

# Assignment 3

DISCRETE STRUCTURES

HASSAM UD DIN | P190029 | BCS – 3A

## File - 01:

### userInput.java:

```
package Assignment3;

public class userInput {
    private int x;
    private int y;

    userInput(int x,int y)
    {
        this.x=x;
        this.y=y;
    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    @Override
```

```

public String toString()
{
    String str= "(" + this.x + "," + this.y + ")";
    return str;
}
}

```

## File – 02:

### Assignment3.java:

```

package Assignment3;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Random;
import java.util.Set;

public class Assignment3 {

    public static boolean checkValid(ArrayList<Integer> A, ArrayList<userInput> R)
    {
        if (A.isEmpty())
        {
            if (R.isEmpty())
            {
                return true;
            }
        }
        else
        {
            System.out.println("The input relation R is not valid for the set A");
            return false;
        }
    }
}

```

```

    }
}
for (int i=0;i<R.size();i++)
{
    if (!A.contains(R.get(i).getX()) || !A.contains(R.get(i).getY()))
    {
        System.out.println("The input relation R is not valid for the set A");
        return false;
    }
}
return true;
}

```

```

public static boolean checkReflexive(ArrayList<Integer> A, ArrayList<userInput> R,boolean
printVar)

```

```

{
    if (R.isEmpty() )
    {
        if (A.isEmpty())
        {
            return true;
        }
        return false;
    }
    int reqCount=A.size();
    int currCount=0;
    boolean foundFlag=false;
    for (int j=0;j<A.size();j++)
    {
        for (int i=0;i<R.size();i++)

```

```

{
    if (A.get(j)==R.get(i).getX() && A.get(j)==R.get(i).getY())
    {
        currCount++;
        foundFlag=true;
        break;
    }
}
if (foundFlag==false)
{
    if (printVar)
    {
        System.out.print("The following tuple was not found: " + "(" + A.get(j) + "," + A.get(j)
+ ")" );
        return false;
    }
}
else
{
    foundFlag=false;
}
}
if (currCount==reqCount)
{
    return true;
}
else
{
    return false;
}

```

```
}
```

```
public static boolean checkSymmetric(ArrayList<Integer> A,ArrayList<userInput> R,boolean  
printVar)
```

```
{  
    if (R.isEmpty() )  
    {  
        return true;  
    }  
    int[] flags= new int[R.size()];  
    for (int i=0;i<R.size();i++)  
    {  
        flags[i]=0;  
    }  
    for (int i=0;i<R.size();i++)  
    {  
        for (int j=0;j<R.size();j++)  
        {  
            if (R.get(i).getX()==R.get(j).getY() && R.get(i).getY()==R.get(j).getX())  
            {  
                flags[i]=1;  
                break;  
            }  
        }  
    }  
    for (int i=0;i<flags.length;i++)  
    {  
        if (flags[i]==0)  
        {  
            if (printVar)
```

```

        {
            System.out.print("Following tuple was not found: " + "(" +
R.get(i).getY()+" "+R.get(i).getX() + ")");
            return false;
        }

```

```

    }
}
return true;
}

```

```

public static boolean checkAntiSymmetric(ArrayList<Integer> A, ArrayList<userInput> R)
{
    boolean ret=false;
    if (R.size()>=A.size())
    {
        if (checkReflexive(A,R,false))
        {
            if (R.size()==A.size())
            {
                ret=true;
            }
            else if (checkSymmetric(A,R,false))
            {
                Random rand= new Random();
                int x= rand.nextInt(R.size());
                while (R.get(x).getX()==R.get(x).getY())
                {
                    x= rand.nextInt(R.size());
                }
            }
        }
    }
}

```

```

        System.out.print( "(" + R.get(x).getX() + "," + R.get(x).getY() + ")" + "and " + "(" +
R.get(x).getY() + "," + R.get(x).getX() + ")" + " exist but " + R.get(x).getX() + "!=" +
R.get(x).getY() + "\t");

```

```

        ret=false;

```

```

    }

```

```

    else

```

```

    {

```

```

        ret= true;

