CIS 1068: Quiz 2 (40 pts) **Name(print)________________________**

Complete the following programs, in order to print out the result of 1+2+3+4+5+…+10.

```
int total = 1;
int c = 0;
while(c < 9)
{
total = __________total+c+2____________________;
c = c +1;
}
System.out.println(total);
```

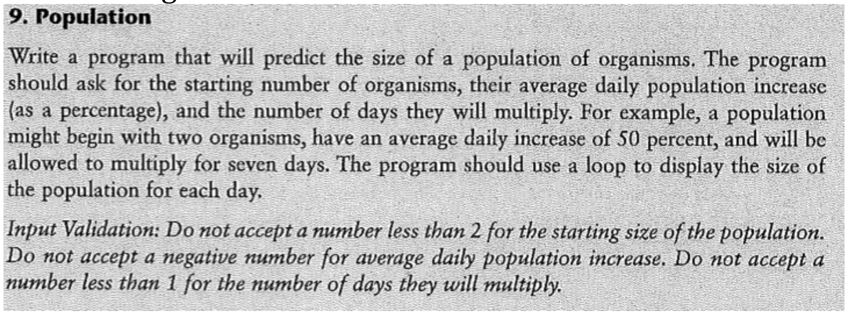
```
int total = 1;
int c = 2;

while(______c<11__________________)
{
total = total + c ;
c = c +1;
}
System.out.println(total);
```

```
int total = 1;
int c = 1;
while(c < 10)
{
total = ______total+c+1__________________ ;
c = c +1;
}
System.out.println(total);
```

```
int total = ______0___________;
int c = 0;

while(______c<10__________________)
{
c = c +1;
total = total + c;
}
System.out.println(total);
```

CIS1068 Quiz2, Population.java, on “Explanation of Issues”

See the assignment in the below.



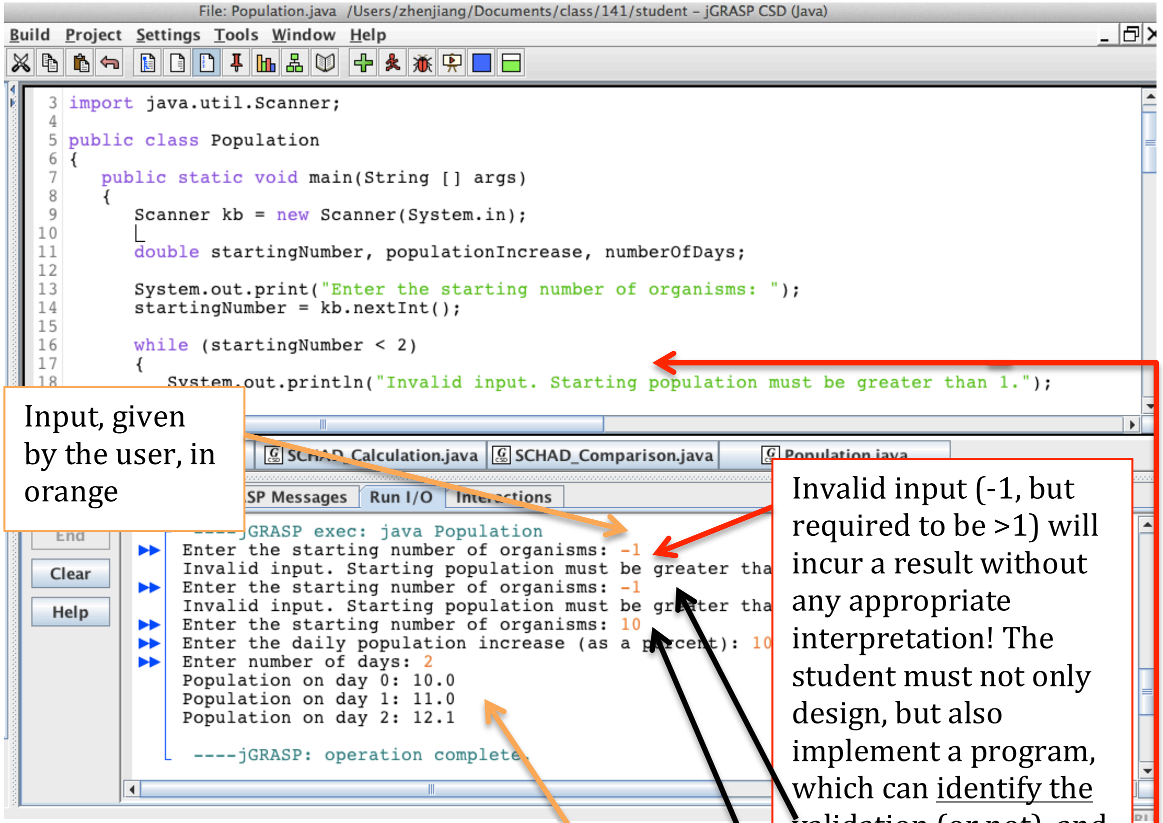
9. Population

Write a program that will predict the size of a population of organisms. The program should ask for the starting number of organisms, their average daily population increase (as a percentage), and the number of days they will multiply. For example, a population might begin with two organisms, have an average daily increase of 50 percent, and will be allowed to multiply for seven days. The program should use a loop to display the size of the population for each day.

Input Validation: Do not accept a number less than 2 for the starting size of the population. Do not accept a negative number for average daily population increase. Do not accept a number less than 1 for the number of days they will multiply.

The student work (i.e., program) is to display the size of population every day: day 0, day 1, day 2, …

Key part in assignment work for assessment: But student is asked to obtain a comprehensive view of the problem when input data has the validation issue and is crucial to the correct calculation result.



File: Population.java /Users/zhenjiang/Documents/class/141/student – jGRASP CSD (Java)
Build Project Settings Tools Window Help
SCHAD_Calculation.java
SCHAD_Comparison.java
Population.java
SP Messages
Run I/O
Interactions
End
Clear
Help

```
import java.util.Scanner;

public class Population
{
   public static void main(String [] args)
   {
      Scanner kb = new Scanner(System.in);

      double startingNumber, populationIncrease, numberOfDays;

      System.out.print("Enter the starting number of organisms: ");
      startingNumber = kb.nextInt();

      while (startingNumber < 2)
      {
         System.out.println("Invalid input. Starting population must be greater than 1.");
```

```
 ----jGRASP exec: java Population
Enter the starting number of organisms: -1
Invalid input. Starting population must be greater tha
Enter the starting number of organisms: -1
Invalid input. Starting population must be greater tha
Enter the starting number of organisms: 10
Enter the daily population increase (as a percent): 10
Enter number of days: 2
Population on day 0: 10.0
Population on day 1: 11.0
Population on day 2: 12.1

 ----jGRASP: operation complete.
```

Input, given by the user, in orange

Invalid input (-1, but required to be >1) will incur a result without any appropriate interpretation! The student must not only design, but also implement a program, which can identify the validation (or not), and then avoid using any bad data in calculation. That is, a valid data must be guaranteed at the end of input process. That requires a loop “while (<2)”, i.e., repeating input until >=2 to stop repeating!

Correct result 10, 11, and 12, after the input 10, 10, and 2

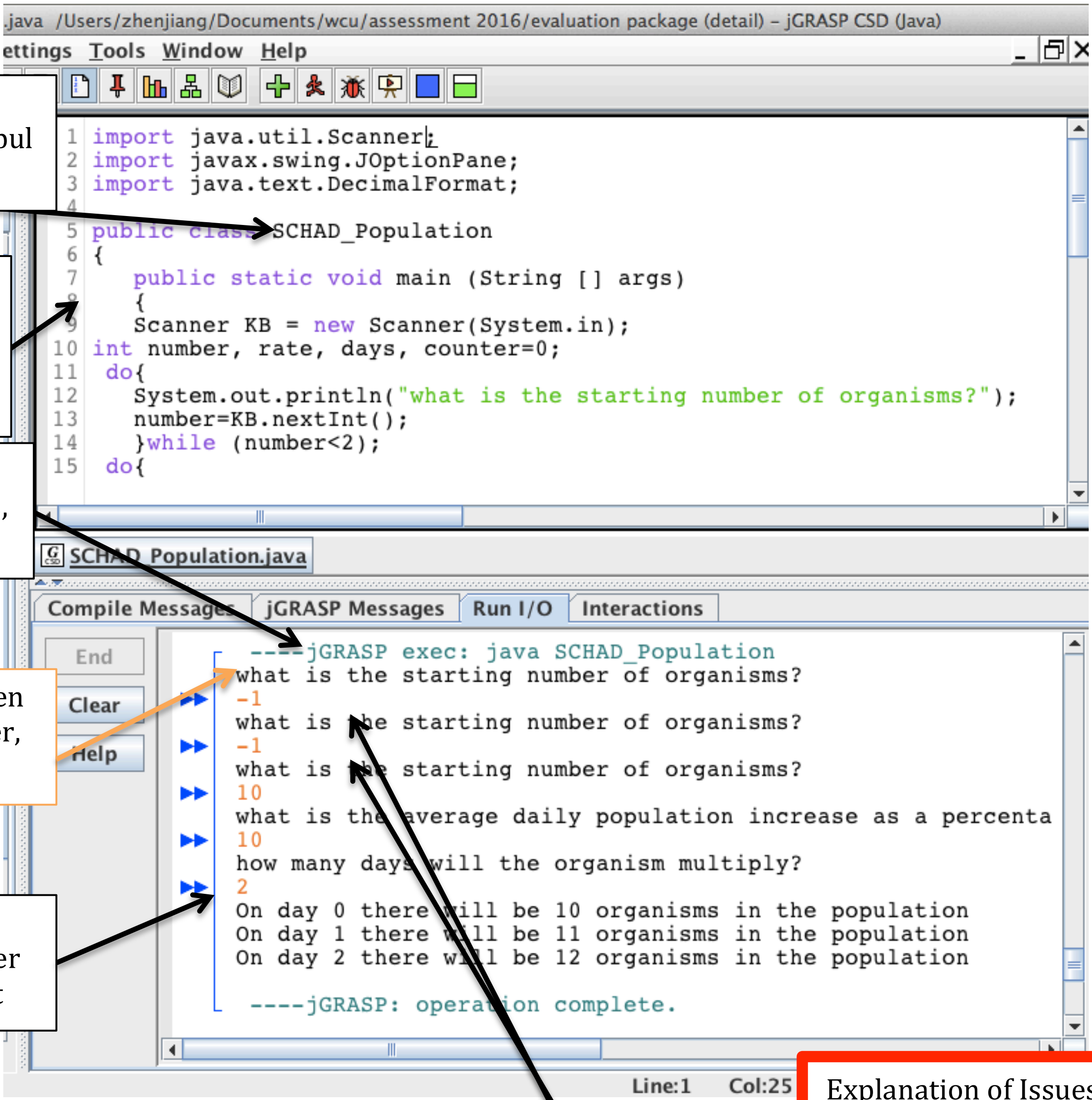
* **Desired assessment**: Note that before the programming work starts, the student should have a comprehensive view of the required population calculation, in where data validation is a critical part. As the correct result is expected at the end of execution, the design and implementation of the process to guide/correct the user’s input, which is beyond the original requirement for the number calculation, is also desired, especially when incorrect input occurs.

