

**4. What is the difference between abstract class and concrete class?**

Ans: There are mainly two differences between an abstract class and concrete class. They are:

a) We cannot create an object of abstract class. Only objects of its non-abstract (or concrete) sub classes can be created.

b) It can have zero or more abstract methods that are not allowed in a non-abstract class (concrete class).

**5. What is Abstract in Java?**

Ans: Abstract is a non-access modifier in java that is applicable for classes, interfaces, methods, and inner classes.

**6. Can abstract modifier applicable for variables?**

Ans: No.

**7. What is Abstract method in Java?**

Ans: A method which is declared with abstract modifier and has no implementation (means no body) is called abstract method in java.

It does not contain any body. It has simply a signature declaration followed by a semicolon. It has the following general form as given below.



Syntax:

abstract type MethodName(arguments); // No body

​

For example:

  abstract void msg(); // No body.

​

**8. Can an abstract method be declared as static?**

Ans: No.

**9. Can an abstract method be declared with private modifier?**

Ans: No, it cannot be private because the abstract method must be implemented in the child class. If we declare it as private, we cannot implement it from outside the class.

**10. What is Concrete method in Java?**

Ans: A concrete method in Java is a method which has always the body. It is also called a complete method in java.

**11. When to use Abstract class in Java?**

Ans: An abstract class can be used when we need to share the same method to all non-abstract sub classes with their own specific implementations.

**12. When to use Abstract method in Java?**

Ans: An abstract method can be used

a) When the same method has to perform different tasks depending on the object calling it.  
b) When you need to be overridden in its non-abstract subclasses.

**13. Is abstract class a pure abstraction in Java?**

Ans: No, It provides 0 to 100% abstraction.

**14. Is it possible to create an object of abstract class in Java?**

Ans: No. It is not possible but we can create an object of its subclass.

**15. Is it possible that an abstract class can have without any abstract method?**

Ans: Yes.

**16. Can an abstract class have constructor?**

Ans: Yes.

**17. Is abstract class allow to define private, final, static, and concrete methods?**

Ans: Yes.

**18. Is it possible to achieve multiple inheritance through abstract class?**

Ans: No.

**19. Can we make an abstract class without abstract keyword?**

Ans: No, a class must be declared with abstract keyword to make an abstract class.

**20. Can we define an abstract method inside non-abstract class (concrete class)?**

Ans: No, we cannot define an abstract method in non-abstract class.

For example:



class Test {

abstract void show();

}

The above code will generate a compile-time error.

**21. Which among the following code have errors?**



a) abstract class A {

    void m1();

  }

b) public class A {

    abstract void m1();

  }

c) abstract public class A {

   abstract void m1();

  }

d) abstract public class A {

    void m1() { }

  }

e) public abstract class A {

   abstract void m1();

   A(){ }

   void m2() { }

  }

f) public abstract class A {

    abstract int x = 100;

    abstract void m1();

    abstract void m2();

  }

g) public abstract class A {

   abstract void m1();

  }

  public class Test {

  public static void main(String[] args) {

   A a = new A();

  }

  }

h) public abstract class A {

   abstract void m1();

   A(){ }

   static void m2() {System.out.println("Hello Java!"); }

  }

   public class B extends A {

    void m1(){

       A.m2();

    }

  }

​

i) public abstract class A {

   abstract void m1();

  private A(){ }

  }

  public class B extends A { }

Ans: a, b, f, g, i.

**22. Will the following code compile successfully? If yes, what will be the output of program?**



1

public abstract class A {

2

 abstract void m1(A a);

3

}

4

public class B extends A {

5

void m1(A a){

6

 System.out.println("One");

7

}

8

}

9

public class C extends B {

10

void m1(B b){

11

System.out.println("Two");

12

super.m1(new B());

13

}

14

}

15

public class Test {

16

public static void main(String[] args){

17

C c = new C();

18

c.m1(new B());

19

}

20

}

Ans: Yes, the above code will be compiled successfully. The output of above program is Two, One.

**23. What will happen if we do not override all abstract methods in subclass?**  
Or, what will happen if we do not provide implementation for all abstract methods in subclass?

Ans: Java compiler will generate compile time error. We will have to override all abstract methods in subclass.

**24. What is the difference between Abstraction and Encapsulation?**

Ans: Abstraction hides the implementation details from users whereas, [encapsulation](https://www.scientecheasy.com/2020/07/encapsulation-in-java.html/) wraps (binds) data and code into a single unit.

**25. Why abstract class has constructor even though you cannot create object?**

Ans: We cannot create an object of abstract class but we can create an object of subclass of abstract class. When we create an object of subclass of an abstract class, it calls the constructor of subclass.

This subclass constructor has a super keyword in the first line that calls constructor of an abstract class. Thus, the constructors of an abstract class are used from constructor of its subclass.

If the abstract class doesn’t have constructor, a class that extends that abstract class will not get compiled.

**26. Why should we create reference to superclass (abstract class reference)?**

Ans: We should create a reference of the superclass to access subclass features because superclass reference allows only to access those features of subclass which have already declared in superclass.

If you create an individual method in subclass, the superclass reference cannot access that method. Thus, any programmer cannot add their own additional features in subclasses other than whatever is given in superclass.

