

Abstract Classes in Java

Difficulty Level : Easy Last Updated : 09 Dec, 2021

In C++, if a class has at least one pure virtual function, then the class becomes abstract. Unlike C++, in Java, a separate keyword *abstract* is used to make a class abstract.

Illustration: Abstract class

```
abstract class Shape
{
    int color;

    // An abstract function
    abstract void draw();
}
```

Following are some important observations about abstract classes in Java.

1. An instance of an abstract class can not be created.
2. Constructors are allowed.
3. We can have an abstract class without any abstract method.
4. Abstract classes can not have final methods because when you make a method final you can not override it but the abstract methods are meant for overriding.
5. We are not allowed to create object for any abstract class.
6. We can define static methods in an abstract class

Let us elaborate on these observations and do justify them with help of clean java programs as follows.

Observation 1: In Java, just likely in C++ an instance of an abstract class cannot be created, we can have references to abstract class type though. It is as shown below via clean java program.

Example

```
// Java Program to Illustrate That an Instance of Abstract
```

```
// Class Can not be created

// Class 1
// Abstract class
abstract class Base {
    abstract void fun();
}

// Class 2
class Derived extends Base {
    void fun()
    {
        System.out.println("Derived fun() called");
    }
}

// Class 3
// Main class
class Main {

    // Main driver method
    public static void main(String args[])
    {

        // Uncommenting the following line will cause
        // compiler error as the line tries to create an
        // instance of abstract class. Base b = new Base();

        // We can have references of Base type.
        Base b = new Derived();
        b.fun();
    }
}
```

Output

Derived fun() called

Observation 2: Like C++, an abstract class can contain constructors in Java. And a constructor of abstract class is called when an instance of an inherited class is created. It is as shown in the program below as follows:

Example

```
// Java Program to Illustrate Abstract Class
// Can contain Constructors

// Class 1
// Abstract class
abstract class Base {

    // Constructor of class 1
    Base()
    {
        // Print statement
        System.out.println("Base Constructor Called");
    }

    // Abstract method inside class1
    abstract void fun();
}

// Class 2
class Derived extends Base {

    // Constructor of class2
    Derived()
    {
        System.out.println("Derived Constructor Called");
    }

    // Method of class2
    void fun()
    {
        System.out.println("Derived fun() called");
    }
}

// Class 3
// Main class
class GFG {

    // Main driver method
    public static void main(String args[])
    {
        // Creating object of class 2
        // inside main() method
        Derived d = new Derived();
    }
}
```

Output

Base Constructor Called

Derived Constructor Called

Observation 3: In Java, we can have an abstract class without any abstract method. This allows us to create classes that cannot be instantiated but can only be inherited. It is as shown below as follows with help of clean java program.

Example

```
// Java Program to illustrate Abstract class
// Without any abstract method

// Class 1
// An abstract class without any abstract method
abstract class Base {

    // Demo method
    void fun()
    {
        // Print message if class 1 function is called
        System.out.println(
            "Function of Base class is called");
    }
}

// Class 2
class Derived extends Base {
}

// Class 3
class Main {

    // Main driver method
    public static void main(String args[])
    {
        // Creating object of class 2
        Derived d = new Derived();

        // Calling function defined in class 1 inside main()
        // with object of class 2 inside main() method
        d.fun();
    }
}
```

Output

Function of Base class is called

Observation 4: Abstract classes can also have final methods (methods that cannot be overridden)

Example

```
// Java Program to Illustrate Abstract classes
// Can also have Final Methods

// Class 1
// Abstract class
abstract class Base {

    final void fun()
    {
        System.out.println("Base fun() called");
    }
}

// Class 2
class Derived extends Base {
}

// Class 3
// Main class
class GFG {

    // Main driver method
    public static void main(String args[])
    {

        // Creating object of abstract class
        Base b = new Derived();

        // Calling method on object created above
        // inside main()
        b.fun();
    }
}
```

Output

Base fun() called

Observation 5: For any abstract java class we are not allowed to create an object i.e., for abstract class instantiation is not possible.