Complete version for reference

```java
import java.util.Scanner;
import javax.swing.JOptionPane;
import java.text.DecimalFormat;

public class Population
{
   public static void main (String [] args)
   {
   Scanner KB = new Scanner(System.in);
int number, rate, days, counter=0;
 do{
   System.out.println("what is the starting number of organisms?");
   number=KB.nextInt();
   }while (number<2);
 do{
   System.out.println("what is the daily population increase?");
   rate = KB.nextInt();
   }while(rate<0);
 do{
   System.out.println("how many days will the organism multiply?");
   days = KB.nextInt();
   }while(days<1);
 while (counter <= days){
 System.out.println("On day  "+counter+" there will be " +number +" organisms in the population");
 number=number+(number*rate)/100;
 counter=counter+1;
 }
     } }

```

Sample of high level work on “Explanation of Issues”

* Interpretation of target work for assessment of explanation, i.e., comprehensive view of problem: Before the design and implementation of the program with the desired validation check, this student should have had a comprehensive view of the validation issue in the loop program of population calculation, including the check (valid/not) and the process to ensure the valid data in calculation ☝.


.java /Users/zhenjiang/Documents/wcu/assessment 2016/evaluation package (detail) – jGRASP CSD (Java)
ettings Tools Window Help
SCHAD_Population.java
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
Line:1
Col:25

```
 1 import java.util.Scanner;
 2 import javax.swing.JOptionPane;
 3 import java.text.DecimalFormat;
 4
 5 public class SCHAD_Population
 6 {
 7    public static void main (String [] args)
 8    {
 9    Scanner KB = new Scanner(System.in);
10 int number, rate, days, counter=0;
11  do{
12    System.out.println("what is the starting number of organisms?");
13    number=KB.nextInt();
14    }while (number<2);
15  do{
```

```
 ----jGRASP exec: java SCHAD_Population
what is the starting number of organisms?
-1
what is the starting number of organisms?
-1
what is the starting number of organisms?
10
what is the average daily population increase as a percenta
10
how many days will the organism multiply?
2
On day 0 there will be 10 organisms in the population
On day 1 there will be 11 organisms in the population
On day 2 there will be 12 organisms in the population

 ----jGRASP: operation complete.
```

Program: Schad_Population.java

Line number in program

Computer print-outs, in black

Input, given by the user, in orange

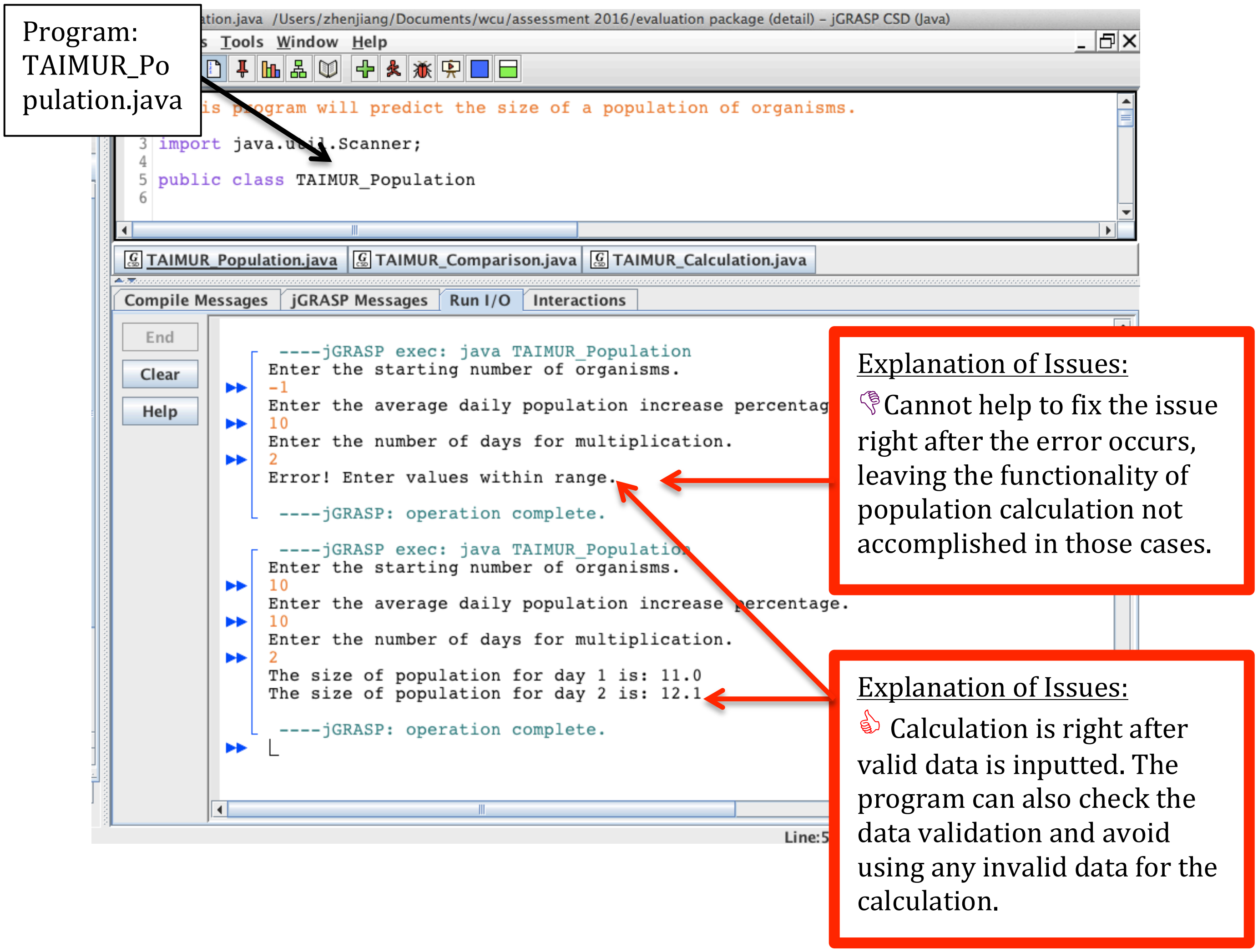
Computer result, after user input

# must be >1. -1 is invalid so the program keeps asking the right number (until 10 is inputted for a satisfaction). See a do-while loop from line 11 to line 14.

Explanation of Issues:
☝ The problem of invalid data is solved completely so that the assignment is understood in a comprehensive manner.

Sample of minimum work expected on "Explanation of Issues"

* Interpretation of target work for assessment of explanation: Before the design of the program with the desired validation check, this student should have considered the validation issue in the loop program, <u>and</u> 👍 <u>include the check (valid/not) and the process to avoid using the invalid data in calculation.</u> However, when the program encounters any invalid data, the execution stops, 👎<u>lacking the guidance for the user to correct/fix the input immediately</u> (i.e., the number entered). But this omission is not serious to impact the execution.



Program: TAIMUR_Population.java

```
is program will predict the size of a population of organisms.
import java.util.Scanner;

public class TAIMUR_Population

```

```
 ----jGRASP exec: java TAIMUR_Population
Enter the starting number of organisms.
-1
Enter the average daily population increase percentag
10
Enter the number of days for multiplication.
2
Error! Enter values within range.

 ----jGRASP: operation complete.

 ----jGRASP exec: java TAIMUR_Population
Enter the starting number of organisms.
10
Enter the average daily population increase percentage.
10
Enter the number of days for multiplication.
2
The size of population for day 1 is: 11.0
The size of population for day 2 is: 12.1

 ----jGRASP: operation complete.
```

<u>Explanation of Issues:</u>

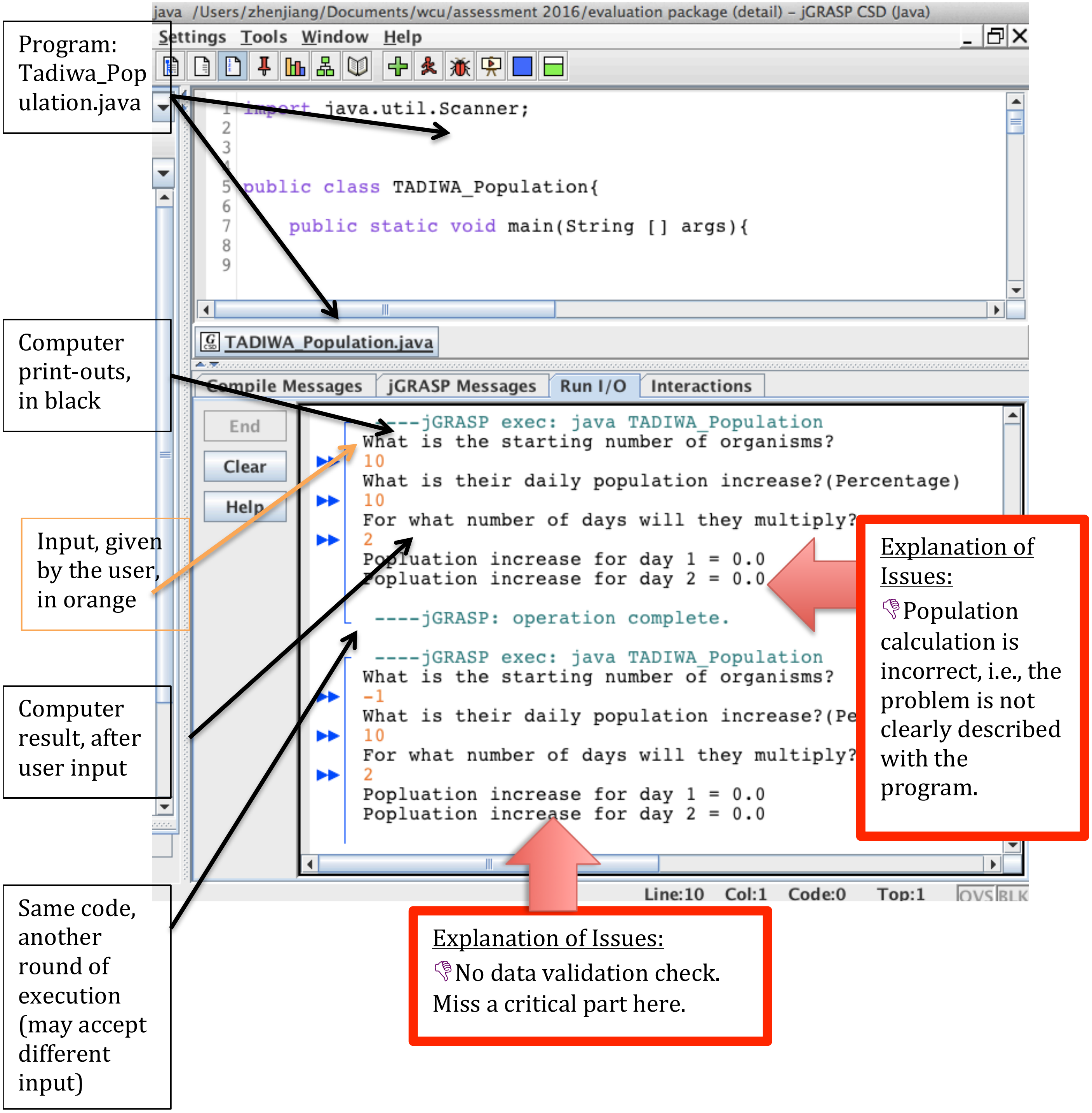
👎Cannot help to fix the issue right after the error occurs, leaving the functionality of population calculation not accomplished in those cases.

<u>Explanation of Issues:</u>

👍 Calculation is right after valid data is inputted. The program can also check the data validation and avoid using any invalid data for the calculation.

