Write a java program that has a class named A16 that has a member n1, and a method name print that will print the following pattern * .

```
* *
* * *
* * * *
* * * * *
```

A subclass of A16 named B16 has a constructor, also has a method named print which will print out all the divisors of n1 started from 1. Create object of B1 and call the methods.

If both the superclass and subclass has the same method (same method name, same amount of parameter, same return type) , then subclass method will always override superclass method. This is called method overriding. If you want to call the superclass method , then you need to use the super keyword= super.superclass method

```
class A16{
    int n;
    void print() {
        int i;
        for(i=1; i<=5; i++) {
            for(j=1; j<=i; j++)
            System.out.print("*");
            System.out.println();
        }
    }
}
class B16 extends A16{
    B16(int n){
        this.n=n;
    }
    void print() {
        super.print();// by using super, I am calling the superclass print method. Since supeclass and sublclass have the same method and subclass method will override superclass method, we have to use super to call the superclass method.
        int i;
```

```
        for(i=1; i<=n; i++) {
            if(n%i==0) {
                System.out.println(i);
            }
        }
    }
}
public class methodoverriding {

    public static void main(String[] args) {
        B16 B1= new B16(20);
        B1.print();// TODO Auto-generated method stub

    }

}
```

Another way we can use to call superclass method in case of method overriding:

```
class A16{
    int n;
    void print() {
        int i;
        for(i=1; i<=5; i++) {
            for(j=1; j<=i; j++)
            System.out.print("*");
            System.out.println();
        }

    }
}
class B16 extends A16{
    B16(int n){
        this.n=n;
    }
```

```
    void print() {
        int i;
        for(i=1; i<=n; i++) {
            if(n%i==0) {
                System.out.println(i);
            }
        }
    }
}
public class methodoverriding {

    public static void main(String[] args) {
        A16 A1 = new A16();
        B16 B1= new B16(20);
        A16 A2;
        // Here I am creating a reference to the
superclass A16
        A2=A1; // Here A2 is equal to A1 which is the
object of the superclass A16
        A2.print();// it will call the print method
of superclass A16
        A2= B1; // Here A2 is equal to B1 which is
the object of subclass B16
        A2.print(); // It will call the print method
of subclass B16
    }

}
```

If multiple superclasses and subclasses has the same method, then we can distinguish these methods by creating a reference of the superclass inside main class. Then we can set the value of the object of a

particular superclass/subclass to that reference. This is called dynamic method dispatch.

Write a java program that has an abstract class employee that has two public members, employee name and employeeid and two private members, hour and rate. Employee has an abstract method named wage that will calculate the regular and overtime wage of an employee (overtime wage is 1.5 times more than the regular wage and hours above 40 will be counted as overtime). A subclass of employee will have a constructor that will set the values to the members of superclass and have a method that will print the employeename and employeeid. Create objects of the subclass and call the methods.

```
abstract class employee17{// Here employee17 is an abstract class since I have used the abstract keyword before the classname.
    String employeename;
    int employeeid;
    private double hours;
    private double rate;
    employee17(double hours, double rate){
        this.hours=hours;
        this.rate=rate;
    }
    double gethours() {
        return hours;
    }
    double getrate() {
```

```
        return rate;
    }
    abstract double wage();// here wage is an
abstract method. An abstract method inside the
superclass should not have any body. The abstract
method should be implemented inside a subclass
}
class w extends employee17{
    double hours=gethours();
    double rate=getrate();
    w(String employeename, int employeeid, double
hours , double rate){
        super(hours,rate);
        this.employeename=employeename;
        this.employeeid=employeeid;
    }
    void print() {
        System.out.println(employeename);
        System.out.println(employeeid);
    }
    double wage() {// here I have implemented the
abstract method wage inside the subclass w. Remember
when you implement it, you should not use the
abstract keyword. Only the method returntype and the
methodname should be used for the implementation of
an abstract method
        double overtime;
        if(hours>40) {
            overtime=hours-40;
            hours= hours-overtime;
            return hours*rate+overtime*1.5*rate;

        }
        else {
```

```
            return hours*rate;
        }
    }
}

public class abstractexample {

    public static void main(String[] args) {
        w w1= new w("Rafsun", 120, 45, 10.5);// TODO
Auto-generated method stub
        w1.print();
        System.out.println(w1.wage());
    }

}
```

Write a java program that has an abstract class named A which has a member named n1 and an abstract method that will print out all the divisors of n1 started from 1. A subclass of A will have constructor to assign the value to the member of superclass and have a method which will print the value of n1. Create object of the subclass and call the method.