Example

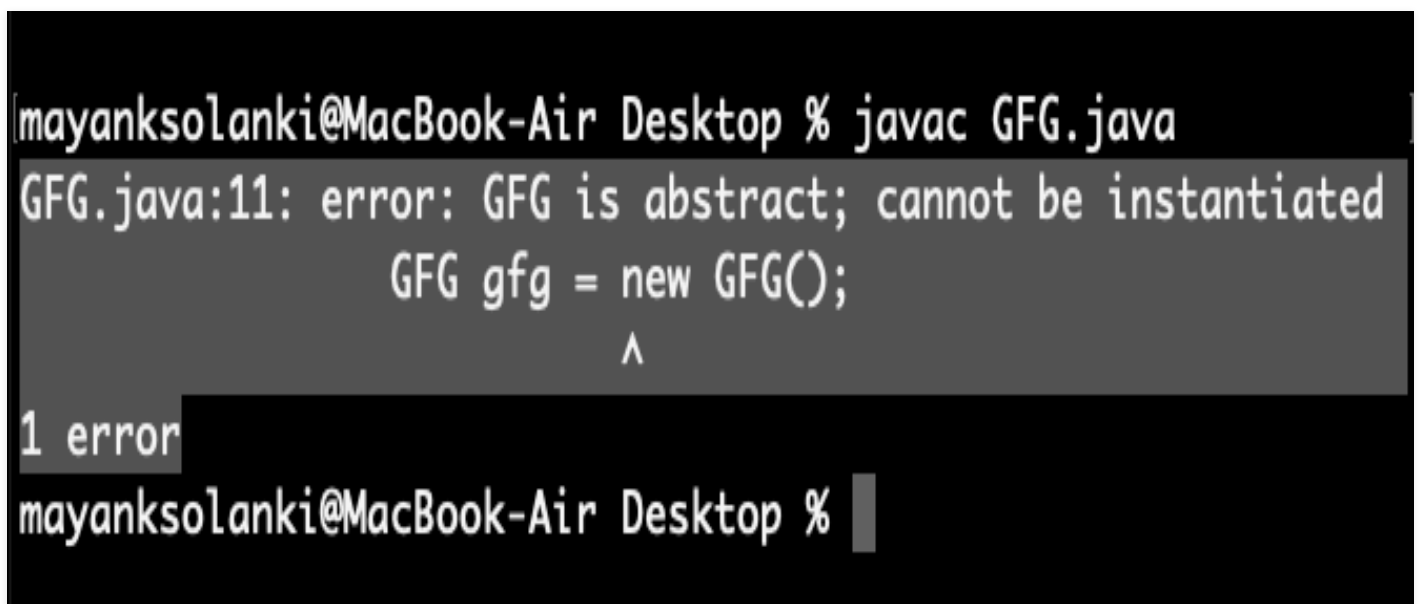
```
// Java Program to Illustrate Abstract Class

// Main class
// An abstract class
abstract class GFG {

    // Main driver method
    public static void main(String args[])
    {

        // Trying to create an object
        GFG gfg = new GFG();
    }
}
```

Output:



The screenshot shows a terminal window with a black background and white text. The prompt is `mayanksolanki@MacBook-Air Desktop %`. The user has entered `javac GFG.java`. The output shows an error: `GFG.java:11: error: GFG is abstract; cannot be instantiated`. Below the error message, the line `GFG gfg = new GFG();` is shown with a caret (^) pointing to the `new` keyword. At the bottom, it says `1 error` and the prompt `mayanksolanki@MacBook-Air Desktop %` is visible again.

```
mayanksolanki@MacBook-Air Desktop % javac GFG.java
GFG.java:11: error: GFG is abstract; cannot be instantiated
        GFG gfg = new GFG();
                    ^
1 error
mayanksolanki@MacBook-Air Desktop %
```

Observation 6: Similar to the interface we can define static methods in an abstract class that can be called independently without an object.

Example

```
// Java Program to Illustrate Static Methods in Abstract
// Class Can be called Independently

// Class 1
// Abstract class
abstract class Helper {

    // Abstract method
    static void demofun()
    {

        // Print statement
        System.out.println("Geeks for Geeks");
    }
}

// Class 2
// Main class extending Helper class
public class GFG extends Helper {

    // Main driver method
    public static void main(String[] args)
    {

        // Calling method inside main()
        // as defined in above class
        Helper.demofun();
    }
}
```

Output

Geeks for Geeks

Must Read:

- [Difference between Abstract class and Interface in Java](#)
- [Difference between Abstract class and Abstract Methods](#)
- [Constructors in Java Abstract Class](#)

- Constructors in Java Abstract Class

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.