Sample of unsatisfied work on “Explanation of Issues”

* Interpretation of target work for assessment of explanation, i.e., comprehensive view of problem: No validation check! A very important part in the target problem is ignored👎. The calculation is incorrect 👎 even when valid data is inputted. That is, the loop program is not developed as guided. The target population calculation lacks a correct classification in this computer program.


java /Users/zhenjiang/Documents/wcu/assessment 2016/evaluation package (detail) – jGRASP CSD (Java)
Settings Tools Window Help
TADIWA_Population.java
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
Line:10 Col:1 Code:0 Top:1 OVS BLK

```
import java.util.Scanner;


public class TADIWA_Population{

    public static void main(String [] args){
```

```
 ----jGRASP exec: java TADIWA_Population
What is the starting number of organisms?
10
What is their daily population increase?(Percentage)
10
For what number of days will they multiply?
2
Popluation increase for day 1 = 0.0
Popluation increase for day 2 = 0.0

 ----jGRASP: operation complete.

 ----jGRASP exec: java TADIWA_Population
What is the starting number of organisms?
-1
What is their daily population increase?(Pe
10
For what number of days will they multiply?
2
Popluation increase for day 1 = 0.0
Popluation increase for day 2 = 0.0
```

Program: Tadiwa_Population.java

Computer print-outs, in black

Input, given by the user, in orange

Computer result, after user input

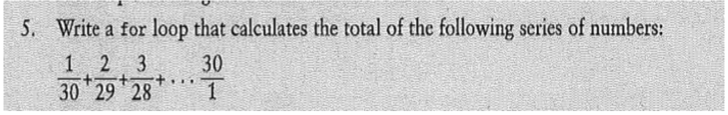
Same code, another round of execution (may accept different input)

Explanation of Issues:
👎Population calculation is incorrect, i.e., the problem is not clearly described with the program.

Explanation of Issues:
👎No data validation check. Miss a critical part here.

CIS1068 Quiz2, Calculation.java, on “Evidence” and “Conclusion”

See the assignment in the below.

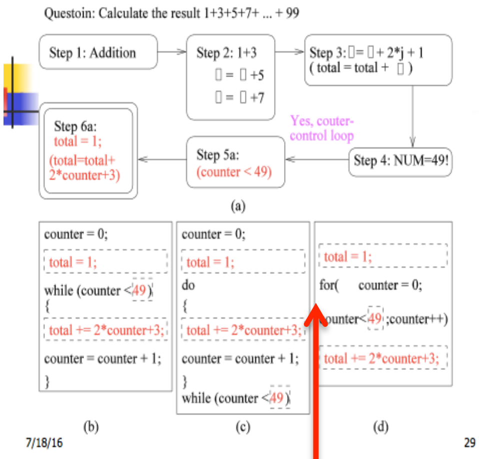


5. Write a for loop that calculates the total of the following series of numbers:

$$\frac{1}{30}+\frac{2}{29}+\frac{3}{28}+\cdots\frac{30}{1}$$

The student work (i.e., program) is to calculate the result of 1/30 + 2/29 +…+ 30/1

Key in assignment work for assessment: It asks students to review slide 37 (see the right picture) in http://www.cis.temple.edu/~zjiang/cis1068c.ppt, where a similar case was discussed: 1+3+5+…



```
counter = 0;
total = 1;
while (counter <49 )
{
total += 2*counter+3;
counter = counter + 1;
}
```

(b)

```
counter = 0;
total = 1;
do
{
total += 2*counter+3;
counter = counter + 1;
}
while (counter <49 )
```

(c)

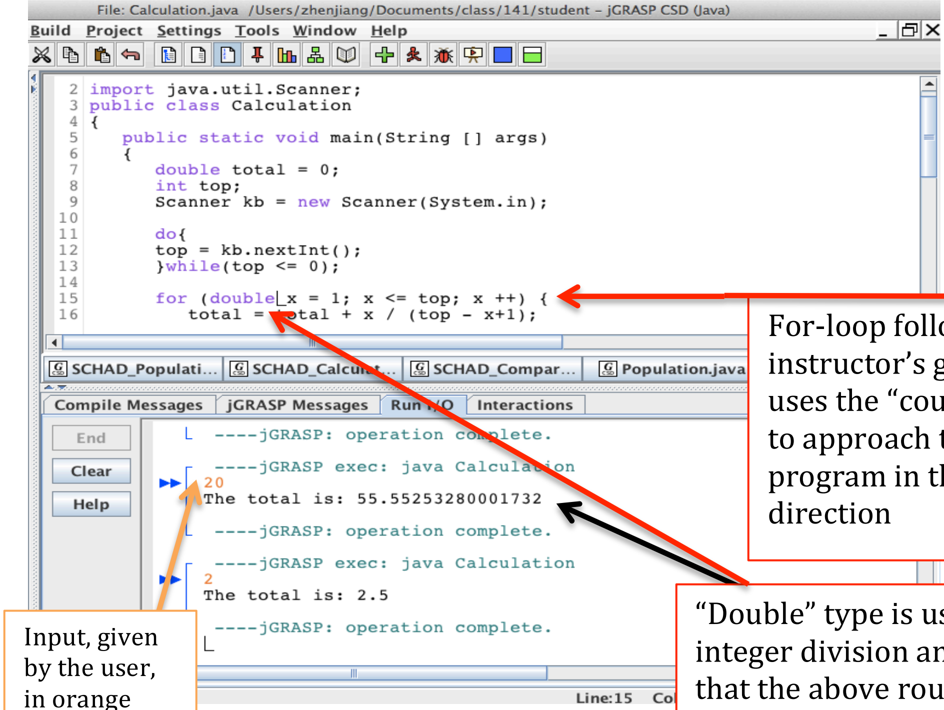
```
total = 1;
for(     counter = 0;
counter<49 ;counter++)
total += 2*counter+3;
```

(d)

When both numerator and denominator are of integer type, the computer’s result will be integer, discarding the decimal part (i.e., round-off error). The materials are discussed on slide 38 in http://www.cis.temple.edu/~zjiang/cis1068a.ppt, on the difference between “int” and “double” types.

* **Desired Assessment for “Evidence”**: Note that this assignment implies the need for students to question on the round-off error when they see integers 1 and 30 in the division 1/30.

* **Desired Assessment for “Conclusion”**: Note that this assignment implies the need for a solution in general. This requires the compatibility of any sequence from 1 to n, where n is not always 30. Such abstraction of a group of tasks is the outcome in assessment for “conclusion”, which is logically related to the target problem and reflects the student’s ability to place every piece of evidence (relevant to round-off issue and the loop development for 1+3+5+…) in the right order and to solve the proposed problem.



```
import java.util.Scanner;
public class Calculation
{
   public static void main(String [] args)
   {
      double total = 0;
      int top;
      Scanner kb = new Scanner(System.in);

      do{
      top = kb.nextInt();
      }while(top <= 0);

      for (double x = 1; x <= top; x ++) {
         total = total + x / (top - x+1);
```

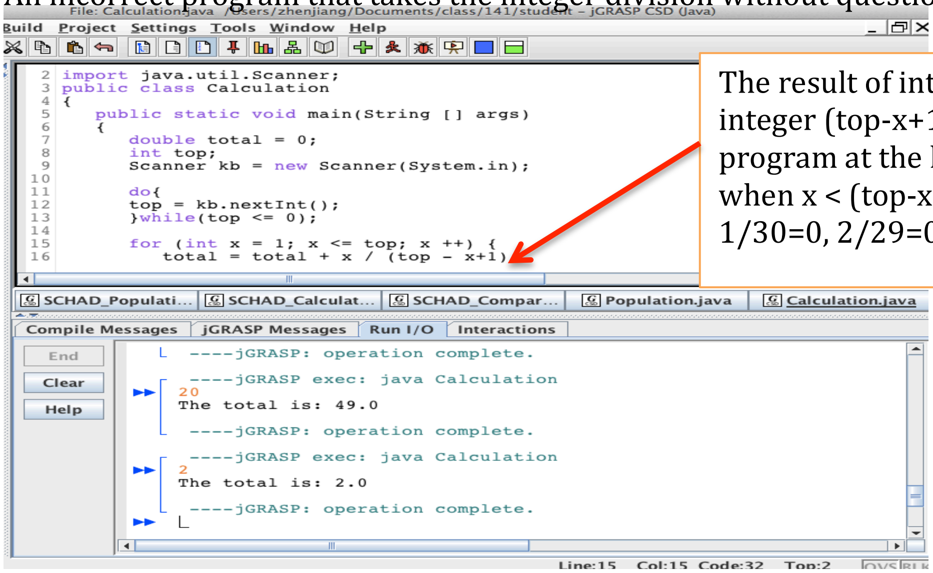
```
 └  ----jGRASP: operation complete.

 ┌  ----jGRASP exec: java Calculation
 20
 The total is: 55.55253280001732
 └  ----jGRASP: operation complete.

 ┌  ----jGRASP exec: java Calculation
 2
 The total is: 2.5

    ----jGRASP: operation complete.
 └
```

For-loop follows the instructor’s guidance and uses the “counter control” to approach the resultant program in the right direction

Input, given by the user, in orange

“Double” type is used. It is not integer division any more so that the above round-off problem can be avoided. The calculation is correct!

An incorrect program that takes the integer division without questioning the error:

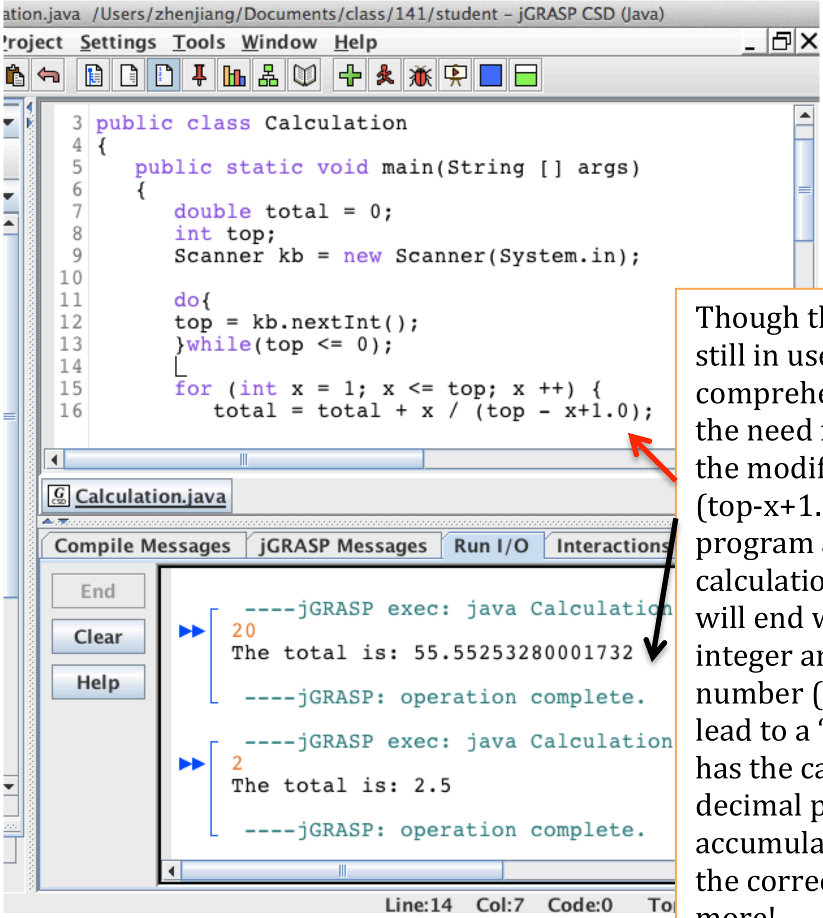

File: Calculation.java /Users/zhenjiang/Documents/class/141/student – jGRASP CSD (Java)
Build Project Settings Tools Window Help
SCHAD_Populati... SCHAD_Calculat... SCHAD_Compar... Population.java Calculation.java
Compile Messages jGRASP Messages Run I/O Interactions
End
Clear
Help
Line:15 Col:15 Code:32 Top:2

```
import java.util.Scanner;
public class Calculation
{
   public static void main(String [] args)
   {
      double total = 0;
      int top;
      Scanner kb = new Scanner(System.in);

      do{
      top = kb.nextInt();
      }while(top <= 0);

      for (int x = 1; x <= top; x ++) {
         total = total + x / (top - x+1)
```

```
 ----jGRASP: operation complete.

 ----jGRASP exec: java Calculation
20
The total is: 49.0

 ----jGRASP: operation complete.

 ----jGRASP exec: java Calculation
2
The total is: 2.0

 ----jGRASP: operation complete.
```

The result of integer x divided by integer (top-x+1), as shown in the program at the left, will end with 0 when x < (top-x+1), for instance 1/30=0, 2/29=0, ... 14/16=0!

