Write a java program which has a class named balances that has two members name and balance, a constructor which will assign values to name and balance, a method which will show the account name and balance. Create an array of objects of the class balance and call the methods.

```
class balance{
    String name;
    double bal;
    balance(String n, double b){
        name=n;
        bal=b;
    }
    void show() {
        if(bal<0) {
            System.out.println("negative balance");
        }
        else {
            System.out.println(name+" $ " +bal);
        }
    }
}
public class accountbalance {

    public static void main(String[] args) {
        balance current[]= new balance[3];// Here I
am creating an array of object for the class balance
that can held 3 objects.
        current[0]=new balance("Rafsun", 123.33);//
this is the 0th index of the “current” object
        current[1]=new balance("Breanne", 150.55);//
this is the 1st index of the “current” object
        current[2]=new balance("Chris", 180.55);//
this is the 2nd index of the “current” object
        current[0].show(); // we are printing out the
show method of the 0th index of the current object
        //current[1].show();
```

```
        //current[2].show();
        for(int i=0; i<3; i++) {
            current[i].show(); // Here we are
printing out all the values of the object “current”
        }//

    }

}
```

Protected member example:

```
class A18{
    protected String bookname;
    protected int bookid;
    protected double customerbill;
    // Here bookname, bookid, and customerbill all
are protected members since I have used the protected
keywork
}
class B18 extends A18{
    void show() {
        System.out.println("The bookname
is"+bookname);
        System.out.println("bookid is"+bookid);
        System.out.println("Customer bill
is"+customerbill);
//A subclass can access the protected members
directly
    }
}
public class protectdemo {

    public static void main(String[] args) {
        B18 b1= new B18();
```

```
        b1.bookname="Angels and demons";// protected
members can be used by any class inside the same
package. If classes or subclasses from different
package want to access protected members, they won't
be able to do that.
        b1.bookid=1001;
        b1.customerbill=48.00;
        b1.show();// TODO Auto-generated method stub

    }

}
```

Write a Java program that has an interface interfacexample that has a method named lcm which will determine the least common multiple between two numbers. A class named interfaceclass implements the interface "interfacexample" and it has two members, a constructor that assigns the value to the members , and a method which will determine if an integer is a multiple of another integer. Create object of the interfaceclass and call the methods.

A= 12, b=4, so A can be divided by B, that means A is a multiple of B

Least common multiple : 3 and 5

3, 6, 9, 12, 15,18,

5, 10 ,15

Here 15 is the common number , that means the lcm of 3 and 5 is 15

The gcd of 3 and 5 is 1

Lcm=(3*5)/1= 15

The gcd of 10 and 60 is 10

Lcm= (10*60)/10= 60 ,

Code:

```
package ice107onlineclass;
interface interfaceexample{// here interfaceexample
is an interface since I am using the interface
keyword
    int lcm();// lcm is an method of the interface
which does not have any body. You cannot declare the
```

```
body of a method inside an interface. Interface is
used to hide the implementation of a method
}
class interfaceclass implements interfaceexample{
    int a, b;
    interfaceclass(int a, int b){
        this.a=a;
        this.b=b;
    }
    void multiple() {
        if(a%b==0) {
            System.out.println("A is a multiple of
B");
        }
        else {
            System.out.println("A is not a multiple
of B");
        }
    }
    public int lcm() { // Here we are declaring the
body of the lcm method which belongs to the interface
inside a class. Remember to add the public keyword
when you are declaring the method of an interface
inside a class.
        int gcd=1;
        int lcm;
        for(int i=1; i<=a&&i<=b;i++) {
            if(a%i==0&&b%i==0) {
                gcd=i;
            }
        }
        lcm=(a*b)/gcd;
        return lcm;
    }
```

```
}
public class interfaceexample2 {

    public static void main(String[] args) {
        interfaceclass i= new
interfaceclass(60,10);// TODO Auto-generated method
stub
        i.multiple();
        System.out.println(i.lcm());
    }

}
```