```

```

    }

```

```

}

```

```

else

```

```

{

```

```

    if (!checkSymmetric(A,R,false))

```

```

    {

```

```

        ret= true;

```

```

    }

```

```

    else

```

```

    {

```

```

        Random rand= new Random();

```

```

        int x= rand.nextInt(R.size());

```

```

        while (R.get(x).getX()!=R.get(x).getY())

```

```

        {

```

```

            x= rand.nextInt(R.size());

```

```

        }

```

```

        System.out.print( "(" + R.get(x).getX() + "," + R.get(x).getY() + ")" + "and " + "(" +
R.get(x).getY() + "," + R.get(x).getX() + ")" + "exist but" + R.get(x).getX() + "!=" +
R.get(x).getY());

```

```

        ret=false;

```

```

    }

```

```

}

```

```

}

```



```

else
{
    if (!checkReflexive(A,R,false))
    {
        if (!checkSymmetric(A,R,false))
        {
            ret= true;
        }
        else
        {
            ret=true;
            for (int i=0;i<R.size();i++)
            {
                if (R.get(i).getX()!=R.get(i).getY())
                {
                    ret=false;
                    break;
                }
            }
        }
    }
}
return ret;
}

```

```

public static void main(String[] args) {
    ArrayList<Integer> A= new ArrayList<Integer>();
    ArrayList<userInput> R= new ArrayList<userInput>();
    A.add(1);
    A.add(2);
}

```

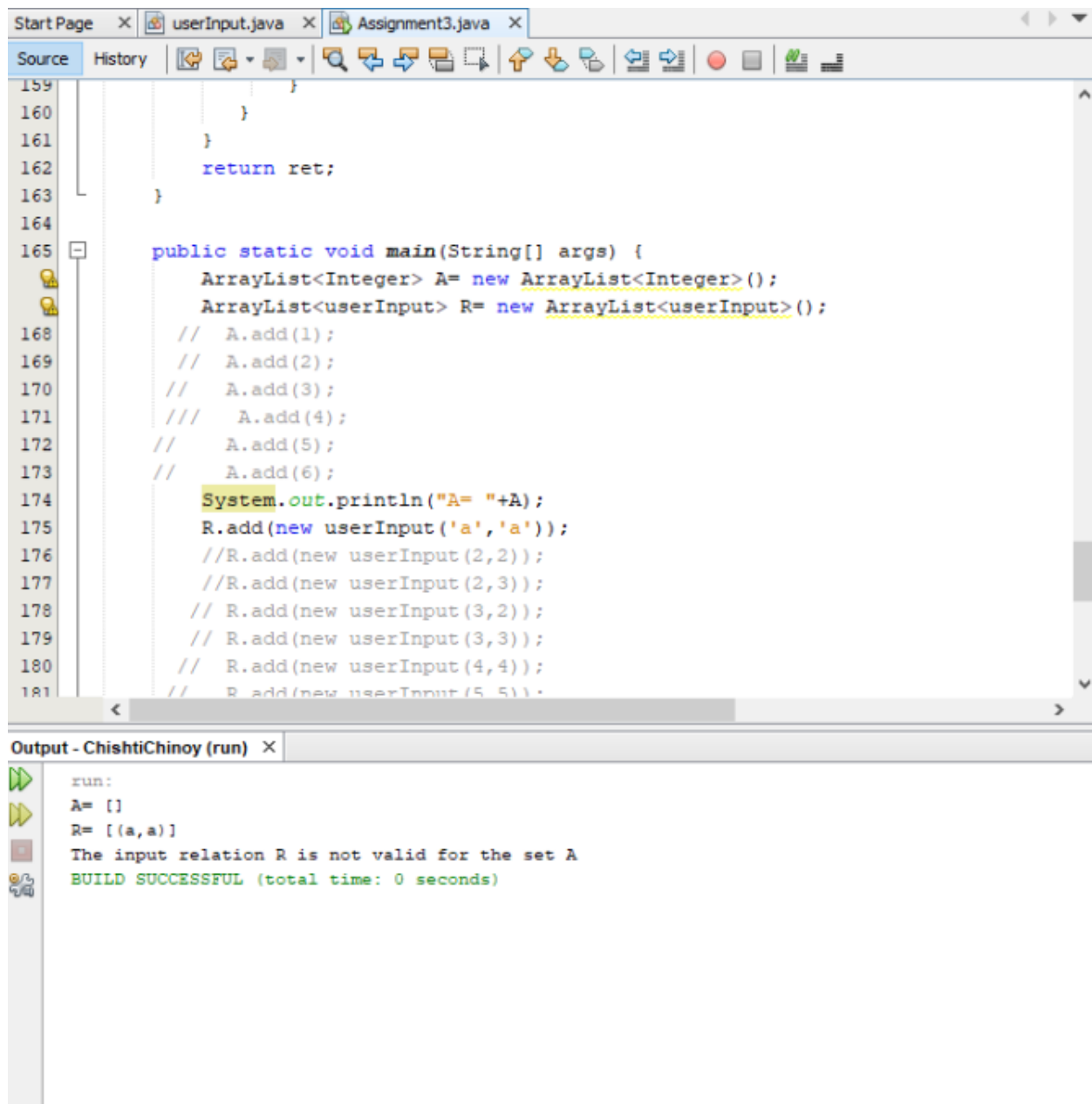
```

A.add(3);
A.add(4);
A.add(5);
A.add(6);
System.out.println("A= "+A);
R.add(new userInput(1,1));
R.add(new userInput(2,2));
//R.add(new userInput(2,3));
//R.add(new userInput(3,2));
// R.add(new userInput(3,3));
// R.add(new userInput(4,4));
// R.add(new userInput(5,5));
// R.add(new userInput(6,6));
System.out.println("R= "+R);
boolean valid= checkValid(A,R);
if (valid)
{
    System.out.println("Reflexive: " + checkReflexive(A,R,true));
    System.out.println("Symmetric: " + checkSymmetric(A,R,true));
    System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
    System.out.println();
}
}
}

