```
Assignment: Implement Unification algorithm
PRAJAKTA DEOKULE
Batch:A1
Roll no:3330

import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;
public class unification {

	public static HashMap<String,String> Substitution = new HashMap<String,String>();
	char Predicate1;
	char Predicate2;

	Scanner in = new Scanner(System.in);

	String[] arguments1 ; // array to store arguments of predicate
	String[] arguments2 ;

	public void Input() //Taking predicates and arguments as input
	{
	    System.out.println("Enter Predicate 1 ");
	    Predicate1 = in.next().charAt(0);

	    System.out.println("Enter number of arguments for Predicate "+Predicate1+" ");
	    int Na1 = in.nextInt();

	    arguments1 = new String[Na1];

	    for(int i = 0;i<Na1;i++){
	       System.out.println("Enter argument "+(i+1));
	        arguments1[i] = in.next();
	     }

	     System.out.println("Enter Predicate 2 ");
	     Predicate2 = in.next().charAt(0);

	     System.out.println("Enter number of arguments for Predicate "+Predicate2+" ");

	     int Na2 = in.nextInt();

	     arguments2 = new String[Na2];

	     for(int i = 0;i<Na2;i++){
	        System.out.println("Enter argument "+(i+1));
	        arguments2[i] = in.next();
	     }
	}
	public void set(String s1, String s2) //setting new values in case of predicate within
predicate
	{
	     this.Predicate1 = s1.charAt(0);
	     this.Predicate2 = s2.charAt(0);

	     this.arguments1 = s1.split(",");
	      this.arguments2 = s2.split(",");

	      //remove predicate symbol and opening bracket
	      arguments1[0] = arguments1[0].substring(2);
	      arguments2[0] = arguments2[0].substring(2);

	    //remove predicate closing bracket
	      int n=arguments1.length-1;
	      int m=arguments2.length-1;
```

```
            arguments1[n] = arguments1[n].substring(0,arguments1[n].length()-1);

            arguments2[m] = arguments2[m].substring(0,arguments2[m].length()-1);
        }
        public boolean unify() //unification algorithm
        {

            //If the initial Predicate symbol are not same then return false

            if(Predicate1!=Predicate2){
              System.out.println("Predicates not same . Unification not possible");
              return false;
             }

            //If the number of arguments are not same, then return false
           if(arguments1.length!=arguments2.length){
               System.out.println("Arguments not same . Unification not possible");
               return false;
             }

            for(int i = 0;i<arguments1.length;i++)
           {
               // check if the arguments are same

            if(!arguments1[i].equals(arguments2[i])){

              // If the argument are variable or constant
                   if(arguments1[i].indexOf('(')==-1||arguments2[i].indexOf('(')==-1)
                   {
                     //if argument in predicate 1 is a variable
                      if(arguments1[i].indexOf('(')==-1)
                       {
                              if(arguments2[i].indexOf(arguments1[i])!=-1)
                               {
                                 System.out.println("Unification not possible");
                                   return false;
                               }

                             // if no false then add the arguments in substitution
                             Substitution.put(arguments2[i],arguments1[i]);
                       }// close if(arguments1[i].indexOf('(')==-1)


                      //if argument in predicate 2 is a variable
                      else if(arguments2[i].indexOf('(')==-1)
                     {
                           // if 1 is present in other then return false
                             if(arguments1[i].indexOf(arguments2[i])!=-1)
                                   return false;

                           // if no false then add the arguments in substitution
                             Substitution.put(arguments1[i],arguments2[i]);

                        }//close else if(arguments2[i].indexOf('(')==-1)

                   }

                   else
                   {
                       //Call Unify function with the ith element of argument in predicate 1
and ith element of argument in predicate 2,
                           //and put the result into substitution

                            Algorithm A2 = new Algorithm();
                          A2.set(arguments1[i], arguments2[i]);
```

```
                    //If unification not possible then return false
                      if(A2.unify()==false)
                        return false;

                   }//close else
                        // perform substitution
                        modify();
              }
          }//close main for loop

            return true;
        }//close unify()


        public void modify() // perform substitution
        {

              Iterator<String> itKey = Substitution.keySet().iterator();
            Iterator<String> itValue = Substitution.values().iterator();

            while(itKey.hasNext()){

                String a = itKey.next();
                String b = itValue.next();

                 for(int i = 0;i<arguments1.length;i++)
                   arguments1[i]=arguments1[i].replace(b,a);

                 for(int i = 0;i<arguments2.length;i++)
                   arguments2[i] = arguments2[i].replace(b,a);

              }//close while loop
        }//close modify()

        public void display() //display the substitution list
        {

           System.out.println("Substitution List -");
           System.out.print("{ ");
              Iterator<String> itKey = Substitution.keySet().iterator();
              Iterator<String> itValue = Substitution.values().iterator();

              while(itKey.hasNext())
                   System.out.print(itKey.next()+"|"+itValue.next()+", " );

           System.out.print("} ");

         }//close display()

}//close class

import java.util.Scanner;
public class UnificationAlgo {
        public static void main(String args[]){

              Scanner sc=new Scanner(System.in);

              System.out.println("Unification Algorithm:");
              String ch;

              do {

                    unification obj=new unification();
                    obj.takeInput();
```

```
                    if(obj.unify()) {

                          obj.display();
                    }
                    System.out.print("Do you wish to continue?(y/n):");
                    ch=sc.next();


              }while(ch.equals("y"));
              sc.close();
       }
}
```

```
OUTPUT
Unification Algorithm
Enter Predicate 1
p
Enter number of arguments for Predicate p
2
Enter argument 1
x
Enter argument 2
y
Enter Predicate 2
p
Enter number of arguments for Predicate p
2
Enter argument 1
a
Enter argument 2
f(X)
Substitution List -
{ a|x, f(X)|y, }

Do you want to continue (y/n):
y
Enter Predicate 1
p
Enter number of arguments for Predicate p
2
Enter argument 1
x
Enter argument 2
y
Enter Predicate 2
q
Enter number of arguments for Predicate q
2
Enter argument 1
a
Enter argument 2
b
Predicates not same . Unification not possible


Do you want to continue (y/n):
n
```