**27. What is the advantage of Abstract class in Java?**

Ans: The main advantages of using abstract class are as follows:

* Abstract class makes programming better and more flexible by giving the scope of implementing abstract methods.
* Programmer can implement abstract method to perform different tasks depending on the need.
* We can easily manage code.

**28. Will the code compile successfully? If yes, what will be the output?**



1

public abstract class A {

2

abstract void m1();

3

}

4

public class B extends A {

5

void m1(){

6

 System.out.println("m1 in class B");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

B b = new B();

12

b.m1();

13

}

14

}

15

​

Ans: Yes, the code will be compiled successfully. The output is m1 in class B.

**29. Consider the below given code.**



1

public abstract class A {

2

abstract void m1();

3

void m2(){

4

System.out.println("One”);

5

}

6

}

7

​

**How to call m2() method in the above code?**

Ans: Make it static and call as A.m2();

**30. Identify the errors in the following code.**



1

public abstract class A {

2

abstract void m1();

3

void m2(){

4

System.out.println("One");

5

}

6

}

7

public class B extends A {

8

void m2(){

9

 System.out.println("Two");

10

}

11

}

12

public class Test {

13

public static void main(String[] args) {

14

B b = new B();

15

b.m2();

16

}

17

}

Ans: The abstract method m1() has been not implemented (overridden) in subclass B. Therefore, we will get a compile-time error.

**31. Will the given code compile successfully? If yes, how?**

a)



1

public abstract class A {

2

abstract void m1();

3

}

4

public class B extends A {

5

private void m1(){

6

 System.out.println("One");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

B b = new B();

12

b.m1();

13

}

14

}

15

​

Ans: No, the code will not be compiled successfully because we cannot reduce the visibility of the inherited method from A.

b)



1

public abstract class A {

2

abstract void m1();

3

}

4

public class B extends A {

5

protected void m1(){

6

 System.out.println("One");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

B b = new B();

12

b.m1();

13

}

14

}

15

​

Ans: Yes, the code will be compiled successfully. The output is One.

c)



1

public abstract class A {

2

public abstract void m1();

3

}

4

public class B extends A {

5

protected void m1(){

6

 System.out.println("One");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

B b = new B();

12

b.m1();

13

}

14

}

15

​

Ans: No, compile-time error because we cannot reduce the visibility of inherited method m1() from A.

d)



1

public abstract class A {

2

private abstract void m1();

3

}

4

public class B extends A {

5

protected void m1(){

6

 System.out.println("One");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

B b = new B();

12

b.m1();

13

}

14

}

15

​

Ans: Abstract method m1() in class A cannot be private.

e)



1

public abstract class A {

2

protected abstract void m1();

3

}

4

public class B extends A {

5

void m1(){

6

 System.out.println("One");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

B b = new B();

12

b.m1();

13

}

14

}

15

​

Ans: No, compile time error.

f)



1

public abstract class A {

2

protected abstract void m1();

3

}

4

public class B extends A {

5

protected void m1(){

6

 System.out.println("One");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

B b = new B();

12

b.m1();

13

}

14

}

15

​

Ans: Yes, the code will be compiled successfully. The output is One.

g)



1

public abstract class A {

2

static abstract void m1();

3

}

4

public class B extends A {

5

void m1(){

6

 System.out.println("One");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

B b = new B();

12

b.m1();

13

}

14

}

15

​

Ans: No, compile-time error because we cannot set non-access modifiers with abstract method m1() in class A.

**32. Will the below code compiled successfully? If yes, what will be the output of the program?**



1

public abstract class A {

2

abstract void m1();

3

abstract void m2();

4

}

5

public class B extends A {

6

void m1(){

7

 System.out.println("One");

8

}

9

void m2(){

10

 System.out.println("Two");

11

}

12

void m3() {

13

 System.out.println("Three");

14

}

15

}

16

public class Test {

17

public static void main(String[] args) {

18

A a = new B();

19

a.m1();

20

a.m2();

21

a.m3();

22

}

23

}

24

​

Ans: No, compile time error because we cannot call m3() method defined in class B.

**33. Will the below code be compiled successfully? If yes, what will be the output of the following program?**



1

public abstract class A {

2

abstract void m1();

3

A(){ System.out.println("Hello"); }

4

static void m2() {System.out.println("Hello Java!"); }

5

}

6

public class B extends A {

7

void m1(){

8

 A.m2();

9

}

10

}

11

public class Test {

12

public static void main(String[] args) {

13

B b = new B();

14

 b.m1();

15

}

16

}

17

​

Ans: Yes, Output is Hello, Hello Java!

**34. What will be the output of the following program?**



1

public abstract class A {

2

abstract void m1();

3

A(){ System.out.println("Red"); }

4

static void m2() {System.out.println("Orange"); }

5

}

6

public class B extends A {

7

void m1(){

8

 A.m2();

9

 System.out.println("Pink");

10

}

11

}

12

public class C extends B {

13

void m3(){

14

   System.out.println("Green");

15

}

16

}

17

public class Test {

18

public static void main(String[] args) {

19

C c = new C();

20

c.m3();

21

c.m1();

22

}

23

}

24

​

Ans: Output: Red, Green, Orange, Pink.