Revised for correction: an analysis on the loop body (i.e., repetition) is expected for the use of "casting". The corresponding synthesis in the resultant program –– That is the expected part for the assessment!


ation.java /Users/zhenjiang/Documents/class/141/student – jGRASP CSD (Java)
roject Settings Tools Window Help
Calculation.java
Compile Messages jGRASP Messages Run I/O Interactions
End
Clear
Help
Line:14 Col:7 Code:0 To

```
public class Calculation
{
   public static void main(String [] args)
   {
      double total = 0;
      int top;
      Scanner kb = new Scanner(System.in);

      do{
      top = kb.nextInt();
      }while(top <= 0);

      for (int x = 1; x <= top; x ++) {
         total = total + x / (top - x+1.0);
```

```
 ----jGRASP exec: java Calculation
20
The total is: 55.55253280001732

 ----jGRASP: operation complete.

 ----jGRASP exec: java Calculation
2
The total is: 2.5

 ----jGRASP: operation complete.
```

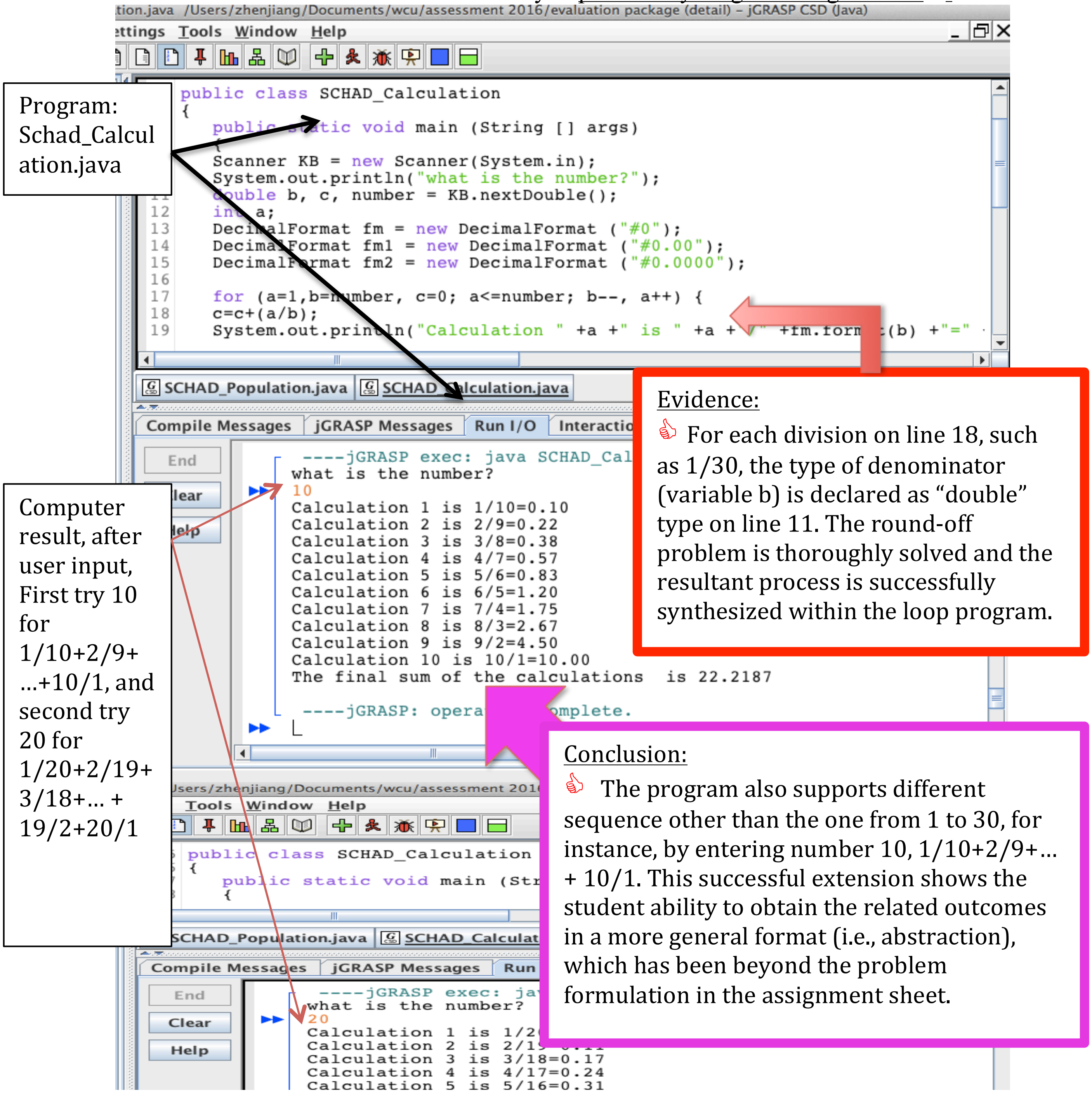
Though the integer division is still in use, the design is comprehensively questioned on the need for the casting: After the modification of (top-x+1) to (top-x+1.0), as shown in the program at the left, the calculation of the denominator will end with an addition of an integer and another double number (i.e., 1.0), which will lead to a "double" result that has the capability of carrying decimal part. Therefore, the accumulative addition will have the correct answer, not 49 any more!

Complete version for reference

```java
import java.util.Scanner;
import javax.swing.JOptionPane;
import java.text.DecimalFormat;

public class Calculation
{
   public static void main (String [] args)
   {
   Scanner KB = new Scanner(System.in);
   System.out.println("what is the number?");
   double b, c, number = KB.nextDouble();
   int a;

   for (a=1,b=number, c=0; a<=number; b--, a++) {
   c=c+(a/b);
   }
   System.out.println("The final sum is "+c);
   }
}


```

Sample of high level work on “Evidence” and on “conclusion”

- Interpretation of work for assessment of evidence: This student questions the “round-off” problem in the body of the loop. By using a “double” type denominator, say variable “b” on line 11, the integer division on line 18 is converted to a double number division. That is, the round-off problem, such as ½=0, is avoided.
- Interpretation of work for assessment of conclusion: This student obtains an abstraction of the accumulative calculation so that any similar sequence can be supported, including the one for from 1 to 30. For example, in the figure in the below, 1/10 + 2/9 + … + 10/1 and 1/20 + 2/19 + … + 20/1. This extensive work logically reflects the student’s understanding of the target problem. Its success of execution indicates student’s ability to place everything in the right order .



Program: Schad_Calculation.java

```
public class SCHAD_Calculation
{
   public static void main (String [] args)
   {
   Scanner KB = new Scanner(System.in);
   System.out.println("what is the number?");
   double b, c, number = KB.nextDouble();
12  int a;
13  DecimalFormat fm = new DecimalFormat ("#0");
14  DecimalFormat fm1 = new DecimalFormat ("#0.00");
15  DecimalFormat fm2 = new DecimalFormat ("#0.0000");
16
17  for (a=1,b=number, c=0; a<=number; b--, a++) {
18  c=c+(a/b);
19  System.out.println("Calculation " +a +" is " +a +"/" +fm.format(b) +"="
```

```
 ----jGRASP exec: java SCHAD_Cal
what is the number?
10
Calculation 1 is 1/10=0.10
Calculation 2 is 2/9=0.22
Calculation 3 is 3/8=0.38
Calculation 4 is 4/7=0.57
Calculation 5 is 5/6=0.83
Calculation 6 is 6/5=1.20
Calculation 7 is 7/4=1.75
Calculation 8 is 8/3=2.67
Calculation 9 is 9/2=4.50
Calculation 10 is 10/1=10.00
The final sum of the calculations  is 22.2187

 ----jGRASP: opera          omplete.
```

Evidence:

For each division on line 18, such as 1/30, the type of denominator (variable b) is declared as “double” type on line 11. The round-off problem is thoroughly solved and the resultant process is successfully synthesized within the loop program.

Computer result, after user input, First try 10 for 1/10+2/9+ …+10/1, and second try 20 for 1/20+2/19+ 3/18+… + 19/2+20/1

```
public class SCHAD_Calculation
{
   public static void main (Str
   {
```

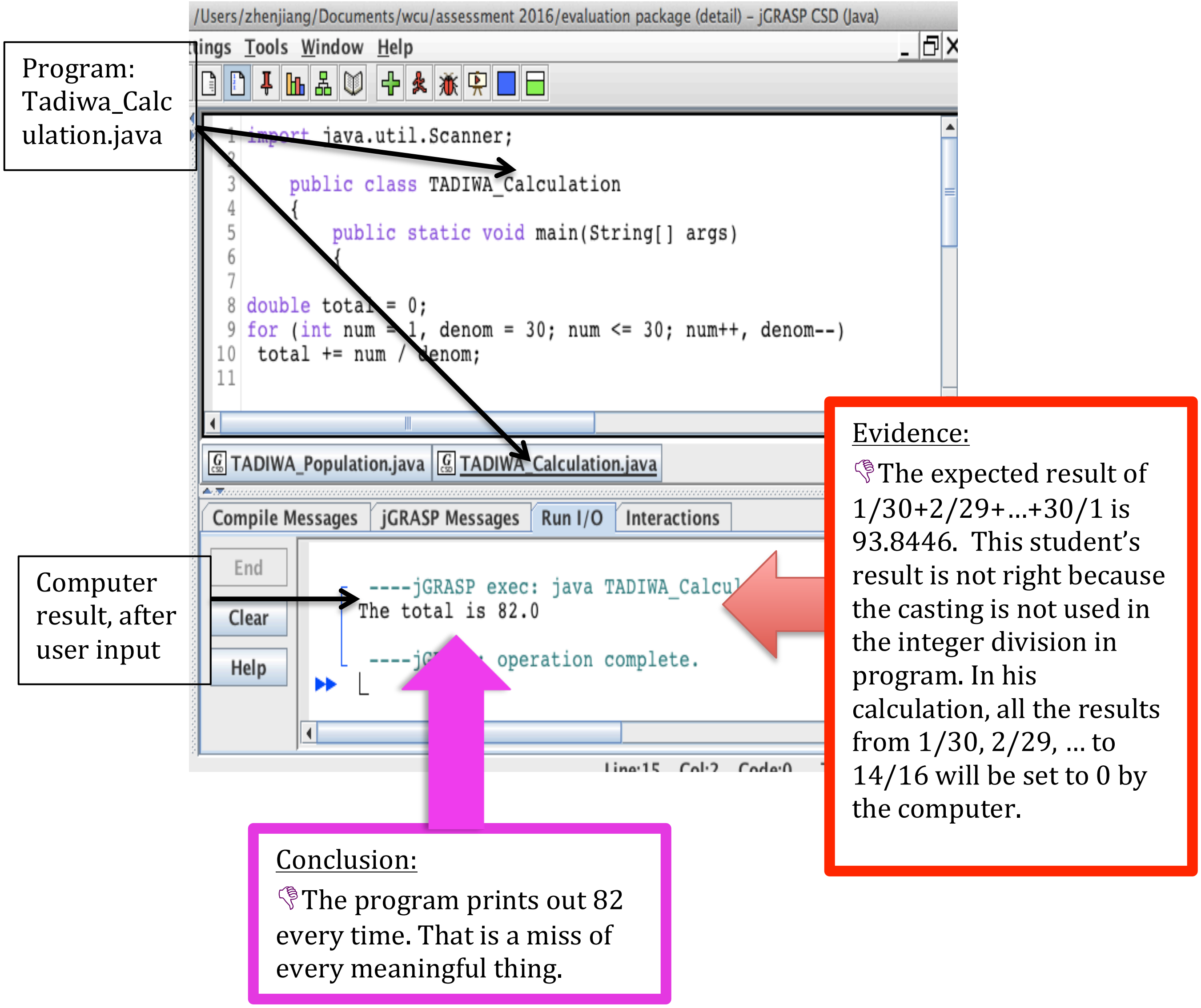
```
 ----jGRASP exec: ja
what is the number?
20
Calculation 1 is 1/2
Calculation 2 is 2/1
Calculation 3 is 3/18=0.17
Calculation 4 is 4/17=0.24
Calculation 5 is 5/16=0.31
```

Conclusion:

The program also supports different sequence other than the one from 1 to 30, for instance, by entering number 10, 1/10+2/9+… + 10/1. This successful extension shows the student ability to obtain the related outcomes in a more general format (i.e., abstraction), which has been beyond the problem formulation in the assignment sheet.

