Abstraction in Java

Abstraction in Java is another OOPs principle that manages complexity. It is a process of hiding complex internal implementation details from the user and providing only necessary functionality to the users.

In other words, abstraction in Java is a technique by which we can hide the data that is not required to a user.

It hides all unwanted data so that users can work only with the required data. It removes all non-essential things and shows only important things to users.

How to achieve Abstraction in Java?

There are two ways to achieve or implement abstraction in java program. They are as follows:

1. Abstract class
2. Interface

Abstract Class in Java

An **abstract class in Java** is a class, which is declared with an abstract keyword. It is just like a normal class but has two differences.

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

2.It can have zero or more abstract methods which are not allowed in a non-abstract class (concrete class). Classloader class is a good example of an abstract class that does not have any abstract methods.

Java Abstract class makes programming more flexible by providing scopes to write abstract method in subclasses of the abstract class.

KeyPoint

1. Abstract is a non-access modifier in java which is applicable for classes, interfaces, methods, and inner classes.It represents an incomplete class that depends on subclasses for its implementation. Creating subclass is compulsory for abstract class.

2. A non-abstract class is sometimes called a concrete class.

3. An abstract concept is not applicable to variables.

When to use Abstract class in Java?

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

Moreover, the common member of the abstract class can also be shared by the subclasses. Thus, abstract class is useful to make the program more
flexible and understandable.

Abstract Method in Java

A method that is declared with abstract modifier in an abstract class and has no implementation (means no body) is called **abstract method in java**. It does not contain any body.

Abstract method has simply a signature declaration followed by a semicolon.

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

When to use Abstract method in Java?

There are the following uses of abstract method in Java. They are as follows:

1. An abstract method can be used when the same method has to perform different tasks depending on the object calling it.

2. A method can be used as abstract when you need to be overridden in its non-abstract subclasses.

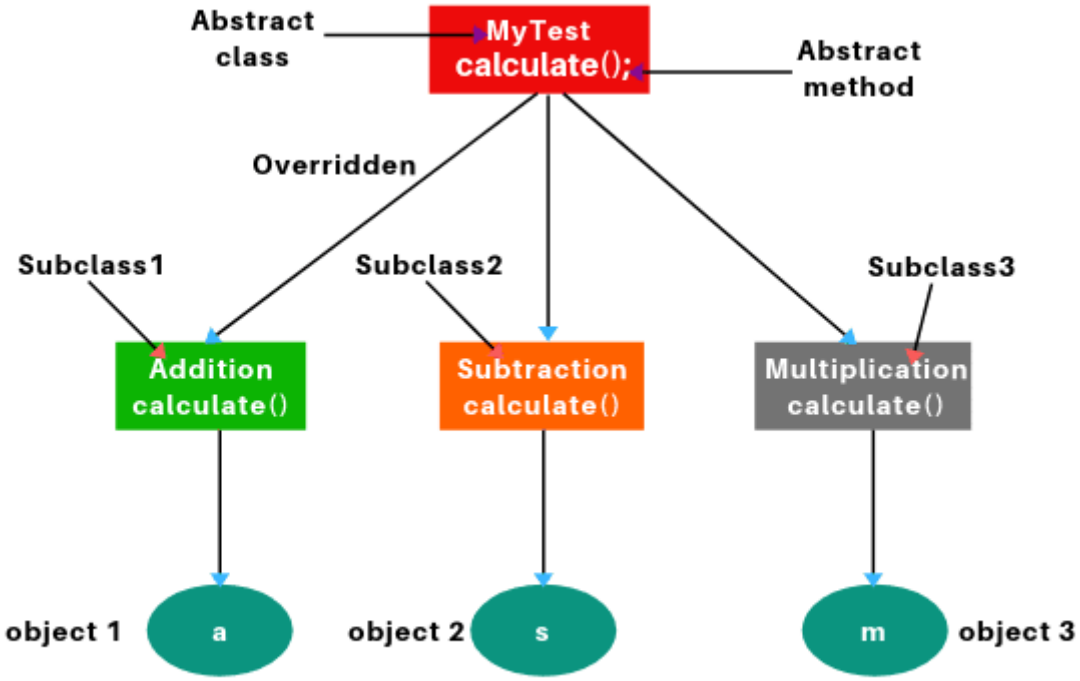

Fig: Abstract class and its subclasses

```
package com.abstraction;
public abstract class MyTest
{
  abstract void calculate(int a, int b); // No body.
 }
public class Addition extends MyTest
{
  void calculate(int a, int b)
  {
    int x = a + b;
    System.out.println(“Sum: ” +x);
   }
}
public class Subtraction extends MyTest
{
  void calculate(int a, int b)
  {
    int y = a - b;
    System.out.println(“Subtract: ” +y);
  }
}
public class Multiplication extends MyTest
{
  void calculate(int a, int b)
  {
    int z = a * b;
    System.out.println(“Multiply: ” +z);
  }
}
public class MyClass
```

```
{
 public static void main(String[] args)
 {
   Addition a = new Addition();
   Subtraction s = new Subtraction();
   Multiplication m = new Multiplication();

    a.calculate(20, 30);
    s.calculate(10, 5);
    m.calculate(10, 20);
  }
}
```

Features of Abstract class in Java

There are following important features of abstract class in Java that should be kept in mind. They are as follows:

1. Abstract class is not a pure abstraction in java.

2.In Java, object creation is not possible for an abstract class because it is a partially implemented class, not fully implemented class.