**35. Identify the error in the below code and correct the error.**



1

public abstract class A {

2

abstract void m1(int x, int y);

3

}

4

public class B extends A {

5

void m1(){

6

 System.out.println(" ");

7

}

8

}

9

​

Ans: The class B must implement inherited abstract method as void m1(int x, int y) { System.println.out(” “);}

**36. Correct the given code and find out the output?**



1

public abstract class A {

2

abstract void m1(int x, double y);

3

}

4

public class B extends A {

5

void m1(double y, int x){

6

 System.out.println("Hello Java!");

7

}

8

}

9

public class Test {

10

public static void main(String[] args) {

11

A a = new B();

12

a.m1(20, 20.50);

13

}

14

}

15

​

Ans: Correction code is:



1

public class B extends A {

2

void m1(int x, double y){

3

 System.out.println("Hello Java!");

4

}

5

}

6

​

The output is “Hello Java!”.

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



1

public abstract class A {

2

private int x;

3

A(int x){

4

 System.out.println("Value of x: " +x);

5

}

6

 abstract void m1(int x, double y);

7

}

8

public class B extends A {

9

private int y;

10

B(int y){

11

  super(10);

12

  System.out.println("Value of y: " +y);

13

}

14

void m1(int x, double y){

15

 System.out.println("One");

16

}

17

void m2(){

18

 System.out.println("Two");

19

 this.m1(5, 10.50);

20

}

21

}

22

public class C extends B {

23

C(){

24

super(30);

25

}

26

void m1(int x, double y){

27

  super.m1(10, 15.15);

28

  System.out.println("Three");

29

}

30

}

31

public class Test {

32

public static void main(String[] args){

33

B b = new C();

34

b.m1(10, 20.50);

35

b.m2();

36

}

37

}

38

​

Ans: Output is:

Value of x: 10  
Value of y: 30  
One  
Three  
Two  
One  
Three

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



1

public abstract class A {

2

abstract void m1(int x, double y);

3

abstract void m2(String name);

4

}

5

public class B extends A {

6

void m1(int x, double y){

7

 System.out.println("One");

8

}

9

void m2(String name){

10

 System.out.println("Two");

11

}

12

}

13

public class C extends B {

14

static void m1(){

15

super.m1(20, 30);

16

}

17

}

18

public class Test {

19

public static void main(String[] args){

20

C.m1();

21

}

22

}

23

​

Ans: The code cannot be compiled because super keyword cannot be used in static method.

**39. What will be the output of the following program?**



1

public abstract class A {

2

abstract void m1(int x, double y);

3

abstract void m2(String name);

4

}

5

public class B extends A {

6

void m1(int x, double y){

7

 System.out.println("One");

8

}

9

void m2(String name){

10

 System.out.println("Two");

11

}

12

void m3(){

13

System.out.println("Three");

14

}

15

void m4(){

16

System.out.println("Four");

17

}

18

}

19

public class C extends A {

20

B b;

21

void m1(int x, double y){

22

b = new B();

23

b.m3();

24

}

25

void m2(String name){

26

 b = new B();

27

 b.m4();

28

}

29

}

30

public class Test {

31

public static void main(String[] args){

32

A a = new C();

33

a.m1(20, 30.0);

34

a.m2("abc");

35

}

36

}

37

​

Ans: Output is Three, Four

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



1

public abstract class A {

2

abstract void m1(A a);

3

abstract void m2(A a);

4

}

5

public class B extends A {

6

void m1(A a){

7

 System.out.println("One");

8

}

9

void m2(A a){

10

 System.out.println("Two");

11

}

12

}

13

public class C extends B {

14

void m1(B b){

15

System.out.println("Three");

16

super.m1(new B());

17

}

18

void m2(B b){

19

System.out.println("Four");

20

super.m2(new B());

21

}

22

}

23

public class Test {

24

public static void main(String[] args){

25

C c = new C();

26

c.m1(new B());

27

c.m2(new B());

28

}

29

}

30

​

Ans: Output: Three, One, Four, Two.

**What is an abstraction**

**in Java?**

Java is an object-oriented programming language. abstraction is an oops concept. Abstraction is a process of hiding the implementation details and showing only the functionality of the application. We can be achieved abstraction in java by using abstract class and interface.

Abstract class vs abstraction 0 to 100%.

Interface vs abstraction 100%.

**Abstract class in Java.**

A Java class which is defined with an abstract keyword is called abstract class. An abstract class is something which is incomplete and you cannot create an instance of the abstract class.If you want to use it you need to extend it and complete it.  
Syntax:

abstract **class** **class**-name{}

**Abstract method in Java.**

A method without implementation is called an abstract method. the abstract method is just prototype of the method or you can signature.

abstract return\_type method\_name ();

**Abstraction in java with example.**

public **class** Main{

public static **void** main(String args[]){

FeedBack obj= **new** FeedBack();

obj.show();

}

}

abstract **class** Message{

public abstract **void** show();

}

**class** FeedBack **extends** Message{

public **void** show(){

System.out.println("Hello CodeBun");

}

}

**Important points you should remember**

* If you define any method as an abstract than class must be abstract.
* If you are extending an abstract class so you need to implement all the method of that class.
* In Java, you cannot create the instance of an abstract class.
* Use abstraction if you know something needs to be in class but the implementation of that varies.
* abstract is a keyword in Java.