Sample work, minimum work on “Evidence” but unsatisfied on “conclusion”

- Interpretation of target work for assessment of evidence: This student did not question the “round-off” problem in the body of the loop👎. By declaring both numerator (num) and denominator (denum) in the “int” type, an integer division is conducted and round-off problem, such as ½=0, occurs. But the rest is correct.
- Interpretation of target work for assessment of conclusion: This program only calculates 1/30 + 2/29 + … + 30/1. But the result is incorrect👍. Though the structure adopted, i.e., for-loop, is appropriate, the entire program (student work) cannot be tied to anything discussed in class in a meaningful manner.



Program: Tadiwa_Calculation.java

```
import java.util.Scanner;

   public class TADIWA_Calculation
   {
      public static void main(String[] args)
      {

double total = 0;
for (int num = 1, denom = 30; num <= 30; num++, denom--)
 total += num / denom;

```

Computer result, after user input

```
 ----jGRASP exec: java TADIWA_Calcul
The total is 82.0

 ----jGRASP: operation complete.
```

Evidence:

👎The expected result of 1/30+2/29+…+30/1 is 93.8446. This student’s result is not right because the casting is not used in the integer division in program. In his calculation, all the results from 1/30, 2/29, … to 14/16 will be set to 0 by the computer.

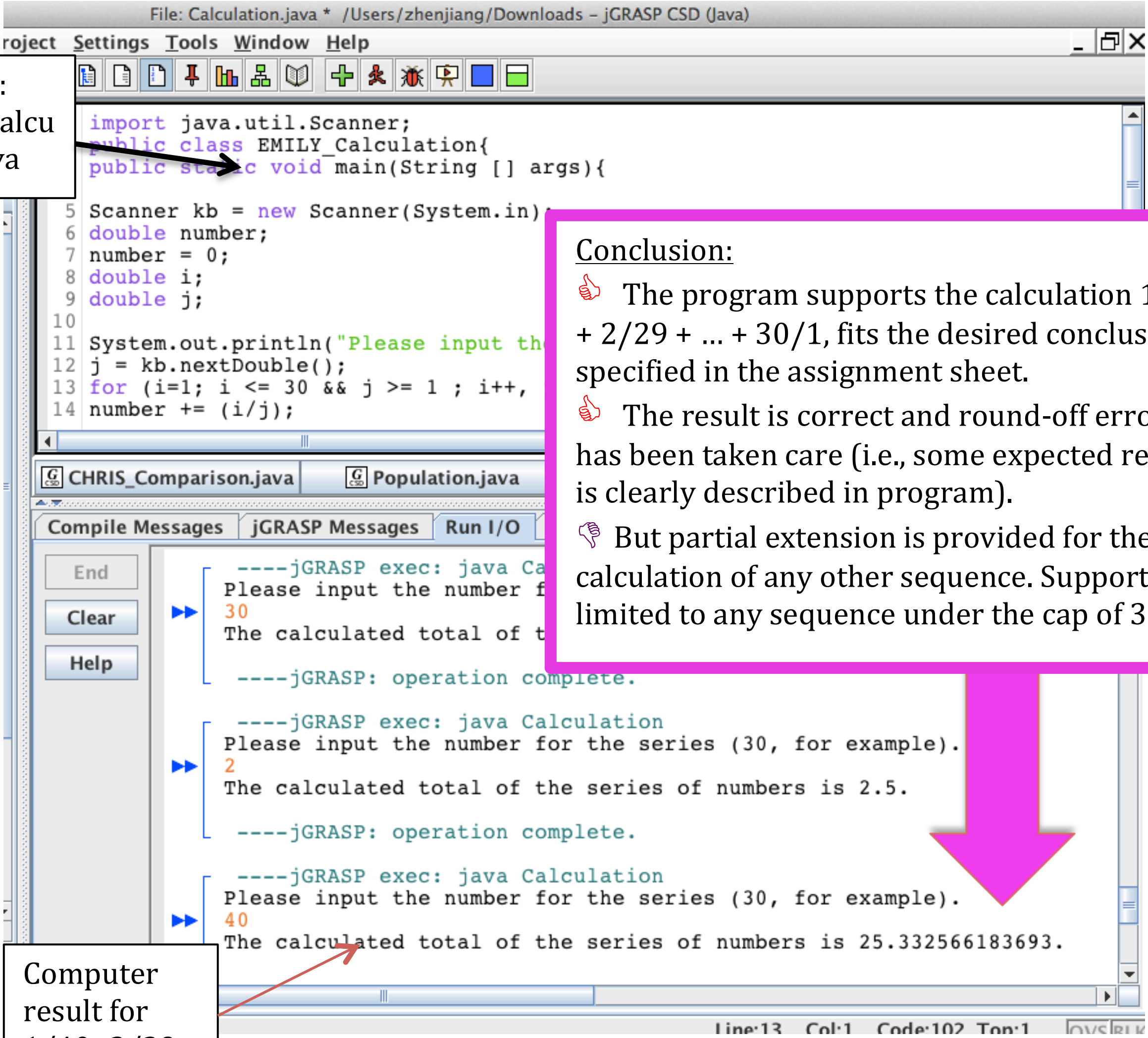
Conclusion:

👎The program prints out 82 every time. That is a miss of every meaningful thing.

Sample of minimum work expected on "Conclusion"

- Interpretation of target work for assessment of conclusion: This student can obtain 👍 an abstraction of the accumulative calculation so that any similar sequence can be supported, while the round-off problem is carefully cared and identified in the program. But that abstraction only works under the cap of 30! For instance, the result of this program for 1/40+2/39+…+40/1 is incorrect, as shown in the below (i.e., outcomes are tied to a range of cases👍, but not thoroughly a completeness👎).

Program: EMILY_Calculation.java


File: Calculation.java * /Users/zhenjiang/Downloads – jGRASP CSD (Java)
roject Settings Tools Window Help
CHRIS_Comparison.java
Population.java
Compile Messages
jGRASP Messages
Run I/O
End
Clear
Help
Line:13 Col:1 Code:102 Top:1

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

Scanner kb = new Scanner(System.in);
double number;
number = 0;
double i;
double j;

System.out.println("Please input th
j = kb.nextDouble();
for (i=1; i <= 30 && j >= 1 ; i++,
number += (i/j);
```

```
 ----jGRASP exec: java Ca
Please input the number f
30
The calculated total of t

 ----jGRASP: operation complete.

 ----jGRASP exec: java Calculation
Please input the number for the series (30, for example).
2
The calculated total of the series of numbers is 2.5.

 ----jGRASP: operation complete.

 ----jGRASP exec: java Calculation
Please input the number for the series (30, for example).
40
The calculated total of the series of numbers is 25.332566183693.
```

Conclusion:

👍 The program supports the calculation 1/30 + 2/29 + ... + 30/1, fits the desired conclusion specified in the assignment sheet.

👍 The result is correct and round-off error has been taken care (i.e., some expected result is clearly described in program).

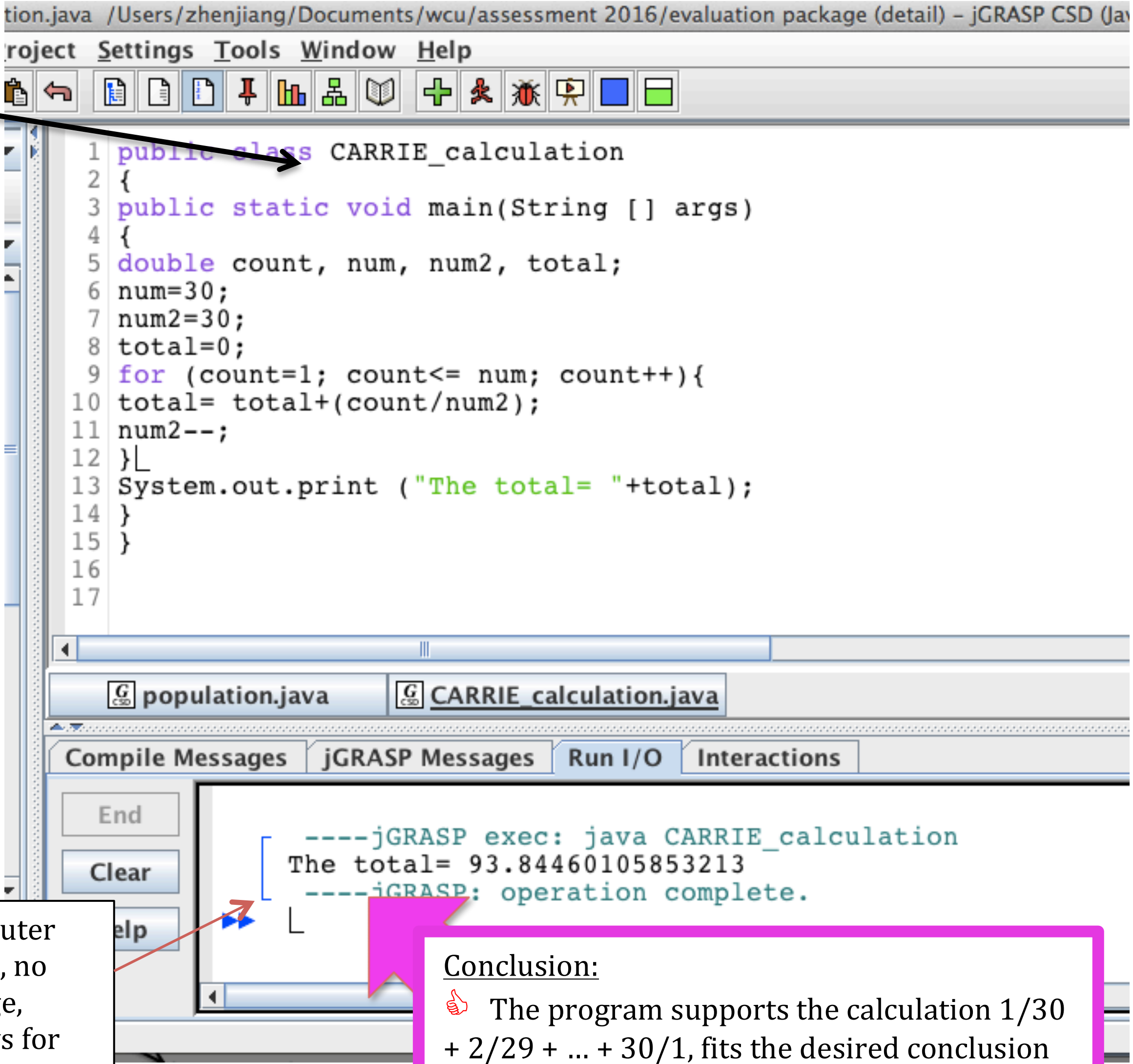
👎 But partial extension is provided for the calculation of any other sequence. Support is limited to any sequence under the cap of 30!

Computer result for 1/40+2/39+…+40/1 is incorrect.

Sample of unsatisfied work on "Conclusion"

- Interpretation of target work for assessment of conclusion: This student cannot obtain 👎 an abstraction of the accumulative calculation so that any similar sequence can be supported. That is, the result of this program is always for 1/30+2/29+…+30/1, as specified in the assignment sheet (i.e., outcome fits the desired/specified conclusion only👎). However the round-off problem is carefully cared and identified in the program 👍.

Program: CARRIE_Calculation.java


tion.java /Users/zhenjiang/Documents/wcu/assessment 2016/evaluation package (detail) – jGRASP CSD (Ja
roject Settings Tools Window Help
population.java
CARRIE_calculation.java
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
elp

