Points to remember:

1-An abstract class can have abstract and/or non-abstract methods.

Ans: Yes, abstract class have abstract and / non abstract methods.

Eg code.,

```
1 package tasks.abstraction;
ac
   3 public abstract class AbstractClassExample {
S
          public abstract void studentMarkDetails(); // ABSTRACT METHOD
   5
1-
   6
ж
          public void studentDetails() { // NON-ABSTRACT METHOD
   7⊝
              System.out.println("Student Name : Kumaresan L");
   8
   9
          }
 10 }
31 11
```

2-The abstract may or may not contain the final variables.

Ans: Yes its true, abstract may contain final variables or may not.

Eg code., contain normal variable:-

```
package tasks.abstraction;
 3 public abstract class AbstractClassExample {
           int a = 5; //normal variable
4
 5
       public abstract void studentMarkDetails(); // ABSTRACT METHOD
 6
 7
       public void studentDetails() { // NON-ABSTRACT METHOD
           System.out.println("Student Name : Kumaresan L");
 9
           a = 6; //re-assign
10
           System.out.println(a);
11
       }
12
13 }
14
```

Eg code., contain final variables

```
1 package tasks.abstraction;
 3 public abstract class AbstractClassExample {
       final int a = 5; // final variable
4
       public abstract void studentMarkDetails(); // ABSTRACT METHOD
 6
 7
       public void studentDetails() { // NON-ABSTRACT METHOD
           System.out.println("Student Name : Kumaresan L");
9
           System.out.println(a);
10
       }
11
12 }
13
```

3-An abstract class can have final, static or non-static or non-final variables

Ans: Above all statements are true

* Abstract class have final variable and static variable or may not. Eg code:-

```
package tasks.abstraction;
 3 public abstract class AbstractClassExample {
       final int a = 5; // final variable
       static int b=10; // static variable;
 5
       int c,d;
6
       public abstract void studentMarkDetails(); // ABSTRACT METHOD
 7
 8
       public void studentDetails() { // NON-ABSTRACT METHOD
9⊝
           System.out.println("Student Name : Kumaresan L");
10
           System.out.println(a);
11
           System.out.println(b);
12
       }
13
14 }
15
```

4-An abstract class may provide interface implementation.

Ans: Yes abstract class provide interface implementation but instance of the interface will be not in abstract class, it may be in normal class or in another interfaces.

Eg code.,

interface A:-

```
package tasks.abstraction;

public interface A {
    void stud2();
}
```

A implements in abstract class and also defining stud2():-

```
1 package tasks.abstraction;
 3 public abstract class AbstractClassExample implements A {
       final int a = 5; // final variable
 4
 5
        static int b = 10; // static variable;
 6
       int c, d;
 7
       public abstract void studentMarkDetails(); // ABSTRACT METHOD
 8
 9
10⊝
       public void stud2() {
            System.out.println("Student2 Name : Gowtham");
11
12
13
       public void studentDetails() { // NON-ABSTRACT METHOD
14⊖
            System.out.println("Student Name : Kumaresan L");
15
16
            System.out.println(a);
17
            System.out.println(b);
18
       }
19 }
```

But calling will be in studentMarks class that inherits AbstractClassExample that implements A so the think happened:

```
1 package tasks.abstraction;
  3 public class StudentMarks extends AbstractClassExample {
          public void studentMarkDetails() {
    System.out.println("Total : 394/500 in 10th class");
    System.out.println(AbstractClassExample.b = 11); // inherit static variable and print
               System.out.println(a + 10); // inherit final variable and modify the value
          public void studentAddress() {
    System.out.println("City : Pudukkottai");
10⊖
11
14<del>0</del>
15
          public static void main(String[] args) {
                StudentMarks SM = new StudentMarks();
                SM.studentDetails();
                SM.studentMarkDetails();
               SM.studentAddress();
SM.stud2(); // calling interface here
18
 19
          }
22
23 }
```

5-An abstract class is inherited using the "extends" keyword.

Ans: Yes abstract class inherited using extends keyword in another class.