```

## OUTPUTS:

### Test case 1:



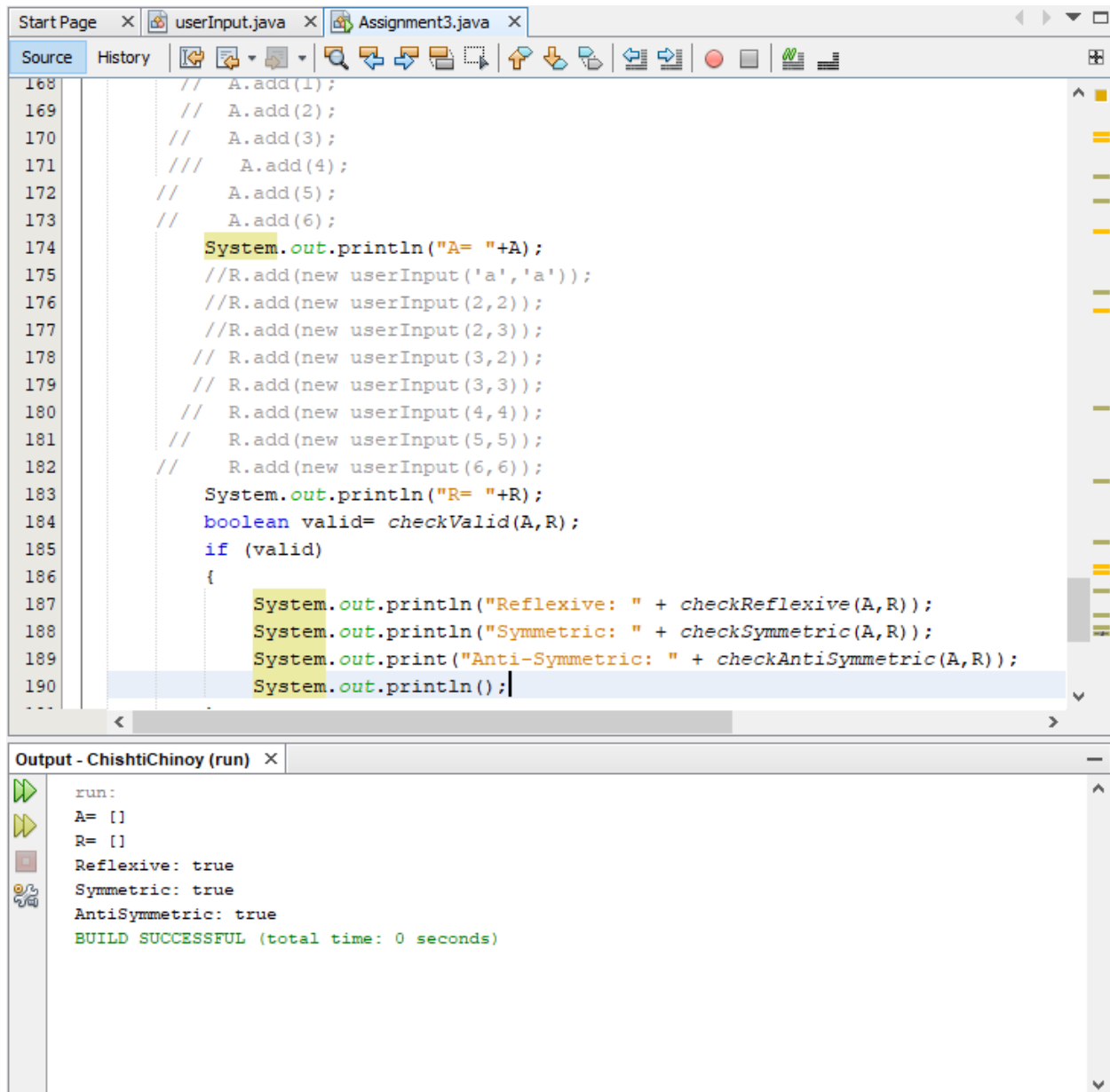
The screenshot displays an IDE with two tabs: 'userInput.java' and 'Assignment3.java'. The 'Assignment3.java' tab is active, showing a Java program. The code defines a `main` method that initializes an `ArrayList<Integer>` named `A` and an `ArrayList<userInput>` named `R`. It then adds several elements to `A` and `R`. The output window at the bottom shows the execution results.

```
159     }
160     }
161     }
162     return ret;
163 }
164
165 public static void main(String[] args) {
166     ArrayList<Integer> A= new ArrayList<Integer>();
167     ArrayList<userInput> R= new ArrayList<userInput>();
168     // A.add(1);
169     // A.add(2);
170     // A.add(3);
171     /// A.add(4);
172     // A.add(5);
173     // A.add(6);
174     System.out.println("A= "+A);
175     R.add(new userInput('a','a'));
176     //R.add(new userInput(2,2));
177     //R.add(new userInput(2,3));
178     // R.add(new userInput(3,2));
179     // R.add(new userInput(3,3));
180     // R.add(new userInput(4,4));
181     // R.add(new userInput(5,5));
```

