Vanna Moore
CMPS 390

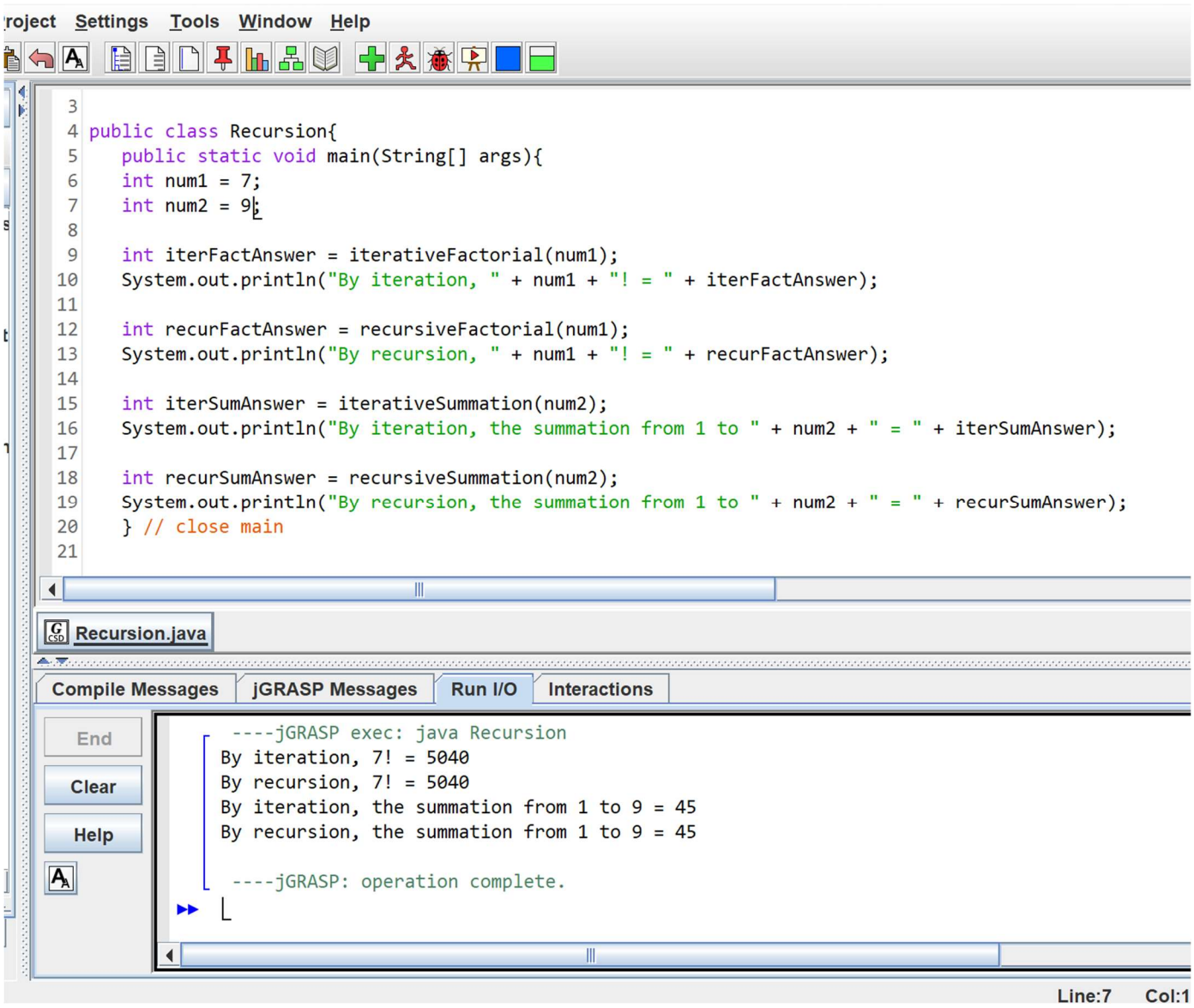
Code

```
/* This program calculates summations and factorials by passing a number
into methods outside of the main method to do each calculation. It uses
four methods: iterativeFactorial(int num), recursiveFactorial(int num),
iterativeSummation(int num), and recursiveSummation(int num). */


public class Recursion{

   public static void main(String[] args){

   int num1 = 23;

   int num2 = 25;


   int iterFactAnswer = iterativeFactorial(num1);

   System.out.println("By iteration, " + num1 + "! = " + iterFactAnswer);


   int recurFactAnswer = recursiveFactorial(num1);

   System.out.println("By recursion, " + num1 + "! = " + recurFactAnswer);


   int iterSumAnswer = iterativeSummation(num2);

   System.out.println("By iteration, the summation from 1 to " + num2 + "
= " + iterSumAnswer);


   int recurSumAnswer = recursiveSummation(num2);

   System.out.println("By recursion, the summation from 1 to " + num2 + "
= " + recurSumAnswer);

   } // close main


/* Method: claculate factorial by iteration */

         static int iterativeFactorial(int num){

            int j;

            int prod = 1;

            for(j = 1; j <= num; j++ ){

               prod = prod * j;
```

```
            }
            return prod;
         }

/* Method: calculate factorial by recursion */
         static int recursiveFactorial(int num){
            int j;
            int prod;
            if(num == 1){
               return 1;
            }
               else{
                  return num * recursiveFactorial(num -1);
               }
         }

/* Method: calculate summation by iteration */
         static int iterativeSummation(int num){
            int j;
            int sum;
            sum = 0;
            for(j = 1; j <= num; j++){
               sum = sum +j;
            }
            return sum;
         }

/* Method: calculate summation by recursion */
         static int recursiveSummation(int num){
            int j;
            int sum = 1;
            if(num == 1){
```

```
            return 1;
        }
        else{
            return num + recursiveSummation(num - 1);
        }

    }
}
```