**Java abstraction interview questions and answers with example**

**Can we define constructors in an abstract class?**

An abstract class can have a constructor but it cannot be instantiated. If you do not define your own constructor compiler will give a default constructor to the Abstract class.

**why we use a constructor in abstract class in java?**

A constructor is used in the abstract method because some time we need to initialize variables in subclass like below code.

class Main **extends** User {

String education;

Main(String education, String name, int age) {

super(name, age);

**this**.education = education;

}

public static **void** main(String[] args) {

Main obj = **new** Main("MCA", "rahul", 22);

System.out.println(obj.education + " " + obj.name + " " + obj.age);

}

@Override

public **void** msg() {

// TODO Auto-generated method stub

// write your code....

}

}

abstract **class** User {

String name;

int age;

User(String name, int age) {

**this**.name = name;

**this**.age = age;

}

public abstract **void** msg();

}

**Can we define an abstract class as a final class?**

My answer is direct no because as abstraction says you are going implement the methods, and as you know if we define as a final we can’t change anything. They are also opposite of each other, abstract keyword enforces to extend a class, for using it, on the other hand, final keyword prevents a class from being extended.

**Is it necessary for an abstract class to have an abstract method?**

No, it’s not necessary because we can define concrete and non-concreate method in abstract class.

**Difference between abstract class and interface in Java.**

An abstract class can contain abstract and non-abstract method but in interface only abstract method its true before java 8 version now we can define. [please read in detail.](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class)

**Can we define the main class as an abstract class?**

Yes, the abstract class can contain the main method.

abstract **class** Main {

public static **void** main(String[] args) {

System.out.println("hello codebun");

}

}

**Can we declare abstract methods as private?**

No, If abstract methods are allowed to be private, then they will not be inherited to subclass.

Question: How To Describe Abstraction In Interview?

* Abstraction is a process of hiding the implementation details and showing only functionality to the user.
* A method that is declared as abstract and does not have implementation is known as abstract method.
* There are two ways to achieve abstraction in java  
  1- By Abstract class (0 to 100%)  , 2- By Interface (100%)

Question: What is abstraction and abstract class in Java?

**Abstraction:**

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstaction

There are two ways to achieve abstraction in java

Abstract class (0 to 100%)  
Interface (100%)

**Abstract class in Java:**

A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented. It cannot be instantiated.

Example abstract class

abstract class A{}

**Abstract method:**

A method that is declared as abstract and does not have implementation is known as abstract method.  
Example abstract method

abstract void printStatus();//no body and abstract

Question: Can abstract class have constructors in Java?

Yes, abstract class can declare and define constructor in Java. Since you can not create instance of abstract class, constructor can only be called during constructor chaining, i.e. when you create instance of concrete implementation class.

Question: Can abstract class implements interface in Java? does they require to implement all methods?

Yes, abstract class can implement interface by using implements keyword. Since they are abstract, they don’t need to implement all methods.

It’s good practice to provide an abstract base class, along with an interface to declare Type.

Question: Can abstract class be final in Java?

No, abstract class can not be final in Java. Making them final will stop abstract class from being extended, which is the only way to use abstract class.

They are also opposite of each other, abstract keyword enforces to extend a class, for using it, on the other hand, final keyword prevents a class from being extended.

In real world also, abstract signifies incompleteness, while final is used to demonstrate completeness. Bottom line is, you can not make your class abstract and final in Java, at same time, it’s a compile time error.

Question: Can you create instance of abstract class?

No, you can not create instance of abstract class in Java, they are incomplete.

Even though, if your abstract class don’t contain any abstract method, you can not create instance of it.

By making a class abstract, you told compiler that, it’s incomplete and should not be instantiated. Java compiler will throw error, when a code tries to instantiate abstract class.

Question: Is it necessary for abstract class to have abstract method?

No, It is not mandatory for an abstract class to have any abstract method. We can make a class abstract in Java, by just using abstract keyword in its declaration.

Question: What is abstract method in Java?

An abstract method is a method without body.  It  uses abstract keyword in method declaration. All method declared inside Java Interface are by default abstract. Here is an example of abstract method in Java

