

CS261

ASSIGNMENT 6

**NAME:**

ARCHIT AGRAWAL

**ROLL NO. :**

202051213

**SECTION:**

A

# 1. What do you mean by 'abstract' keyword? Write a program in which your class contain more than one abstract method.

*abstract* is a non-access modifier in java applicable for classes, methods but **not** variables. It is used to achieve abstraction which is one of the pillars of Object Oriented Programming (OOP). Abstraction of data is a process of showing essential features without going into its background details.

**Abstract Classes and Abstract Methods:-** Classes with partial implementation. In abstract classes, all the methods present does not contain method definition. All or few methods does not have any body in abstract class. Due to their, partial implementation, objects of abstract class cannot be created. These methods with no body are known as **abstract methods**. When this abstract class is inherited by another class, the abstract methods are defined in the subclass. This is why these methods are also known as **subclasser responsibility** as they have no implementation in the super class. Thus, a subclass must **override** them to provide method definition.

## **Properties of Abstract Classes and Abstract Methods:-**

- Abstract class can have instance methods/variables, static methods/variables, final methods/variables, and constructors.
- Any class that contains abstract methods, must also be declared abstract.
- An abstract class can also contain non-abstract methods.
- A non-abstract class cannot contain abstract methods.
- Objects of abstract classes can not be created.
- Any class that inherits an abstract class must provide a method definition for each abstract method in the super class **or else** it should be declared abstract itself.
- Any abstract method cannot be declared '**final**' as it must be overridden in the subclass and methods declared '**final**' can't be overridden.

- Any abstract method cannot be given '**private**' access because these methods are to be accessed in a subclass.
- Any '**static**' method cannot be abstract. Static methods belong to class and not the object and hence can't be overridden. But, an abstract method must be overridden.