Running



Method: Factorial calculation by iteration

```
   int recurSumAnswer = recursiveSummation(num2);
   System.out.println("By recursion, the summation from 1 to " + num2 + " = " + recurSumAnswer);
   } // close main

/* Method: claculate factorial by iteration */
         static int iterativeFactorial(int num){
            int j;
            int prod = 1;
            for(j = 1; j <= num; j++ ){
               prod = prod * j;
            }
            return prod;
         }

/* Method: calculate factorial by recursion */
         static int recursiveFactorial(int num){
            int j;
            int prod;
```

```
----jGRASP exec: java Recursion
By iteration, 21! = -1195114496
By recursion, 21! = -1195114496
By iteration, the summation from 1 to 19 = 190
By recursion, the summation from 1 to 19 = 190

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

Method: Factorial calculation by recursion

oject Settings Tools Window Help

```java
        }

/* Method: calculate factorial by recursion */
        static int recursiveFactorial(int num){
           int j;
           int prod;
           if(num == 1){
              return 1;
           }
              else{
                 return num * recursiveFactorial(num -1);
              }
        }

/* Method: calculate summation by iteration */
        static int iterativeSummation(int num){
           int j;
           int sum;
           sum = 0;
```

Recursion.java

Compile Messages | jGRASP Messages | Run I/O | Interactions

End
Clear
Help

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

----jGRASP exec: java Recursion
By iteration, 13! = 1932053504
By recursion, 13! = 1932053504
By iteration, the summation from 1 to 17 = 153
By recursion, the summation from 1 to 17 = 153
```

Method: Summation calculation by iteration:

```
            }
        }

/* Method: calculate summation by iteration */
        static int iterativeSummation(int num){
           int j;
           int sum;
           sum = 0;
           for(j = 1; j <= num; j++){
              sum = sum +j;
           }
           return sum;
        }

/* Method: calculate summation by recursion */
        static int recursiveSummation(int num){
           int j;
           int sum = 1;
           if(num == 1){
```

Recursion.java

Compile Messages | jGRASP Messages | Run I/O | Interactions

End | Clear | Help

```
----jGRASP exec: java Recursion
By iteration, 11! = 39916800
By recursion, 11! = 39916800
By iteration, the summation from 1 to 15 = 120
By recursion, the summation from 1 to 15 = 120

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

Line:7 Col:17 Code:59

Method: Summation calculation by recursion:

```
            }
            return sum;
         }

/* Method: calculate summation by recursion */
         static int recursiveSummation(int num){
            int j;
            int sum = 1;
            if(num == 1){
               return 1;
            }
            else{
               return num + recursiveSummation(num - 1);
            }

         }
}
```

Recursion.java

Compile Messages | jGRASP Messages | Run I/O | Interactions

End | Clear | Help

```
 ----jGRASP exec: java Recursion
By iteration, 23! = 862453760
By recursion, 23! = 862453760
By iteration, the summation from 1 to 25 = 325
By recursion, the summation from 1 to 25 = 325

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

Line:7 Col:17 Code:59