'abstract' keyword in Java

The abstract keyword

'abstract' keyword is used to declare the method or a class as abstract.

Abstract Class

A class which contains the **abstract** keyword in its declaration is known as an abstract class.

- Abstract classes may or may not contain *abstract methods*, i.e., methods without a body (public void get();)
- But, if a class has at least one abstract method, then the class **must** be declared abstract.
- If a class is declared abstract, it cannot be instantiated.
- To use an abstract class, you have to inherit it from another class, provide implementations for the abstract methods in it.
- If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

Example

This section provides you an example of the abstract class. To create an abstract class, just use the **abstract** keyword before the class keyword, in the class declaration.

```java
/* File name : Employee.java */
public abstract class Employee {
   private String name;
   private String address;
   private int number;

   public Employee(String name, String address, int number) {
      System.out.println("Constructing an Employee");
      this.name = name;
      this.address = address;
      this.number = number;
   }
   public double computePay() {
      System.out.println("Inside Employee computePay");
      return 0.0;
   }
   public void mailCheck() {
      System.out.println("Mailing a check to " + this.name + " " + this.address);
   }
   public String toString() {
      return name + " " + address + " " + number;
   }
   public String getName() {
      return name;
   }
   public String getAddress() {
      return address;
   }
   public void setAddress(String newAddress) {
      address = newAddress;
   }
   public int getNumber() {
      return number;
   }
}
```

You can observe that except abstract methods the Employee class is the same as the normal class in Java. The class is now abstract, but it still has three fields, seven methods, and one constructor.

Now you can try to instantiate the Employee class in the following way −

```
/* File name : AbstractDemo.java */
public class AbstractDemo {

   public static void main(String [] args) {
      /* Following is not allowed and would raise error */
      Employee e = new Employee("George W.", "Houston, TX", 43);
      System.out.println("\n Call mailCheck using Employee reference--");
      e.mailCheck();
   }
}
```

When you compile the above class, it gives you the following error −

```
Employee.java:46: Employee is abstract; cannot be instantiated
   Employee e = new Employee("George W.", "Houston, TX", 43);
      ^
1 error
```

Abstract Methods

If you want a class to contain a particular method but you want the actual implementation of that method to be determined by child classes, you can declare the method in the parent class as an abstract.

- The **abstract** keyword is used to declare the method as abstract.
- You have to place the **abstract** keyword before the method name in the method declaration.

- An abstract method contains a method signature, but no method body.
- Instead of curly braces, an abstract method will have a semicolon (;) at the end.

Following is an example of the abstract method.

Example

```
public abstract class Employee {
   private String name;
   private String address;
   private int number;

   public abstract double computePay();
   // Remainder of class definition
}
```

Declaring a method as abstract has two consequences −

- The class containing it must be declared as abstract.
- Any class inheriting the current class must either override the abstract method or declare itself as abstract.

Note − Eventually, a descendant class has to implement the abstract method; otherwise, you would have a hierarchy of abstract classes that cannot be instantiated.

Suppose Salary class inherits the Employee class, then it should implement the **computePay()** method as shown below −

```
/* File name : Salary.java */
public class Salary extends Employee {
   private double salary;   // Annual salary

   public double computePay() {
      System.out.println("Computing salary pay for " + getName());
      return salary/52;
   }
   // Remainder of class definition
}
```