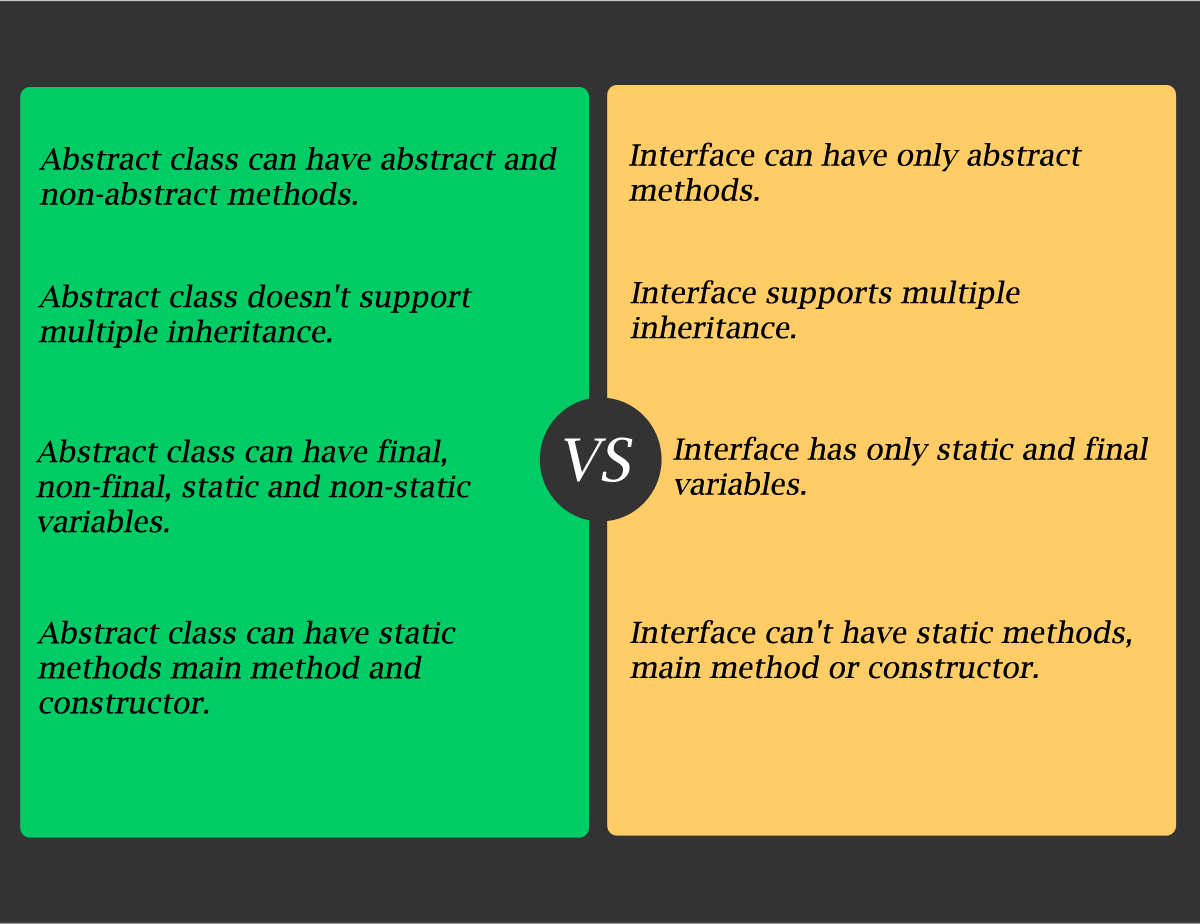
public void abstract printVersion();

Now, In order to implement this method, We need to extend this abstract class and override its method.

Question: Can abstract class contains main method in Java?

Yes, abstract class can contain main method, it just another static method and you can execute Abstract class with main method, until you don’t create any instance.

Question: Abstract Class Vs Interface



**(1) What is abstraction in java?**

**In java, Abstraction means show functionality and hide complexity or internal details or hide implementation details to the user is know as abstraction in java.**

**For example:**

**The best example in the world of abstraction is ATM machine where we can withdraw or transfer money easily but how it happens? we don't know. We don't know internal details or implementation details.**

**(2) How to achieve abstraction in java?**

**There are two ways in java we can achieve abstraction.**

* **By using abstract class(0 to 100%).**
* **By using interface(100%).**

**(3) What is abstract class in java?**

**When we declared any class with "abstract" keyword is known as abstract class. In abstract class we can keep abstract method(without body) and non-abstract method(with body).**  
 **For example:**  
 **abstract class Demo**  
**{**  
**abstract void show();//abstract method**  
**void show(){}//non-abstract method**  
**}**

**(4) Can we create instance of abstract class?**

**No, We cannot create an instance of an abstract class.**

**(5) Can we define abstract class without abstract method?**

**Yes, We can define abstract class without abstract method.**

**(6) Can we declare abstract method in non-abstract class?**

**No, We can't declare abstract method in non-abstract class.**  
 **For example:**  
 **class Demo**  
**{**  
**abstract void show();**  
**}**  
 **Above example will throw compile time error.**

**(7) What is interface in  java?**

**An interface in java is a blueprint of a class that have static constants and abstract method by default.**

**(8) Why we use interface in java?**

**In java we use interface so that we can achieve fully abstraction because through abstract class we can't achieve full abstraction.**

**(9) Can we create instance of interface?**

**No, We cannot create object of both an interface and abstract class.**

**(10) Can we declare abstract method as static?**

**No, We can't use static keyword with abstract method.**

**(11) Can we declare abstract method as final?**

**No, We cannot use final keyword with abstract class.**

**(12) Can we declare abstract method as private?**

**No, We cannot declare abstract method as private.**

**(13) Can we use public, protected and default modifiers with abstract method?**

**Yes, We can use public, protected and default modifiers with abstract method.**

**(14) Can we declare local inner class as abstract?**

**Yes, We can declare local inner class as abstract.**

**(15) Method in interface are by default public and abstract. true or false?**

**Yes, It is true.**

**(16) Data member in interface are by default public, static, and final. true of false?**

**Yes, It is true.**

**(17) Can abstract class implements interface in java?**

**Yes, Abstract class can implement interface.**

**(18) Can we use abstract keyword with constructor?**

**No, We can't use abstract keyword with constructor.**

**(19) Abstract classes can be nested. true or false?**

**Yes, Abstract classes can be nested.**

**(20) Can abstract class have constructor in java?**

**Yes, We can declare constructor in abstract class.**

**(21) Difference between abstraction and encapsulation in java?**

**There are may**[**differences between abstraction and encapsulation in java**](https://javatutorial95.blogspot.com/2017/05/java-encapsulation.html)**.**

