Topics

- Abstract Methods
- Abstract Classes

Abstract Methods and Abstract Classes

- Abstract Method → Method with only declaration part and without implementation [Incomplete Method]
- Abstract Classes → If a class has any one abstract method [Incomplete Class]
- Example

```
abstract class Sample
{
        public void doSome1()
        {
                System.out.println("Hello");
        }
        public void doSome2()
        {
                System.out.println("Hi");
        }
        public abstract void doAny();        // Abstract Method
}// End of class Sample
```

Abstract Class

Non-Abstract Methods

Abstract-Methods

Abstract Classes

- An abstract class is a class which has *abstract methods* (i.e. a method with only heading with no body of executable statements)
- An object or instance of abstract classes can not be instantiated
- An abstract class needs to be extended by sub classes to provide the implementation for the abstract methods.
- Abstract classes may contain static methods also. However, abstract and static keyword combination is wrong

abstract static void print(); // wrong

- Abstract classes may extend either another abstract class or concrete (complete or non-abstract) class
- Abstract classes may include constructors, nested classes and interfaces
- Abstract classes has either public, protected, private or package accessibility

Abstract Classes : Syntax

- **Syntax :**

```
<scope>      abstract   class     <class-name>         [extends     <super-class-name>]
    [implements         interface-1, interface-2, …., interface-n]
{
...........................
<scope>      abstract   <return type>          method-name-1(<parameter List>);
<scope>      abstract   <return type>          method-name-2(<parameter List>);
                                 …………………………………..
<scope>      abstract <return type> method-name-n(<parameter List>);
}
```

Note:

1. Abstract class can have one or more abstract methods
2. Abstract classes may extend another class , implements another interface , may have concrete methods

Abstract Classes : Fact I

- A class can be declared as abstract even if it does not have any abstract method
- Example:

```
abstract        class           A
{
        public void doS(int a, int b)
        {
                System.out.println(a+b);
        }// End of Method
}// End of class A
```

Abstract Classes : Fact II

- Only instance methods (object methods) can be declared as abstract.
- ‘static’ and ‘abstract’ forms illegal combination
- Example:

```
abstract        class           A
{
        public static abstract void doS(int a, int b);

}// End of class A
```

Abstract Classes : Fact III

- An abstract class may extend either another abstract class or a concrete (non-abstract) class
- Example

```
abstract class A
{
}// End of class A
abstract class B extends A
{
}// End of class B
```

```
class A
{
}// End of class A
abstract class B extends A
{
}// End of class B
```

Abstract Classes : Fact IV

- An instance or object cannot belong to an abstract class. However a variable can belong to an abstract class type.
- Example

```
abstract class A
{
}// End of class A

class B extends A
{
}// End of class B

class C extends A
{
}// End of class C
```

$$\underbrace{\texttt{A a1 = new A();}}_{\textbf{Compile-Time Error: Object Cannot belong to Abstract Class Type}}$$

$$\underbrace{\texttt{A a1 = new B();}}_{\textbf{Correct: Abstract Class Type Variable Can Point to Any Concrete Sub-class Instance}}$$

$$\underbrace{\texttt{a1 = new C();}}_{\textbf{Correct: Abstract Class Type Variable Can Point to Any Concrete Sub-class Instance}}$$

Abstract Classes : Fact V

- When any class say ‘X’ extends an abstract class say ‘Y’ then in order for class ‘X’ to be a complete or concrete class, the class ‘X’ must implement all the abstract methods of class ‘Y’ otherwise class ‘X’ has to be declared as abstract.

innovate achieve lead

Thank You