Output - ChishtiChinoy (run) X

```
run:
A= []
R= [(a,a)]
The input relation R is not valid for the set A
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Test case 2:



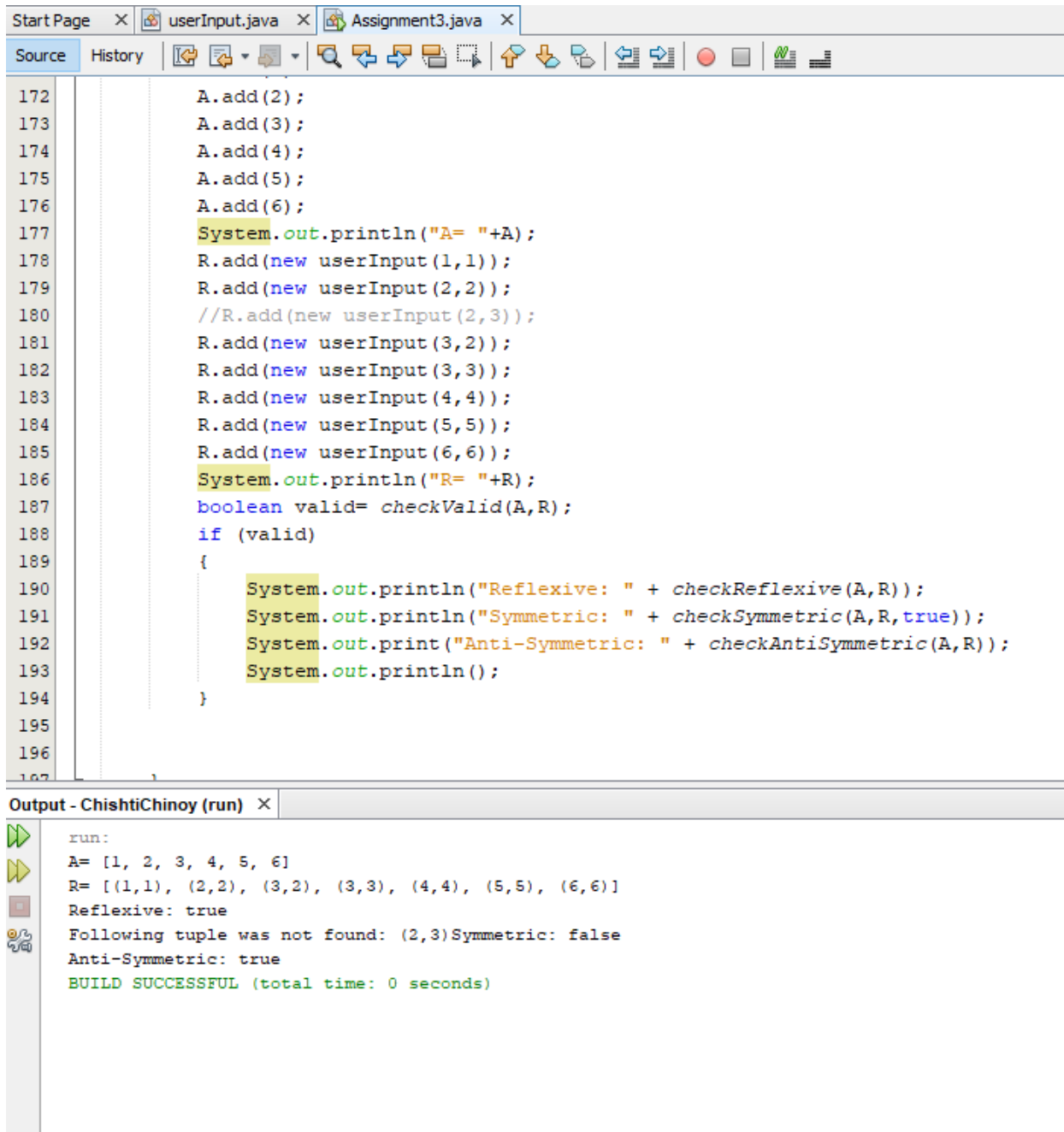
The screenshot shows an IDE with two tabs: 'userInput.java' and 'Assignment3.java'. The 'Assignment3.java' tab is active, displaying Java code from line 168 to 190. The code defines a set A and a relation R, then checks for Reflexive, Symmetric, and Anti-Symmetric properties. The output window at the bottom shows the results of these checks, all of which are 'true'. The build was successful.

```
168      // A.add(1);
169      // A.add(2);
170      // A.add(3);
171      /// A.add(4);
172      // A.add(5);
173      // A.add(6);
174      System.out.println("A= "+A);
175      //R.add(new userInput('a','a'));
176      //R.add(new userInput(2,2));
177      //R.add(new userInput(2,3));
178      // R.add(new userInput(3,2));
179      // R.add(new userInput(3,3));
180      // R.add(new userInput(4,4));
181      // R.add(new userInput(5,5));
182      // R.add(new userInput(6,6));
183      System.out.println("R= "+R);
184      boolean valid= checkValid(A,R);
185      if (valid)
186      {
187          System.out.println("Reflexive: " + checkReflexive(A,R));
188          System.out.println("Symmetric: " + checkSymmetric(A,R));
189          System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
190          System.out.println();
```

Output - ChishtiChinoy (run) ×