**(22) What is the difference between abstract class and interface?**

**You can click on this link to see what is the**[**difference between abstract class and interface in java**](https://javatutorial95.blogspot.com/2017/07/difference-between-abstract-class-and-interface-in-java.html)**with example**

**(23) Can abstract method declaration include throws clause?**

**Yes.**

**(24) What will happen if we do not override all the abstract methods in sub-class?**

**It will throw compile-time error. We have to override all the abstract method in sub-class. If you do not want to override all the abstract method in sub-class then you have to make your sub-class as abstract class.**

**What are the Abstract classes in Java? Or Explain Abstract classes?**

* A class with an abstract keyword in the class declaration is known as an abstract class in Java
* Unlike a class, an abstract class can contain both abstract methods as well as concrete methods (i.e.; methods with braces and method body or method implementation)

**What is an Abstract method in Java?**

* A method declaration preceded/prefixed with the abstract keyword with nobody or no implementation detail which ends its method signature with a semicolon(;) is known as an abstract method

**Whether abstract class compiles successfully if it contains both concrete & abstract methods together?**

* Yes, abstract class compile successfully as it can contain both abstract methods and concrete

**Write an example of the abstract classes containing both concrete & abstract method?**

* **AbstractExample.java**
* **package** com.hcl.tmobile.submitservicechange;
* **public** **abstract** **class** AbstractExample
* **{**
* String demoString;
* **static** int demoCounter;
* // default no-arg constructor
* **AbstractExample()**
* **{**
* // do some initialization logic here
* **}**
* // abstract method declaration in abstract class
* **abstract** **void** **myAbstractMethod()**;
* // concrete method definition in abstract class
* **void** **myConcreteMethod()**
* **{**
* System.out.println**(**"AbstractExample: This is my concrete method in abstract class"**)**;
* **}**
* **}**

**Check Also:**[**Arithmetic Exception In Java**](https://www.softwaretestingo.com/arithmetic-exception/)

**What happens if subclass extending abstract classes doesn’t override abstract methods?**

* Compiler throws an error to implement all abstract methods
* **Compiler-time error:** The type AbstractExampleMain must implement the inherited abstract method AbstractExample.myAbstractMethod()

**What all options available for subclass extending an abstract class to not override abstract methods?**

* There are 2 options; either implement all abstract methods or make the extending class as abstract
* This way, the next extending class must provide implementation or else again it can be marked as abstract
* **Options:**1. Add unimplemented methods  
  2. Make type ‘ExtendingClass’ abstract
* **Note:** See above screen capture from the previous question for details

**Can abstract class implements interface?**

* Yes, an abstract class can implement an interface and this is allowed

**Can an abstract class be defined without any abstract methods?**

* Yes, a class can be declared with abstract keyword even if it doesn’t get 1 abstract method
* But vice-versa is not true; means if a class contains abstract methods then the class has to be declared with abstract keyword

**Whether it is mandatory to have abstract methods in an abstract class? If not, why such a design is required?**

* It’s not mandatory to have abstract methods in an abstract class
* Even without a single abstract method in a class can be declared as abstract
* This is to flag compiler that this class is not for instantiation

**Can we define an abstract class without an abstract method? Why it is needed?**

* Yes, a class can be declared with abstract keyword even if it doesn’t get 1 abstract method
* This is to flag compiler that this class is not allowed to instantiate

**Can we define an abstract class without an abstract keyword in the class declaration?**

* No, an abstract keyword is required at class declaration to declare an abstract class

**Whether class compiles successfully if a class contains abstract methods and no abstract keyword at class declaration?**

* Compiler throws error
* **Compiler-time error:** This method requires a body instead of a semicolon

**Can we define constructor inside an abstract class?**

* Yes, we can define constructor inside an abstract class
* Both default & parameterized constructors are allowed inside an abstract class

**Can an abstract class be instantiated?**

* No, an abstract class cannot be instantiated
* Instantiating abstract class throws a compile-time error
* **Compiler-time error:**Cannot instantiate the type <Abstarct\_Class\_Name>

**Why abstract classes cannot be instantiated if constructors can be defined inside an abstract class?**

* True, an abstract class cannot be instantiated; still having instance data members and constructor
* This is to instruct the compiler that no one should create an object of type abstract class
* The reason is, every object has got some default behavior and specific behavior. In this case, an abstract class is apt
* So, we can put more common & general behavior with concrete method implementation and later extend (sub-classing) class can provide a specific implementation for abstract methods in their own way