3. It can be abstract even with no abstract method.

4. It can have one or more abstract methods or non-abstract methods (or concrete methods) or a combination of both methods.

5.Abstract class allows to define private, final, static and concrete methods. Everything is possible to define in an abstract class as per application requirements.

6. It can have constructors.

7. Abstract class does not support multiple inheritance in java but allows in interfaces.

8. It can implement one or more interfaces in java.

Rules of Abstract class in Java

There are the following rules to define an abstract class in Java program. They are as follows:

1. Class must be declared with abstract keyword to make an abstract class.

2.We cannot instantiate an abstract class but we can create object of subclass of the abstract class provided they must implement abstract method.

3. If any method is abstract in a class, the class must be declared as abstract.

4. To use methods declared in an abstract class, the abstract class must be extended by an ordinary class and must implement (override) all abstract methods in that ordinary class.

5. If a new abstract method is added in the abstract class, all non-abstract subclasses which extend
that abstract class, must implement the newly added abstract method. If it does not implement all the abstract methods, the class must be declared as abstract.

6. If a new instance method is added in the abstract class, all non-abstract subclass which extends that abstract class, is not necessary to implement newly added instance method.

7.Inside the abstract class, we can create any number of constructors. Ifyou do not create a constructor, the compiler will create a default constructor.

Rules of Abstract method in Java

The rules of abstract method to define in an abstract class are as follows:

1. Abstract method can only be declared in an abstract class.

2. A non-abstract class cannot have an abstract method, whether it is inherited or declared in Java.

3. It must not provide a method body/implementation in the abstract class for which it is defined.

4. Method name and signature must be the same as in the abstract class.

5. The visibility of the method in the subclass cannot be reduced while overriding abstract method.

6. Abstract method cannot be static or final.

7.It cannot be private because the abstract method must be implemented in the subclass. If we declare it private, we cannot implement it from outside the class.

create an object of abstract class but the compiler will show a compile-time error.

```
package com.abstraction;
public abstract class AbsClass
{
```

```
  // No abstract method here.
}
// Creating a subclass that inherits Abstract class.
public class Subclass extends AbsClass
{
 public static void main(String[] args)
 {
   AbsClass c = new AbsClass(); // Compile-time error.
   AbcClass obj;
  }
}
```

where an abstract class Hello contains both abstract method and instance method. The abstract method “msg2” will be implemented in Test class that extends a class Hello.

```
package com.abstraction;
public abstract class Hello
{
// Declaration of instance method.
  public void msg1()
  {
    System.out.println("msg1-Hello");
  }
  abstract public void msg2();
}
public class Test extends Hello
{
// Overriding abstract method.
  public void msg2()
  {
    System.out.println("msg2-Test");
  }
public static void main(String[] args)
{
// Creating object of subclass Test.
   Test obj = new Test();
    obj.msg1();
    obj.msg2();
  }
}
```

program where an abstract class can have a data member, constructor, abstract, final, static, and instance method (non-abstract method).

```
package com.abstraction;
public abstract class AbstractClass
{
  int x = 10; // Data member.
  AbstractClass()
  {
```

```
    System.out.println("AbstractClass constructor");
  }
final void m1()
{
   System.out.println("Final method");
 }
void m2()
{
   System.out.println("Instance method");
 }
static void m3()
{
   System.out.println("Static method");
 }
  abstract void msg();
 }
public class AbsTest extends AbstractClass
{
  AbsTest()
  {
     System.out.println("AbsTest class constructor");
   }
 void msg()
 {
    System.out.println("Hello Java");
 }
public static void main(String[] args)
{
   AbsTest t = new AbsTest();
    t.msg();
    t.m1();
    t.m2();
    m3();
   System.out.println("x = " +t.x);
  }
}
```

Why abstract class has constructor even though we cannot create object?

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 super 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 a constructor, a class that extends that abstract class will not get compiled.

```
package com.abstraction;
public abstract class Employee
```

```
{
  private String name;
  private int id;
 public Employee(String name, int id)
 {
   this.name = name;
   this.id = id;
  }
// Declaration of concrete method.
  void m1()
  {
    System.out.println("Name: " +name);
    System.out.println("Id: " +id);
  }
}
public class Engineer extends Employee
{
  public Engineer(String name, int id)
  {
    super(name, id); // This statement is used to call super class constructor.
  }
public static void main(String[] args)
{
// Creating an object of the subclass of abstract class.
   Engineer e = new Engineer("Deep", 10202); e.m1();
 }
}
```

program in which an abstract class reference refers to the subclass objects. Abstract class reference can be used to call methods of the subclass.

```
package Abstarctclass;
public abstract class Identity
{
  abstract void getName(String name);
  abstract void getGender(String gender);
  abstract void getCity(String city);
}
public class Person extends Identity
{
  void getName(String name)
  {
    System.out.println("Name : " +name);
  }
void getGender(String gender)
{
  System.out.println("Gender : " +gender);
}
void getCity(String city)
{
  System.out.println("City: " +city);
}

// Newly added method in subclass.
```

```
  void getCountry(String country)
  {
    System.out.println("Country: " +country);
   }
 }
public class Mainclass
{
  public static void main(String[] args)
  {
// Declaring abstract class reference equal to subclass objects.
    Identity i = new Person();
     i.getName("DEEPAK");
     i.getGender("MALE");
     i.getCity("DHANBAD");
     i.getCountry("INDIA"); // Compile-time error because we cannot access newly added
method in subclass using superclass reference.
  }
}
```

Advantage of Abstract class in Java

The main advantages of using abstract class in java application are as follows:

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