```
run:
A= []
R= []
Reflexive: true
Symmetric: true
AntiSymmetric: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

### Test case 3:



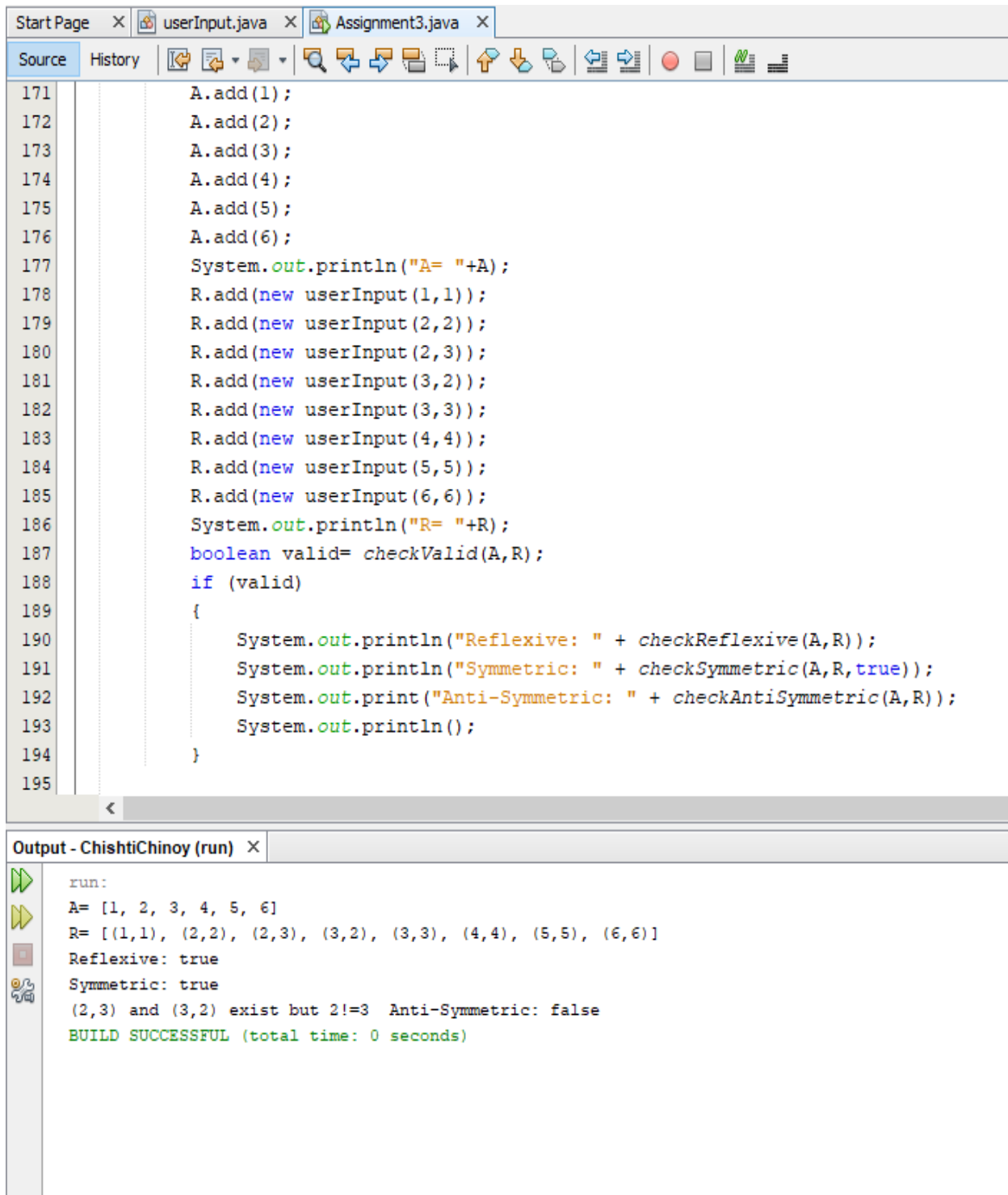
The screenshot shows an IDE with two tabs: 'userInput.java' and 'Assignment3.java'. The 'Assignment3.java' tab is active, displaying the following code:

```
172     A.add(2);
173     A.add(3);
174     A.add(4);
175     A.add(5);
176     A.add(6);
177     System.out.println("A= "+A);
178     R.add(new userInput(1,1));
179     R.add(new userInput(2,2));
180     //R.add(new userInput(2,3));
181     R.add(new userInput(3,2));
182     R.add(new userInput(3,3));
183     R.add(new userInput(4,4));
184     R.add(new userInput(5,5));
185     R.add(new userInput(6,6));
186     System.out.println("R= "+R);
187     boolean valid= checkValid(A,R);
188     if (valid)
189     {
190         System.out.println("Reflexive: " + checkReflexive(A,R));
191         System.out.println("Symmetric: " + checkSymmetric(A,R,true));
192         System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
193         System.out.println();
194     }
195
196
197
```