**Can an abstract class be final?**

* No, an abstract class cannot be final
* Abstract methods need to be implemented; therefore it is overridden in the subclass
* But by marking final, we are restricting it to override
* **A compile-time error** will be thrown: The abstract method display in type <abstract-class-name> can only set a visibility modifier, one of public or protected

**Can we declare an abstract method with a static modifier inside an abstract class?**

* No, an abstract class cannot be **static**
* **A compile-time error** will be thrown: The abstract method display in type <abstract-class-name> can only set a visibility modifier, one of public or protected

**Can we declare a concrete (non-abstract) method with the final modifier inside an abstract class?**

* Yes, the concrete method can be declared with **the final** modifier

**Can we declare an abstract method with a private modifier inside an abstract class?**

* No, an abstract class cannot be declared with private accessibility
* A compilation error will be thrown with below error
* **Compile-time error**: The abstract method display in type <abstract-class-name> can only set a visibility modifier, one of public or protected

**Why modifiers such as final, static & private are not allowed for abstract methods declared in abstract class?**

* **Final:** as sub-class needs to provide method implementation for all abstract methods inside an abstract class, therefore abstract cannot be marked as final
* **Static:** abstract methods belong to instance not a class, therefore it cannot be marked as static
* **Private:** abstract methods need to be overridden in the sub-class for this we need wider accessibility
* By marking abstract method declaration with final or static or private modifier –> results in a compilation error
* **Compile-time error**: The abstract method display in type <abstract-class-name> can only set a visibility modifier, one of public or protected

**What are all modifiers allowed for abstract method declaration?**

* public and protected access modifiers are allowed for abstract method declaration

**What are all modifiers allowed for an abstract class declaration?**

* private, static and final modifiers are NOT allowed for abstract method declaration

**Can we define private constructor inside an abstract class?**

* Yes, it is allowed to have private constructor inside an abstract class

**Is it ok to declare an abstract method inside a non-abstract class?**

* No, it is not allowed to have an abstract method inside a concrete class
* If there any abstract method, the class must be marked with the abstract modifier

**Can we declare static fields inside an abstract class?**

* Yes, static fields and static methods are allowed to declare inside an abstract class

**Can we define static methods giving concrete implementation inside an abstract class?**

* Yes, static methods are allowed inside an abstract class

**Whether an abstract method inside the abstract class can throw exceptions?**Or **Can abstract method declaration include throws clause?**

* Yes, abstract methods can throw an exception
* See below screen capture

**Can abstract class contain the main() method and starts the execution? Write a program?**

* Yes, **the main()** method allowed inside the abstract class; also we can execute
* **AbstractExampleMain.java**
* **package** in.software.testing.abstractclass.example;
* // abstract class
* **public** **abstract** **class** AbstractExampleMain
* **{**
* // abstract method throwing exception
* **abstract** **void** **display()** **throws** ClassCastException;
* **static** **void** **staticDisplay()** **{**
* System.out.println**(**"Displaying: main() method execution"**)**;
* **}**
* **public** **static** **void** **main(**String arg**[])** **{**
* **staticDisplay()**;
* **}**
* **}**

**Output**

Displaying: main() method execution

**Q) Can we an overload abstract method in Java?**

* Yes, abstract methods can be overloaded
* And its extending class will provide an implementation for all abstract methods

**What is an Abstract Class and what is its purpose?**  
**Ans:** A class that doesn’t provide complete implementation is defined as an abstract class. Abstract Classes enforce abstraction. Any class with an abstract method is automatically abstract itself and must be declared as such. A class may be declared abstract even if it has no abstract methods. This prevents it from being instantiated. Abstract class must be extended/subclassed (to be useful). It serves as a template. A class that is abstract may not be instantiated (ie. you may not call its constructor), abstract class may contain static data.

**Can an abstract class be declared final?**  
**Ans:** No. An abstract class is of no use unless it is inherited. Otherwise, it just results in a compile-time error.

**What is the use of an abstract variable?**  
**Ans:** There is nothing called Abstract Variables as variables can’t be declared as abstract. Only classes and methods can be declared as abstract.

**Can you create an object of an abstract class?**  
**Ans:** No. Abstract classes can’t be instantiated.

**Can an abstract class be defined without any abstract methods?**  
**Ans:** Yes, it is possible. This is performed, to avoid instance creation of the class.

**What is an abstract method?**  
**Ans:** An abstract method is a method whose implementation is deferred to a subclass.

**Can an abstract class be final?**  
**Ans:** An abstract class may not be declared as final.

**What is the final class?**  
**Ans:** A final class can’t be extended, i.e. the final class may not be subclassed. A final method can’t be overridden when its class is inherited. You can’t change the value of a final variable (is a constant).

**What if the main() method is declared as private?**  
**Ans:** The program compiles properly but at runtime, it will give the “main() method not public.” message.

**What if the static modifier is removed from the signature of the main() method?**  
**Ans:** Program compiles. But at runtime throws an error “NoSuchMethodError”.

**What if I write static public void instead of the public static void?**  
**Ans:** Program compiles and runs properly.

**What is the difference between abstract class and interface?  
Ans:**

* All the methods declared inside an interface are abstract whereas abstract class must have at least one abstract method and others may be concrete or abstract.
* In abstract class, keyword abstract must be used for the methods whereas interface we need not use that keyword for the methods.
* Abstract class must have subclasses whereas interface can’t have subclasses.