```
public class CARRIE_calculation
{
public static void main(String [] args)
{
double count, num, num2, total;
num=30;
num2=30;
total=0;
for (count=1; count<= num; count++){
total= total+(count/num2);
num2--;
}
System.out.print ("The total= "+total);
}
}
```

```
 ----jGRASP exec: java CARRIE_calculation
The total= 93.84460105853213
 ----jGRASP: operation complete.
```

Computer result, no change, always for 1/30+2/29+ ...+30/1.

Conclusion:

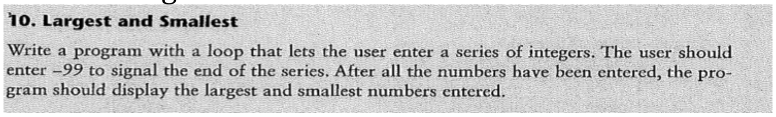
👍 The program supports the calculation 1/30 + 2/29 + ... + 30/1, fits the desired conclusion specified in the assignment sheet.

👍 The result is correct and round-off error has been taken care (i.e., some expected result is clearly described in program).

👎 But no extensive work is provided for the calculation of any other sequence.

CIS1068 Quiz2, Comparison.java, on “Assumptions”

See the assignment in the below.



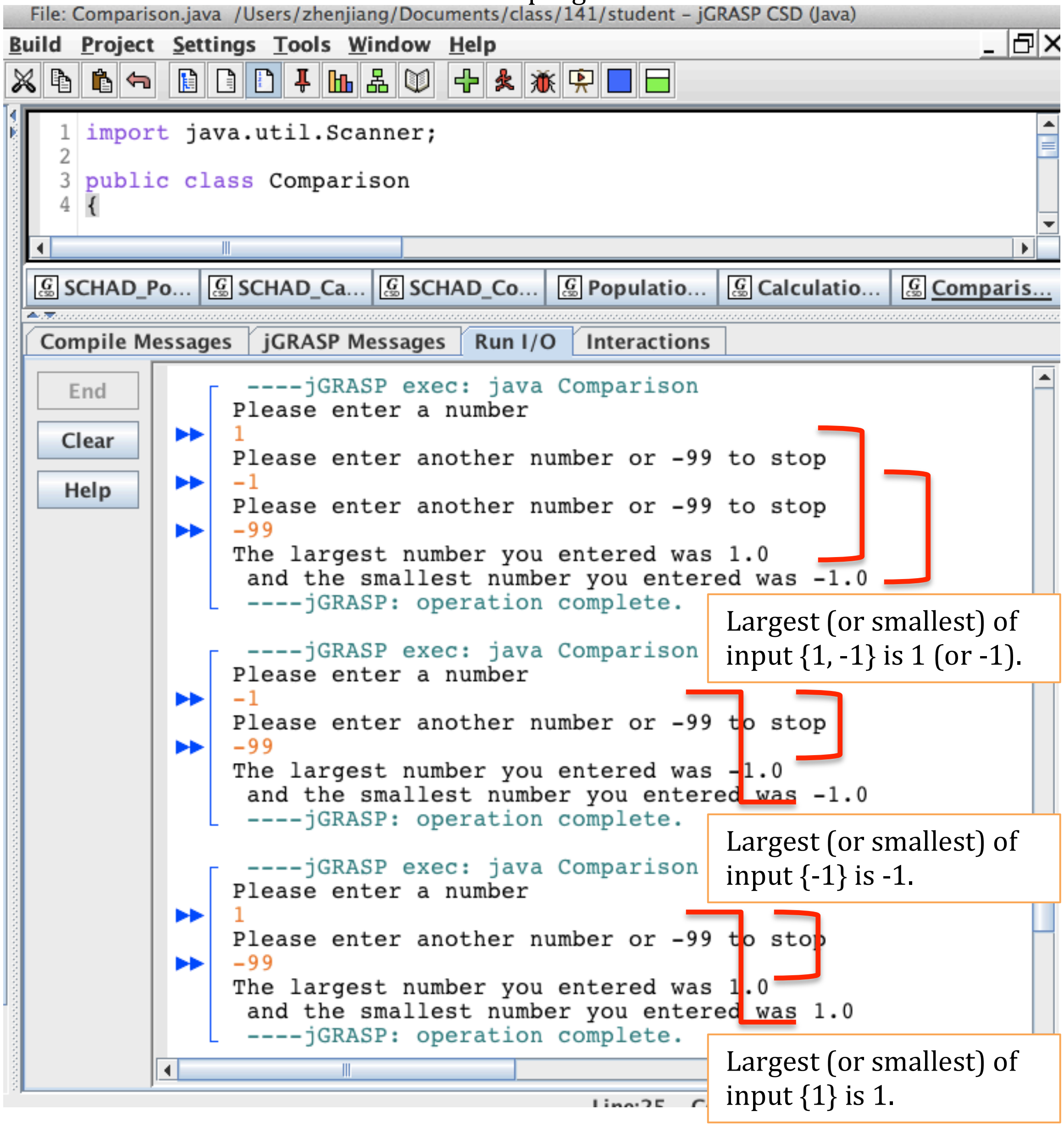
10. Largest and Smallest

Write a program with a loop that lets the user enter a series of integers. The user should enter −99 to signal the end of the series. After all the numbers have been entered, the program should display the largest and smallest numbers entered.

The student work (i.e., program) is to displays the largest and smallest entered.

Key part in assignment work for assessment: The sequence consists of any integer number, except for -99! That is, -99 cannot be in the final display.

* **Desired Assessment**: Note that, any of these two in display, largest or smallest, must be from the entered numbers (in input range)! Moreover, it is assumed that the data input process will stop at “-99”! Students’ ability to catch these assumptions is assessed here via the execution of their programs.



File: Comparison.java /Users/zhenjiang/Documents/class/141/student – jGRASP CSD (Java)

```java
import java.util.Scanner;

public class Comparison
{
```

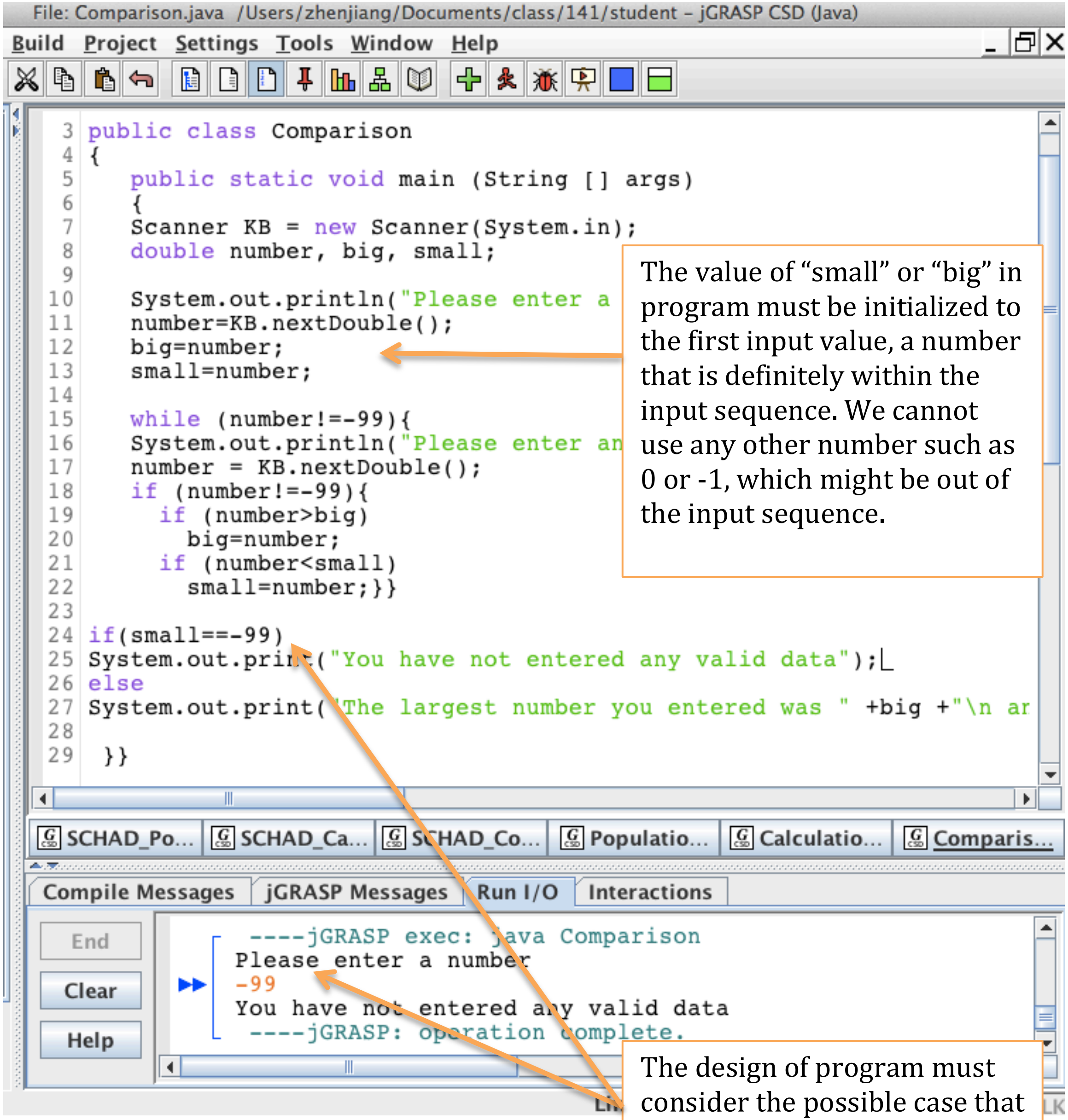
```
 ----jGRASP exec: java Comparison
Please enter a number
1
Please enter another number or -99 to stop
-1
Please enter another number or -99 to stop
-99
The largest number you entered was 1.0
 and the smallest number you entered was -1.0
 ----jGRASP: operation complete.

 ----jGRASP exec: java Comparison
Please enter a number
-1
Please enter another number or -99 to stop
-99
The largest number you entered was -1.0
 and the smallest number you entered was -1.0
 ----jGRASP: operation complete.