Below the code editor is the 'Output - ChishtiChinoy (run)' window, which displays the following output:

```
run:
A= [1, 2, 3, 4, 5, 6]
R= [(1,1), (2,2), (3,2), (3,3), (4,4), (5,5), (6,6)]
Reflexive: true
Following tuple was not found: (2,3)Symmetric: false
Anti-Symmetric: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Test case 4:



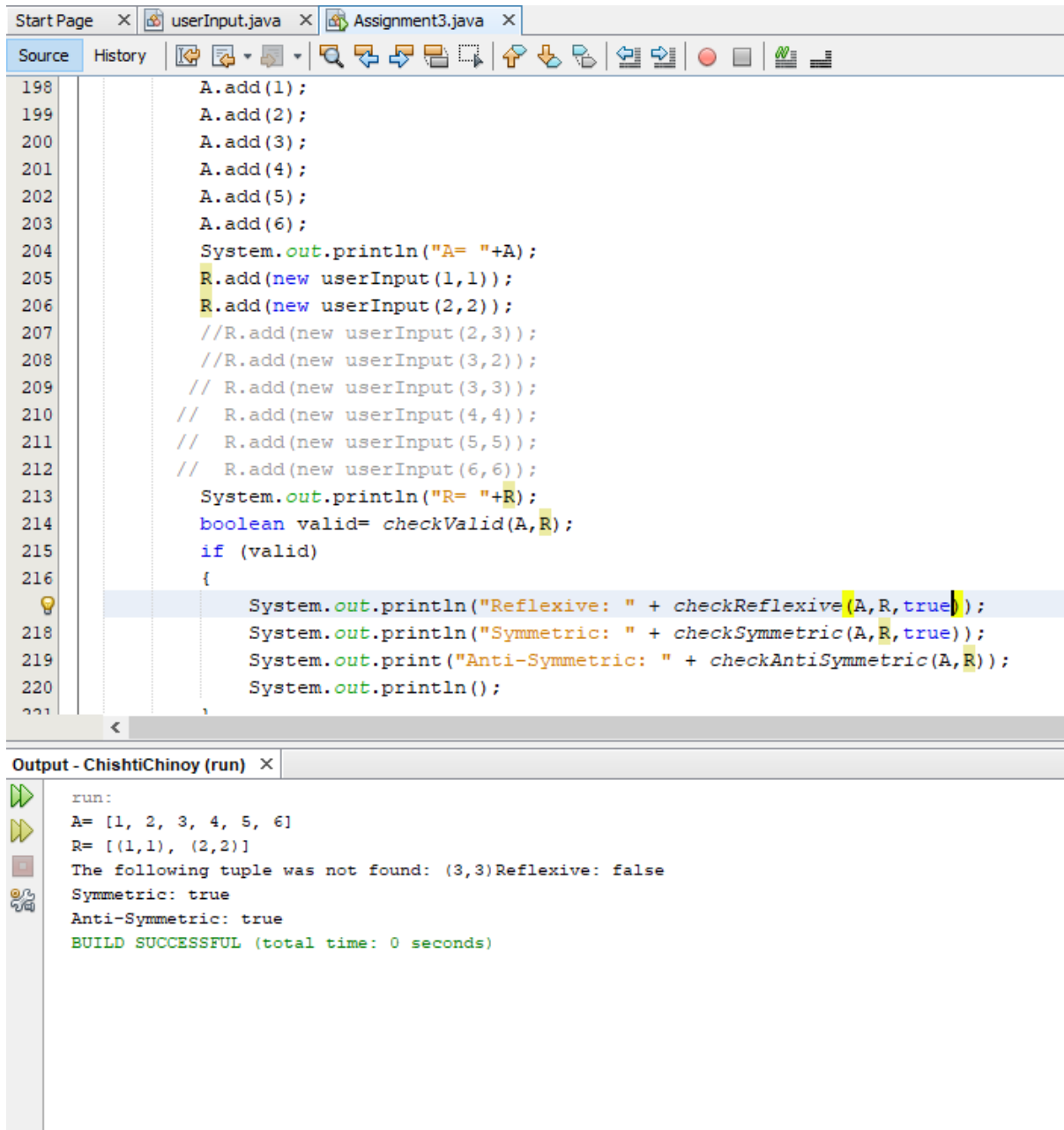
The screenshot displays an IDE with two tabs: 'userInput.java' and 'Assignment3.java'. The 'Assignment3.java' tab is active, showing a Java program. The code defines two sets, A and R, and checks for Reflexive, Symmetric, and Anti-Symmetric properties. The output window shows the results of the program execution.