Eg.,

Abstract class:

```
package tasks.abstraction;
 3 public abstract class AbstractClassExample implements A {
        final int a = 5; // final variable
 4
        static int b = 10; // static variable;
 5
        int c, d;
 6
 7
        public abstract void studentMarkDetails(); // ABSTRACT METHOD
 8
 9
        public void stud2() {
10⊝
            System.out.println("Student2 Name : Gowtham");
11
12
        }
13
        public void studentDetails() { // NON-ABSTRACT METHOD
14⊖
            System.out.println("Student Name : Kumaresan L");
15
            System.out.println(a);
16
            System.out.println(b);
17
        }
18
19 }
```

That is inherited in StudentNames class:-

```
package tasks.abstraction;
 3 public class StudentNames extends AbstractClassExample {
 4
5⊜
       public void studentMarkDetails() {
           System.out.println(a+11);
 6
7
       }
 8
       public static void main(String[] args) {
 9⊜
10
           StudentNames SN = new StudentNames();
11
           SN.studentMarkDetails();
12
       }
13
14
15 }
16
```

6-An abstract class can extend other classes or implement multiple interfaces.

Ans: Yes, absrtact class can extend other classes alone. (without interface) and also implement multiple interfaces.

Eg., Abstract class implement multiple interface.

Interface A -

```
package tasks.abstraction;

public interface A {
   void stud2();
}
```

Interface B -

```
package tasks.abstraction;

public interface B {
    void stud3();
}
```

Implements A and B in abstract class named AbstractClassExample -

```
1 package tasks.abstraction;
 3 public abstract class AbstractClassExample implements A, B {
       final int a = 5; // final variable
       static int b = 10; // static variable;
 5
 6
       int c, d;
 7
       public abstract void studentMarkDetails(); // ABSTRACT METHOD
 8
 9
10⊝
       public void stud2() {
           System.out.println("Student2 Name : Gowtham");
11
12
       }
13
       public void stud3() {
14⊖
           System.out.println("Student3 Name : Dinesh ");
15
16
       }
17
       public void studentDetails() { // NON-ABSTRACT METHOD
18⊖
           System.out.println("Student Name : Kumaresan L");
19
20
           System.out.println(a);
21
           System.out.println(b);
       }
22
23 }
```

But calling will be in studentMarks class that inherits AbstractClassExample that implements A and B so the think happened:-

```
1 package tasks.abstraction;
  3 public class StudentMarks extends AbstractClassExample {
△ 4⊝
        public void studentMarkDetails() {
             System. out.println("Total: 394/500 in 10th class");
System. out.println(AbstractClassExample.b=11); // inherit static variable and print
  6
             System.out.println(a + 10); // inherit final variable and modify the value
  8
  9
        public void studentAddress() {
 10⊝
             System.out.println("City : Pudukkottai");
 11
 12
 13
        public static void main(String[] args) {
14⊖
             StudentMarks SM = new StudentMarks();
             SM.studentDetails();
 16
             SM.studentMarkDetails();
 17
             SM.studentAddress();
 18
             SM.stud2(); // calling interface A here
 19
             SM.stud3(); // calling interface B here
 20
 21
22
        }
 23
 24 }
```

Eg., An abstract class can extend other classes -

Now, here abstract class extends sample class:

sample.java -

```
package tasks.abstraction;

public class Sample {
    public void addition() {
        int a = 5, b = 6;
        System.out.println(a + b);
    }
}
```

Abstract class named AbstractClass2 extends Sample class:

```
package tasks.abstraction;

public abstract class AbstractClass2 extends Sample {
    public void studentPhone() {
        System.out.println("student phone 1 : 9876543210");
    }
}
```

But calling will be in studentPhone class that inherits AbstractClass2 that extends Sample so the think happened:-

```
1 package tasks.abstraction;
 2
   public class studentPhone extends AbstractClass2 {
 3
 4
 5⊜
       public static void main(String[] args) {
            studentPhone sp = new studentPhone();
 6
            sp.addition();
 7
            sp.studentPhone();
 8
       }
 9
10
11 }
12
```

7-An abstract class can have private or protected data members apart from public members.

Ans :- Yes, both the above statements are true. Abstract class can have private or protected members.

Eg.,

```
1 package tasks.abstraction;
 3 public abstract class AbstractClassExample implements A, B {
       final int a = 5; // final variable
       static int b = 10; // static variable;
 6
 7
       int c, d;
       protected int f = 12; // Protected member
 8
       private int e = 110; // private member
 9
10
11
       public abstract void studentMarkDetails(); // ABSTRACT METHOD
12
13⊝
       public void stud2() {
           System.out.println("Student2 Name : Gowtham");
14
15
           System.out.println(e); // private member print
16
17
-18⊖
       public void stud3() {
           System.out.println("Student3 Name : Dinesh ");
19
20
21
       public void studentDetails() { // NON-ABSTRACT METHOD
22⊖
23
            System.out.println("Student Name : Kumaresan L");
24
           System.out.println(a);
25
           System.out.println(b);
       }
26
27 }
```