Example:

```
interface interfaceexample{
    int lcm();
}
interface b{ // There can be multiple interfaces
inside a package
    int l();
    int a();
}
class interfaceclass implements interfaceexample{//
Here interfaceclass is implementing the interface
“interfaceexample” by using the “implements” keyword
    int a, b;
    interfaceclass(int a, int b){
        this.a=a;
        this.b=b;
    }
    void multiple() {
        if(a%b==0) {
            System.out.println("A is a multiple of
B");
```

```
        }
        else {
            System.out.println("A is not a multiple
of B");
        }
    }
    public int lcm() {
        int gcd=1;
        int lcm;
        for(int i=1; i<=a&&i<=b;i++) {
            if(a%i==0&&b%i==0) {
                gcd=i;
            }
        }
        lcm=(a*b)/gcd;
        return lcm;
    }

}
class interfaceclass2 implements interfaceexample{
// Multiple class can inherit the same interface.
However the body of the methods inside the interface
needs to implemented inside each classes
    int n;
    int a,b;
    interfaceclass2(int n){
        this.n=n;
    }
    void print() {
        System.out.println(n);
    }
    public int lcm() { // We are implementing the lcm
method from the interface
        int c=a+b;
```

```
        return c;
    }
}
class interfaceclass3 implements b{ // Here
“interfaceclass3” is implementing the interface b.
That means you need to declare both the methods of
the interface b inside the class “interfaceclass3”
    int c, d;
    public int l() {
        return c-d;
    }
    public int a() {
        return c*d;
    }

}
public class interfaceexample2 {

    public static void main(String[] args) {
        interfaceclass i= new
interfaceclass(60,10);// TODO Auto-generated method
stub
        i.multiple();
        System.out.println(i.lcm());
    }

}
```

Write a Java Program which has an interface named anagraminterface that has a method named anagram which will determine if two strings are anagram of each other. A class named anagramclass will have two string members, a constructor which will assign the values to the members and it will implement the anagraminterface. Create object of the class and call the methods.

Anagram example :

String1 = “listen”

String2 = “silent”

String1= eilnst

String2= eilnst

These two strings are different words however they have the same characters and the lengths are same. That means they are anagram of each other

Algorithm to find anagram:

1. First we need to determine the lengths of two strings and check if they are the same
2. Sort the two strings
3. If the values of the sorted strings are same, then they are anagram of each other

```
4.package ice107onlineclass;
5.import java.util.*;
6.interface anagraminterface{
7.    boolean anagram();
8.}
9.class anagramclass implements anagraminterface{
10.        String s1, s2;
11.        anagramclass(String s1, String s2){
12.            this.s1=s1;
13.            this.s2=s2;
14.        }
15.        public boolean anagram() {
16.            int n1=s1.length();
17.            int n2=s2.length();
18.            char ch1[]=s1.toCharArray();// Here
  I am converting a string to an array of
  characters by using the tocharArray function
19.            char ch2[]=s2.toCharArray();
20.            if(n1!=n2) {// I am checking the
  length of string s1 and s2
21.                System.out.println("They are not
  anagram");
22.            }
```

```
23.             Arrays.sort(ch1);// Here I am
  sorting the ch1 array
24.             Arrays.sort(ch2);
25.             for(int i=0; i<n1; i++) {
26.                 if(ch1[i]!=ch2[i]) {// Here I am
  comparing the characters from ch1 and ch2 array
27.                     return false; // if the
  characters are not same, then return false
28.                 }
29.             }
30.             return true;// if the characters
  from the arrays ch1 and ch2 are same, then return
  true
31.         }
32.     }
33.
34.     public class ana {
35.
36.         public static void main(String[] args) {
37.             anagramclass a12= new
  anagramclass("listen", "silent");// TODO Auto-
  generated method stub
38.             System.out.println(a12.anagram());
39.         }
40.
41.     }
```