Can I create an abstract variable?

No, abstract keyword is applicable only for classes and methods.

What is an abstract class?

A class that has only partial implementation and has abstract methods.

Can an abstract class be instantiated?

No. Abstract classes cannot be instantiated since it is abstract and not concrete.

Can a abstract class be defined without any abstract methods?

**Yes.** It cannot be instantiated since declaring a class as abstract restrict it to be not instantiated on its own.

What is the difference between an Interface and an Abstract class?

* All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.
* A class may implement a number of Interfaces, but can extend only one abstract class.
* In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
* Abstract classes can implement interfaces without even providing the implementation of interface methods.
* Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
* Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
* An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated, but can be invoked if it contains a main method.

Can an abstract method be declared static in Java?

No. An abstract method cannot be static. You cannot override a static method, so making it abstract would not make any sense.

Java 8, difference between abstract class and Interface.

Both abstract class and interface are used for abstraction by hiding the internal implementation of the feature and only showing the functionality to the users.

**method types**: Interface can have only abstract methods. From Java 8, interface can have default and static methods also. Abstract class can have abstract and non-abstract methods.

**Final Variables**: Variables declared in a Java interface are by default final. An abstract class may contain non-final variables.

**Type of variables**: Abstract class can have final, non-final, static and non-static variables. Interface has only static and final variables.

**Accessibility of Data Members**: Members of a Java interface are public by default. A Java abstract class can have class members like private, protected, etc.

**Implementation**: Abstract class can provide the implementation of interface while Interface cannot provide the implementation of abstract class.

**Inheritance vs Abstraction**: A Java interface can be implemented using keyword "implements" and abstract class can be extended using keyword "extends".

**Multiple implementation**: An interface can extend another Java interface only, an abstract class can extend another Java class and implement multiple Java interfaces.

Can abstract class have Constructor?

Yes, but its object cannot be created by calling the constructor instead it is invoked during constructor chaining.

Can abstract class implement an interface in Java?

Yes, an abstract class can implement the interface by using implements keyword. Since they are abstract, they **dont need to implement all methods**.

Can an abstract class be final in Java?

No.

Can abstract class have a static method in Java?

Yes, the abstract class can declare and define static methods. It is not recommended though as per design principles.

What is an abstract method in Java?

An abstract method is a method without a body. You just declare method, without defining it and use abstract keyword in method declaration. All method declared inside Java Interface are by default abstract. Here is an example of an abstract method in Java.

**public** **void** **abstract** **anExampleAbstractMethod**();

Can an abstract class have the main method?

Yes, it is also a static method and it can execute the abstract class.

**public** **abstract** **class** **AbstractClassMain** {

**public** **static** **void** **main**(String[] args) {

System.out.println("Hello from abstract class");

}

}

Can we call an abstract method from a non-abstract method?

Yes.

Can we mark a local inner class as abstract?

Yes.

**public** **class** **OuterClass** {

**public** **void** **instanceMethod**(){

**abstract** **class** **InnerClass**{

}

}

}

Below is an example that illustrates the usage of local inner abstract class.

**public** **class** **SendNotification** {

**public** **enum** NotifyBy {

MESSAGE, EMAIL, PHONECALL

}

**public** Runnable **createNotification**(NotifyBy type) {

**abstract** **class** **ProtoType** **implements** Runnable {

**public** **void** **run**() {

prepareContent();

sendNotification();

}

**abstract** **void** **sendNotification**();

**private** **void** **prepareContent**() {

// notification message content

}

}

**switch** (type) {

**case** **EMAIL:** {

**return** **new** **ProtoType**() {

**void** **sendNotification**() {

System.out.println("Sent using email.");

}

};

}

**case** **MESSAGE:** {

**return** **new** **ProtoType**() {

**void** **sendNotification**() {

System.out.println("communicated by text message.");

}

};

}

}

**return** **new** **ProtoType**() {

**void** **sendNotification**() {

System.out.println("communicated by phone call.");

}

};

}

**public** **static** **void** **main**(String[] args) {

**new** **Thread**(**new** SendNotification().createNotification(NotifyBy.MESSAGE)).start();

;

}

}

Can abstract method include throws clause?

Yes. Abstract methods can throw exceptions.

Can an abstract class have a final concrete method?

Yes. An abstract class can have the final concrete method. See the below example.

**public** **class** **AbstractClassFinalConcreteMethod** {

**public** **static** **void** **main**(String[] args) {

Parent p = **new** Child();

p.parentMethod();

}

}

**abstract** **class** **Parent** {

**final** **void** **parentMethod**() {

System.out.println("Final Method in a abstract class");

}

}

**class** **Child** **extends** Parent {

}

Output:

Final Method in a abstract class

In the above example, the parent abstract class has a final concrete method 'parentMethod' which is allowed and also called directly from the parent reference.

What is Double Brace initialization in Java?

Double brace initialization creates an anonymous class derived from the specified class and provides an initializer block.

**new** ArrayList<String>() {{

add("ONE");

add("TWO");

}};

The outer brace creates the anonymous inner class while the inner represents the initializer block.