 ----jGRASP exec: java Comparison
Please enter a number
1
Please enter another number or -99 to stop
-99
The largest number you entered was 1.0
 and the smallest number you entered was 1.0
 ----jGRASP: operation complete.
```

Largest (or smallest) of input {1, -1} is 1 (or -1).

Largest (or smallest) of input {-1} is -1.

Largest (or smallest) of input {1} is 1.

Correct code:



```
public class Comparison
{
   public static void main (String [] args)
   {
   Scanner KB = new Scanner(System.in);
   double number, big, small;

   System.out.println("Please enter a
   number=KB.nextDouble();
   big=number;
   small=number;

   while (number!=-99){
   System.out.println("Please enter an
   number = KB.nextDouble();
   if (number!=-99){
     if (number>big)
       big=number;
     if (number<small)
       small=number;}}

if(small==-99)
System.out.print("You have not entered any valid data");
else
System.out.print("The largest number you entered was " +big +"\n ar

 }}
```

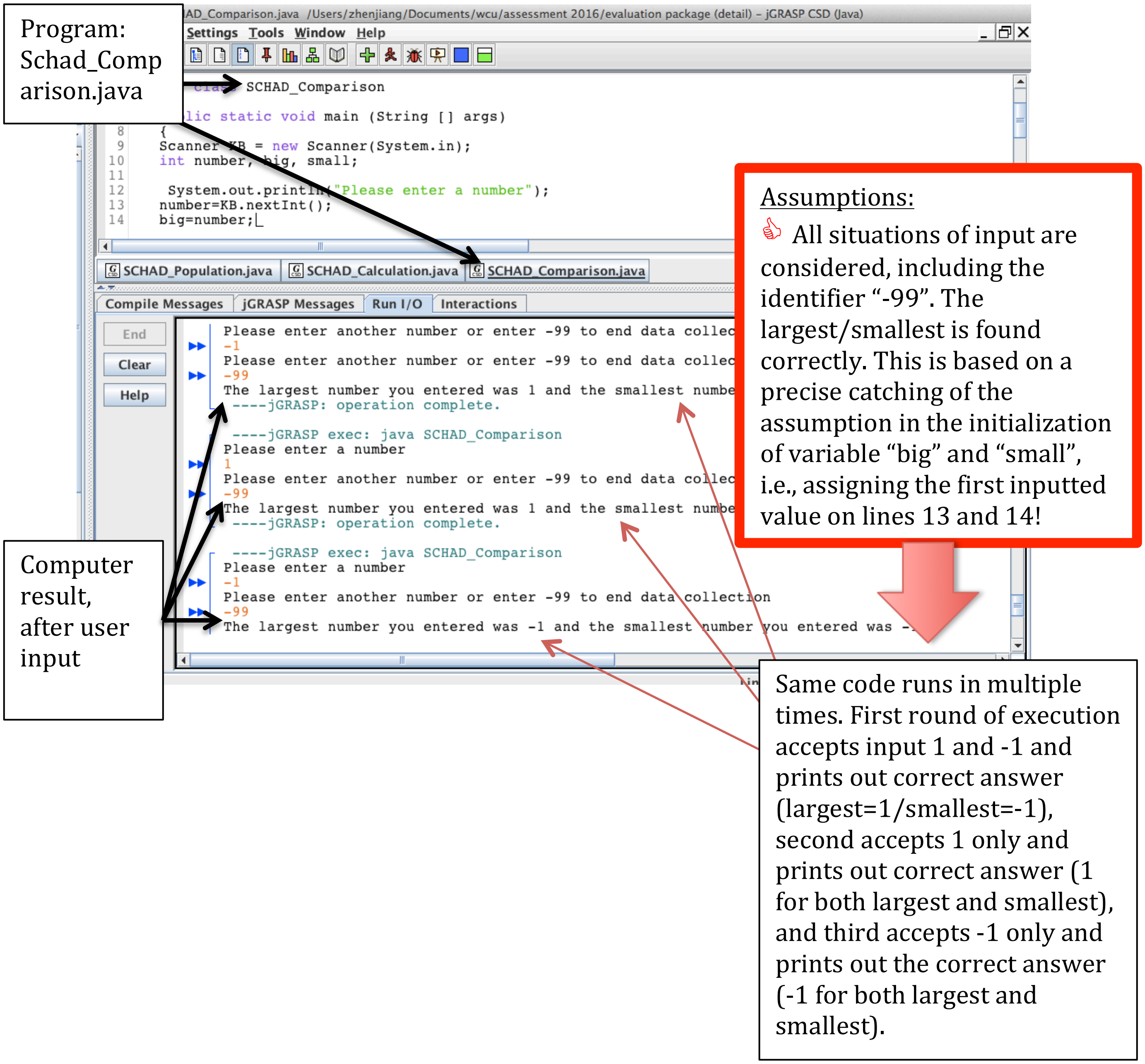
```
 ----jGRASP exec: java Comparison
Please enter a number
-99
You have not entered any valid data
 ----jGRASP: operation complete.
```

The value of “small” or “big” in program must be initialized to the first input value, a number that is definitely within the input sequence. We cannot use any other number such as 0 or -1, which might be out of the input sequence.

The design of program must consider the possible case that the input process needs to stop immediately at the beginning, without any valid data at all.

Sample of high level work on “Assumptions”

* Interpretation of the target work for assessment of assumptions: For selecting the largest and smallest value correctly (i.e., position stated in the assessment), this student precisely analyze and catch all assumptions 👍. The program is successfully implemented with such a constraint.



Program: Schad_Comparison.java

```
   clas  SCHAD_Comparison

   lic static void main (String [] args)
 8    {
 9    Scanner KB = new Scanner(System.in);
10    int number, big, small;
11
12     System.out.println("Please enter a number");
13    number=KB.nextInt();
14    big=number;
```

AD_Comparison.java /Users/zhenjiang/Documents/wcu/assessment 2016/evaluation package (detail) – jGRASP CSD (Java)
Settings Tools Window Help
SCHAD_Population.java
SCHAD_Calculation.java
SCHAD_Comparison.java
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help

```
Please enter another number or enter -99 to end data collec
-1
Please enter another number or enter -99 to end data collec
-99
The largest number you entered was 1 and the smallest numbe
 ----jGRASP: operation complete.

 ----jGRASP exec: java SCHAD_Comparison
Please enter a number
1
Please enter another number or enter -99 to end data collec
-99
The largest number you entered was 1 and the smallest numbe
 ----jGRASP: operation complete.

 ----jGRASP exec: java SCHAD_Comparison
Please enter a number
-1
Please enter another number or enter -99 to end data collection
-99
The largest number you entered was -1 and the smallest number you entered was -
```

Assumptions:

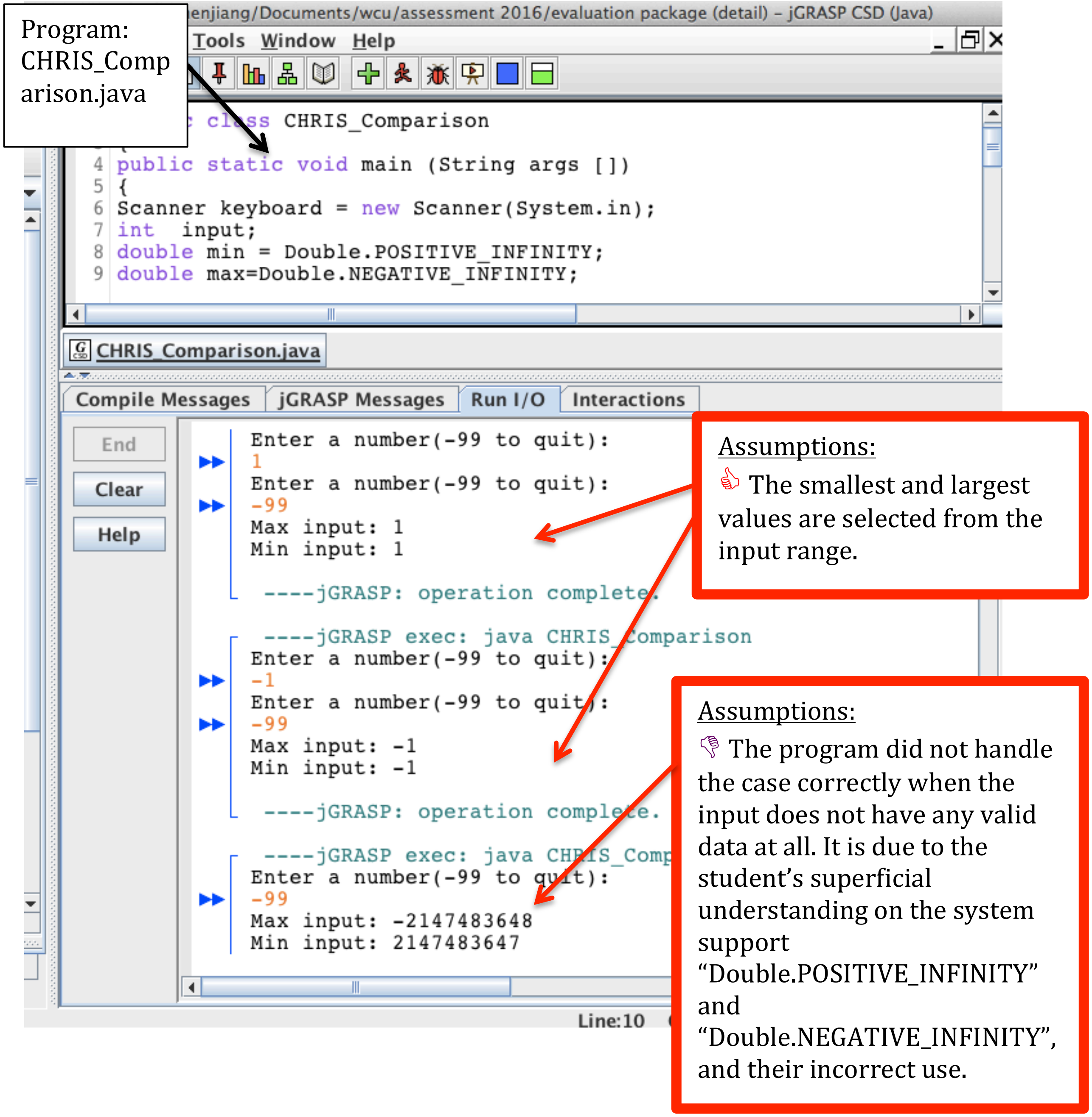
👍 All situations of input are considered, including the identifier “-99”. The largest/smallest is found correctly. This is based on a precise catching of the assumption in the initialization of variable “big” and “small”, i.e., assigning the first inputted value on lines 13 and 14!

Computer result, after user input

Same code runs in multiple times. First round of execution accepts input 1 and -1 and prints out correct answer (largest=1/smallest=-1), second accepts 1 only and prints out correct answer (1 for both largest and smallest), and third accepts -1 only and prints out the correct answer (-1 for both largest and smallest).

Sample of minimum work expected on “Assumptions”

* Interpretation of target work for assessment of assumptions: The assumption that these two values in display (the largest and the smallest) come from the entered numbers is followed👍. The program accepts the assumption 👍 that uses “-99” as the identifier to stop the execution. However, this program 👎 adopts a different problem description without a thorough analysis👎, leaving unnecessary information in display when no valid number is inputted (see the below display at the stop after “-99” is entered before any other valid number).



Program: CHRIS_Comparison.java

enjiang/Documents/wcu/assessment 2016/evaluation package (detail) – jGRASP CSD (Java)

```
c class CHRIS_Comparison

public static void main (String args [])
{
Scanner keyboard = new Scanner(System.in);
int  input;
double min = Double.POSITIVE_INFINITY;
double max=Double.NEGATIVE_INFINITY;
```

```
Enter a number(-99 to quit):
1
Enter a number(-99 to quit):
-99
Max input: 1
Min input: 1

 ----jGRASP: operation complete.

 ----jGRASP exec: java CHRIS_Comparison
Enter a number(-99 to quit):
-1
Enter a number(-99 to quit):
-99
Max input: -1
Min input: -1

 ----jGRASP: operation complete.