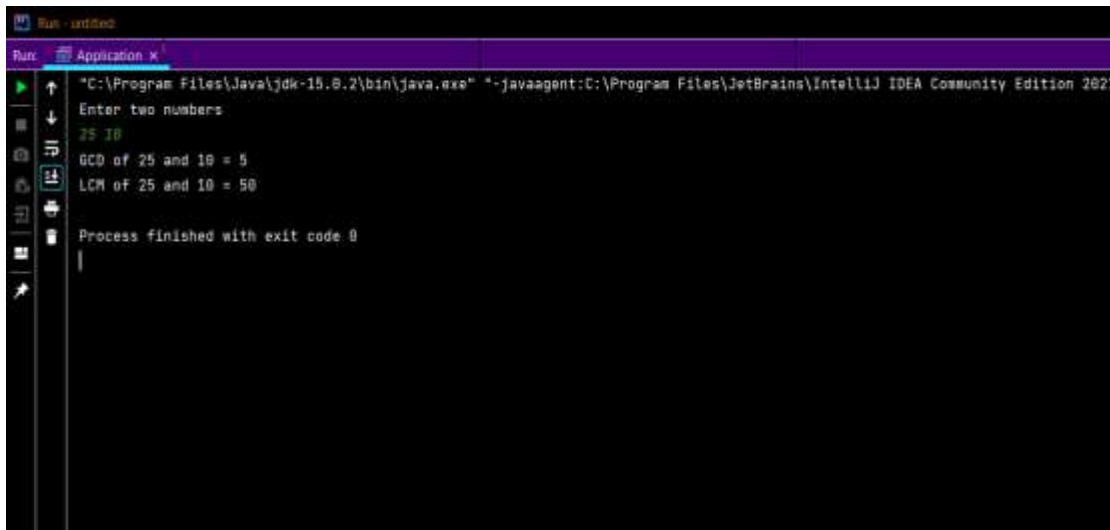
## Code

```
import java.util.*;
abstract class Tasks
{
    //abstract methods cannot have a body
    abstract int lcm(int a, int b);
    abstract int gcd(int a, int b);
}

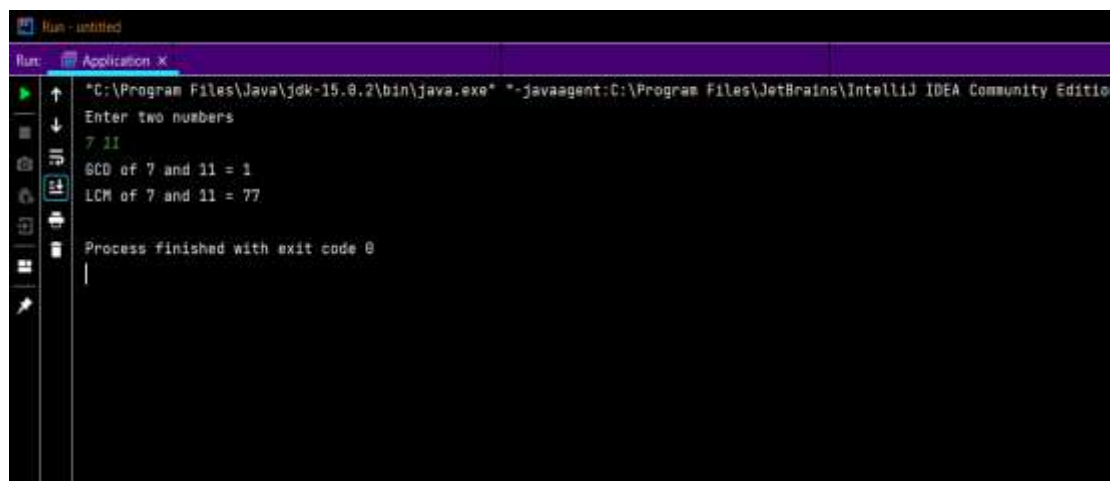
class Calculations extends Tasks
{
    @Override
    //abstract methods must be overridden in subclass
    int gcd(int a, int b)
    {
        if(b == 0)
            return a;
        else
            return gcd(b, a%b);
    }
    @Override
    int lcm(int a, int b)
    {
        return (a * b)/gcd(a, b);
    }
}

public class Application {
    public static void main(String[] args)
    {
        Scanner Sc = new Scanner(System.in);
        //Object of subclass used to call the abstract methods
        Calculations Obj = new Calculations();
        //taking input from user to check functioning of methods
        System.out.println("Enter two numbers");
        int a = Sc.nextInt();
        int b = Sc.nextInt();
        System.out.println("GCD of "+a+" and "+b+" = "+Obj.gcd(a,
b));
        System.out.println("LCM of "+a+" and "+b+" = "+Obj.lcm(a,
b));
    }
}
```

## Output



```
Run - untitled
Run: Application X
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2\lib\idea_rt.jar=62737:C:\Program Files\Java\jdk-15.0.2\bin" -Dfile.encoding=UTF-8
Enter two numbers
25 10
GCD of 25 and 10 = 5
LCM of 25 and 10 = 50
Process finished with exit code 0
```



```
Run - untitled
Run: Application X
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2\lib\idea_rt.jar=62737:C:\Program Files\Java\jdk-15.0.2\bin" -Dfile.encoding=UTF-8
Enter two numbers
7 11
GCD of 7 and 11 = 1
LCM of 7 and 11 = 77
Process finished with exit code 0
```

## 2. What would be the behaviour if this() and super() used in a method?

- this() is used to call one constructor from the other of the same class.
- super() in Java is a reference variable that is used to refer parent class constructors.

Since constructors cannot be called from a method, if we use this() and super() in a method, a **compile time error** will be encountered.