```
171     A.add(1);
172     A.add(2);
173     A.add(3);
174     A.add(4);
175     A.add(5);
176     A.add(6);
177     System.out.println("A= "+A);
178     R.add(new userInput(1,1));
179     R.add(new userInput(2,2));
180     R.add(new userInput(2,3));
181     R.add(new userInput(3,2));
182     R.add(new userInput(3,3));
183     R.add(new userInput(4,4));
184     R.add(new userInput(5,5));
185     R.add(new userInput(6,6));
186     System.out.println("R= "+R);
187     boolean valid= checkValid(A,R);
188     if (valid)
189     {
190         System.out.println("Reflexive: " + checkReflexive(A,R));
191         System.out.println("Symmetric: " + checkSymmetric(A,R,true));
192         System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
193         System.out.println();
194     }
195
```

Output - ChishtiChinoy (run) ×

```
run:
A= [1, 2, 3, 4, 5, 6]
R= [(1,1), (2,2), (2,3), (3,2), (3,3), (4,4), (5,5), (6,6)]
Reflexive: true
Symmetric: true
(2,3) and (3,2) exist but 2!=3 Anti-Symmetric: false
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Test case 5:



The screenshot shows an IDE with two tabs: 'userInput.java' and 'Assignment3.java'. The 'Assignment3.java' tab is active, displaying Java code for a relation test. The code defines a set A with elements 1 through 6, and a relation R with tuples (1,1) and (2,2). It then checks for Reflexive, Symmetric, and Anti-Symmetric properties. The output window shows the results of these checks: Reflexive is false, Symmetric is true, and Anti-Symmetric is true. The build was successful.

```
198     A.add(1);
199     A.add(2);
200     A.add(3);
201     A.add(4);
202     A.add(5);
203     A.add(6);
204     System.out.println("A= "+A);
205     R.add(new userInput(1,1));
206     R.add(new userInput(2,2));
207     //R.add(new userInput(2,3));
208     //R.add(new userInput(3,2));
209     // R.add(new userInput(3,3));
210     // R.add(new userInput(4,4));
211     // R.add(new userInput(5,5));
212     // R.add(new userInput(6,6));
213     System.out.println("R= "+R);
214     boolean valid= checkValid(A,R);
215     if (valid)
216     {
217         System.out.println("Reflexive: " + checkReflexive(A,R,true));
218         System.out.println("Symmetric: " + checkSymmetric(A,R,true));
219         System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
220         System.out.println();
221     }
```

Output - ChishtiChinoy (run) ×

```
run:
A= [1, 2, 3, 4, 5, 6]
R= [(1,1), (2,2)]
The following tuple was not found: (3,3)
Reflexive: false
Symmetric: true
Anti-Symmetric: true
BUILD SUCCESSFUL (total time: 0 seconds)
```