 ----jGRASP exec: java CHRIS_Comp
Enter a number(-99 to quit):
-99
Max input: -2147483648
Min input: 2147483647
```

Assumptions:

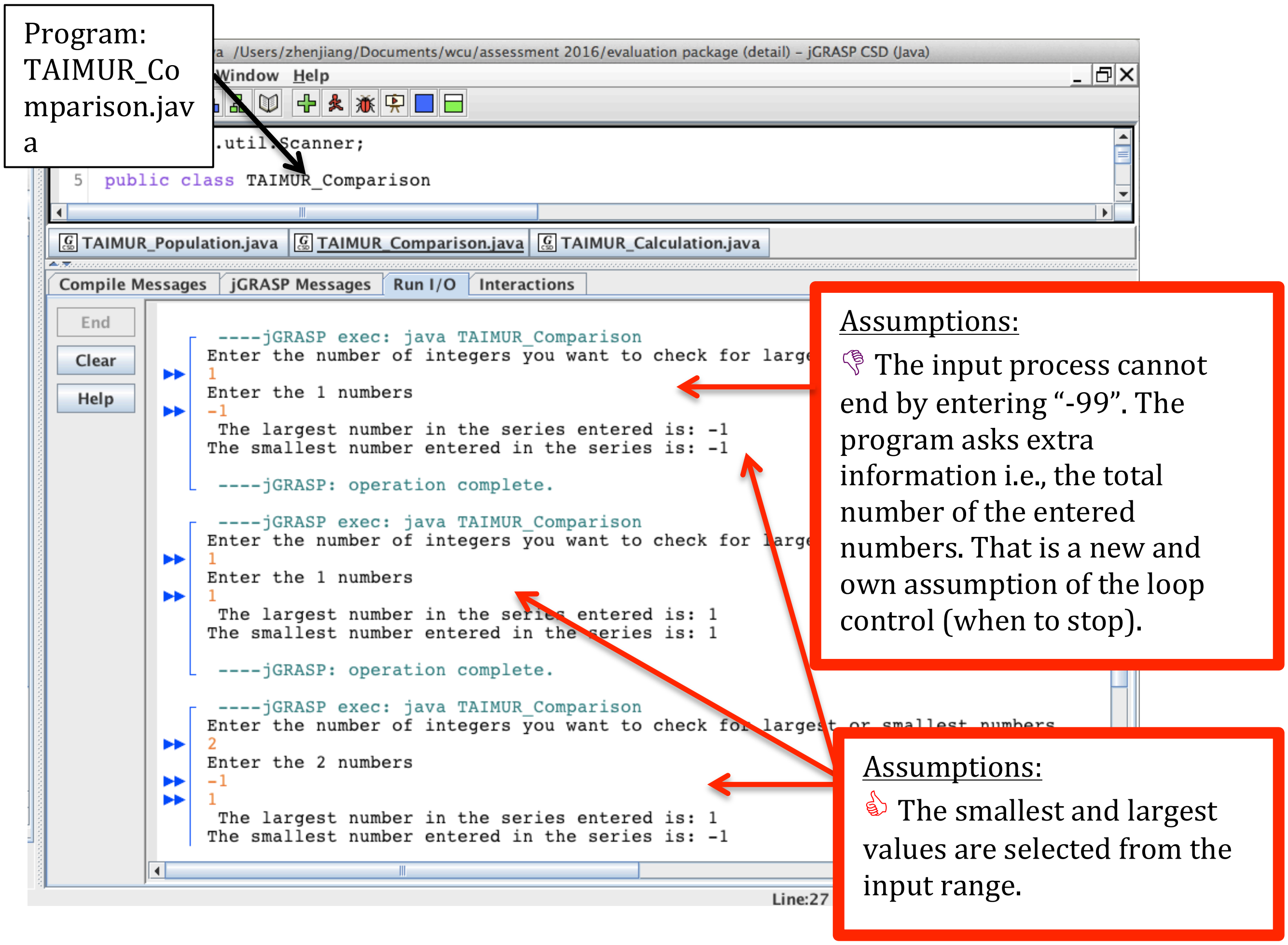
👍 The smallest and largest values are selected from the input range.

Assumptions:

👎 The program did not handle the case correctly when the input does not have any valid data at all. It is due to the student’s superficial understanding on the system support “Double.POSITIVE_INFINITY” and “Double.NEGATIVE_INFINITY”, and their incorrect use.

Sample of unsatisfied work on "Assumptions"

* Interpretation of target work for assessment of assumptions: The assumption that these two values in display (the largest and the smallest) come from the entered numbers is followed👍. However, this program 👎 did not follow (or ignore) the assumption in assignment that needs the identifier "-99" to stop the input process. This student 👎 adds a new and own assumption that the entire input process is controlled by a counter's value. Overall, the assumption is identified and applied, but not in an exactly accurate manner.



Program: TAIMUR_Comparison.java

```
.util.Scanner;
public class TAIMUR_Comparison
```

```
 ----jGRASP exec: java TAIMUR_Comparison
Enter the number of integers you want to check for large
1
Enter the 1 numbers
-1
 The largest number in the series entered is: -1
The smallest number entered in the series is: -1

 ----jGRASP: operation complete.

 ----jGRASP exec: java TAIMUR_Comparison
Enter the number of integers you want to check for large
1
Enter the 1 numbers
1
 The largest number in the series entered is: 1
The smallest number entered in the series is: 1

 ----jGRASP: operation complete.

 ----jGRASP exec: java TAIMUR_Comparison
Enter the number of integers you want to check for largest or smallest numbers
2
Enter the 2 numbers
-1
1
 The largest number in the series entered is: 1
The smallest number entered in the series is: -1
```

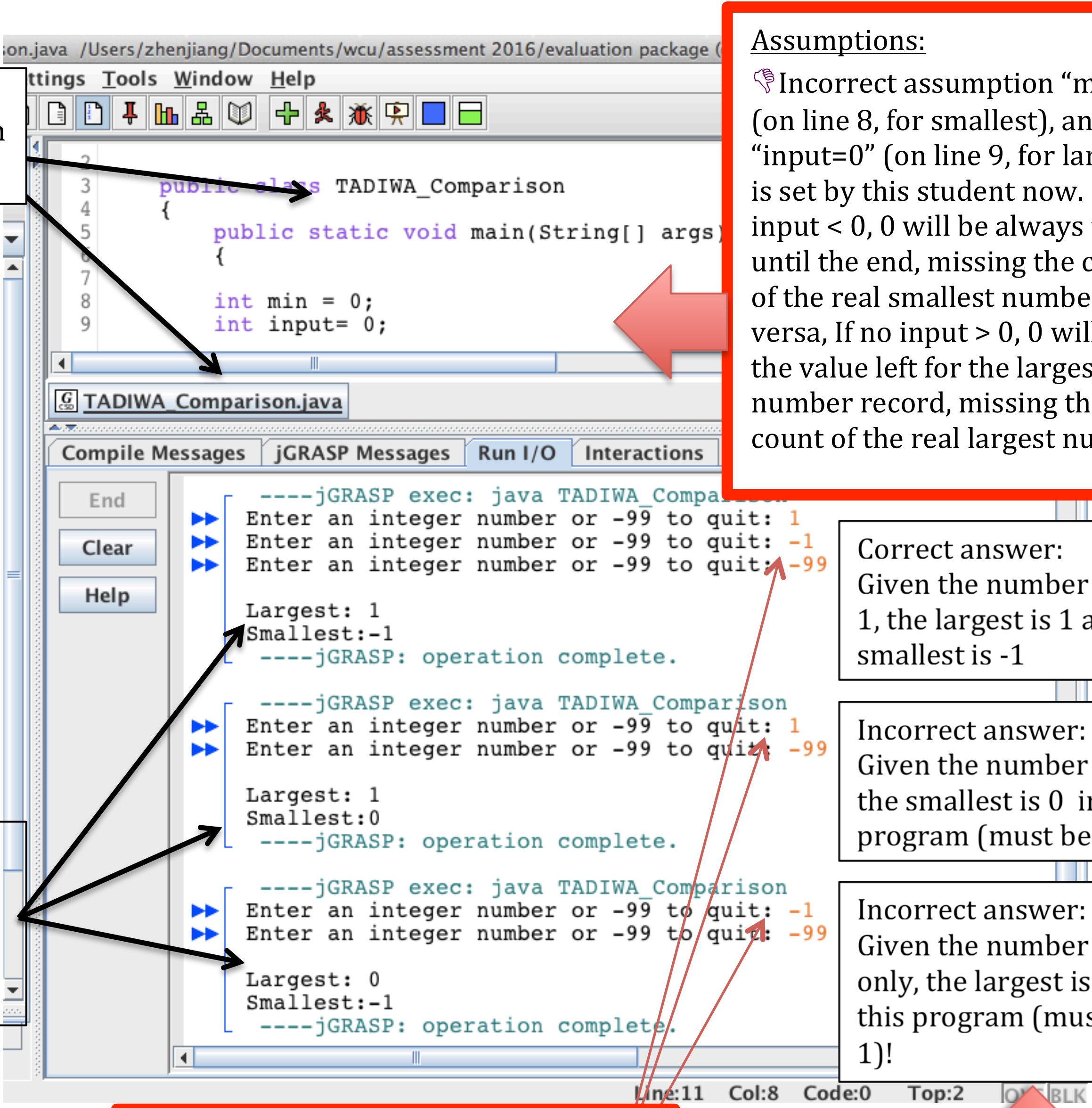
Assumptions:
👎 The input process cannot end by entering "-99". The program asks extra information i.e., the total number of the entered numbers. That is a new and own assumption of the loop control (when to stop).

Assumptions:
👍 The smallest and largest values are selected from the input range.

Sample of unsatisfied work on "Assumptions"

* Interpretation of target work for assessment of assumptions: The assumption that these two values in display (the largest and the smallest) must come from the entered numbers is NOT followed👎. Moreover, in the program, the student adds his own assumption 👎that there must be one negative and one positive both (2) numbers existing in the entered numbers. Overall, no assumption required is identified and applied, but incorrect assumption is added.

Program: Tadiwa_Comparison.java



on.java /Users/zhenjiang/Documents/wcu/assessment 2016/evaluation package (
ttings Tools Window Help
TADIWA_Comparison.java
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
Line:11 Col:8 Code:0 Top:2 OVR BLK

```
2
3    public class TADIWA_Comparison
4    {
5        public static void main(String[] args)
6        {
7
8            int min = 0;
9            int input= 0;
```

```
 ----jGRASP exec: java TADIWA_Comparison
Enter an integer number or -99 to quit: 1
Enter an integer number or -99 to quit: -1
Enter an integer number or -99 to quit: -99

Largest: 1
Smallest:-1
 ----jGRASP: operation complete.

 ----jGRASP exec: java TADIWA_Comparison
Enter an integer number or -99 to quit: 1
Enter an integer number or -99 to quit: -99

Largest: 1
Smallest:0
 ----jGRASP: operation complete.

 ----jGRASP exec: java TADIWA_Comparison
Enter an integer number or -99 to quit: -1
Enter an integer number or -99 to quit: -99

Largest: 0
Smallest:-1
 ----jGRASP: operation complete.
```

Assumptions:
👎Incorrect assumption "min=0" (on line 8, for smallest), and "input=0" (on line 9, for largest) is set by this student now. If no input < 0, 0 will be always there until the end, missing the count of the real smallest number. Vice versa, If no input > 0, 0 will be the value left for the largest number record, missing the count of the real largest number.

Correct answer: Given the number 1 and -1, the largest is 1 and the smallest is -1

Incorrect answer: Given the number 1 only, the smallest is 0 in this program (must be 1)!

Incorrect answer: Given the number -1 only, the largest is 0 in this program (must be -1)!

Computer result, after user input

Assumptions:
👍Same code runs in multiple times. First round of execution accepts input 1 and -1, second accepts 1 only, and third accepts -1 only. -99 is the identifier to end the process.

Assumptions:
👎Assumption of the display numbers from the entered numbers is not followed.