**3. I would like to set data as a = 2 and b = 3, so I came up with the following code. But it won't give the required output. Check and update the code so it would give the output as per the need.**

```
class Account{
    int a;
    int b;

    public void setData(int a, int b) {
        a = a;
        b = b;
    }

    public void showData() {
        System.out.println("Value of A = " + a);
        System.out.println("Value of B = " + b);
    }

    public static void main ( String args[] ) {
        Account obj = new Account ();
        obj.setData(2, 3);
        obj.showData();
    }
}
```

The above code will give the following output:

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if
Value of A = 0
Value of B = 0
```

The reason for this output is described in this paragraph. In the 'setData' method, we have two parameters 'a' and 'b'. When the method is called, these two variables are created in the memory. These variables are local and will not take up space when the method call is executed. In the body of the 'setData' method, the statements 'a = a' and 'b = b' are not setting the instance variables of object of 'Account' class. Instead, these statements are assigning 'a' and 'b' to local variables 'a' and 'b' itself.

To set the value of local variables 'a' and 'b' to the instance variables of object of class 'Account', the following updated code is required.

```
class Account{
    int a;
    int b;

    public void setData(int a, int b) {
        this.a = a;
        this.b = b;
    }

    public void showData() {
        System.out.println("Value of A = " + a);
        System.out.println("Value of B = " + b);
    }

    public static void main ( String args[] ) {
        Account obj = new Account ();
        obj.setData(2, 3);
        obj.showData();
    }
}
```

Now, in the 'setData' method, the statements are 'this.a = a' and 'this.b = b'. Now, 'this.a' and 'this.b' are references to the instance variables of the object on which the method is called. And, the parameters 'a' and 'b' are local variables. So, basically, we are assigning the values stored in 'a' and 'b' (local) to the instance variables of object (using 'this', as it is a reference to the current object). The correct output is given.

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ;
Value of A = 2
Value of B = 3
PS C:\Users\Archit\Desktop\cprog> █
```

#### 4. When do static members of a class get initialized? How can we call non-static methods from static methods in java? Explain with code.

Static members are initialized only once, at the start of execution. These variables will be initialised first, before the initialisation of any instance variables.

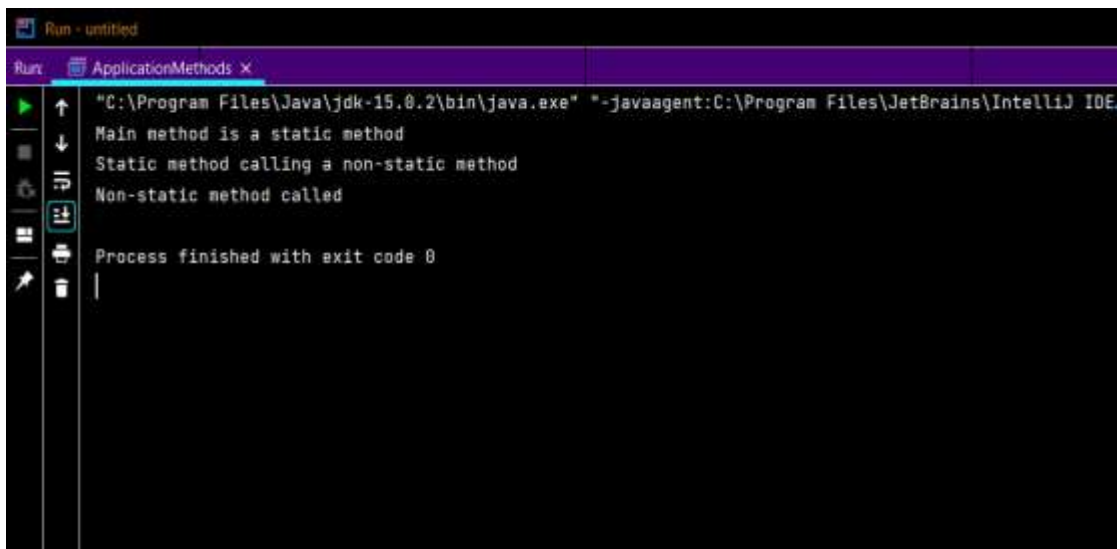
By definition, a non-static method is one that is called on an instance of some class whereas a static method belongs to the class itself.

To call a non-static method from a static method, we need to have an instance of the class containing the non-static method.

### Code

```
public class ApplicationMethods {  
  
    //non-static method  
    public void method2()  
    {  
        System.out.println("Non-static method called");  
    }  
  
    //static method  
    public static void method1()  
    {  
        // here we cannot call method2 like method2() because  
        // because method2 is non-static and  
        // method1 is static  
  
        ApplicationMethods Obj = new ApplicationMethods();  
  
        System.out.println("Static method calling a non-static  
method");  
        Obj.method2();  
    }  
  
    public static void main(String[] args)  
    {  
        System.out.println("Main method is a static method");  
        //calling static method method1 from main  
        method1();  
    }  
}
```

### Output



```
Run - untitled  
Run: ApplicationMethods x  
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE  
Main method is a static method  
Static method calling a non-static method  
Non-static method called  
Process finished with exit code 0
```

## 5. Show the difference between this() and super() with help of a code.

- this() is used to call one constructor from the other of the same class.
- The super() in Java is a reference variable that is used to refer parent class constructors.

this() calls the constructor of the current class while super() calls the constructor of the parent class. The following code makes it more clear.

### Code

```
//Super class
class Parent
{
    double x,y;
    Parent()
    {
        System.out.println("Constructor of Superclass called");
        x = 0;
        y = 0;
    }
    Parent(double x, double y)
    {
        System.out.println("Constructor of Superclass called");
        this.x = x;
        this.y = y;
    }
}

//subclass
class Child extends Parent{

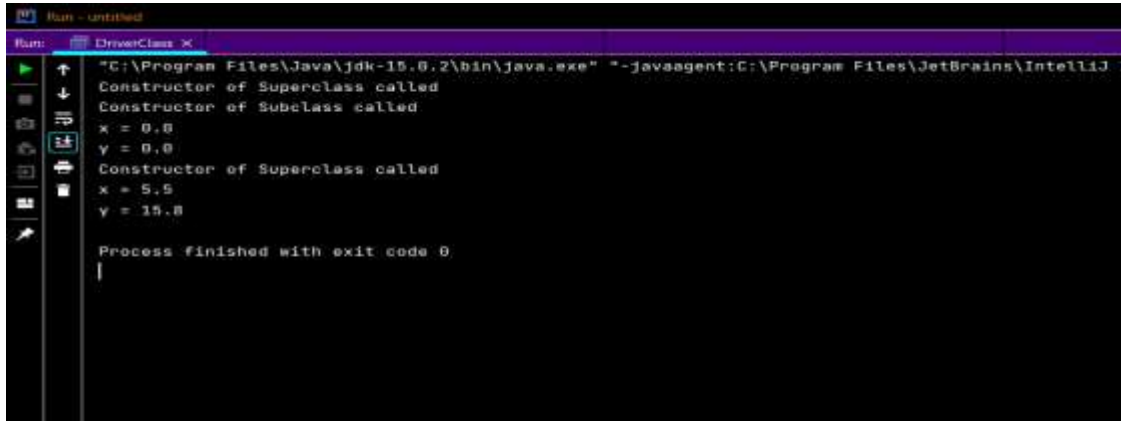
    Child()
    {
        //using this()
        this(0.0, 0.0);
        System.out.println("Constructor of Subclass called");
    }

    Child(double x, double y)
    {
        //using super()
        super(x, y);
    }
    public void display()
    {
        System.out.println("x = " + x);
        System.out.println("y = " + y);
    }
}
```



```
//Driver class to show the difference between this() and super()
public class DriverClass {
    public static void main(String[] args) {
        //obj1 shows usage of this()
        Child obj1 = new Child();
        obj1.display();
        //obj2 shows usage of super()
        Child obj2 = new Child(5.5, 15.8);
        obj2.display();
    }
}
```

## Output



```
Run - untitled
Run: DriverClass x
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Constructor of Superclass called
Constructor of Subclass called
x = 0.0
y = 0.0
Constructor of Superclass called
x = 5.5
y = 15.8
Process finished with exit code 0.
```