

Java

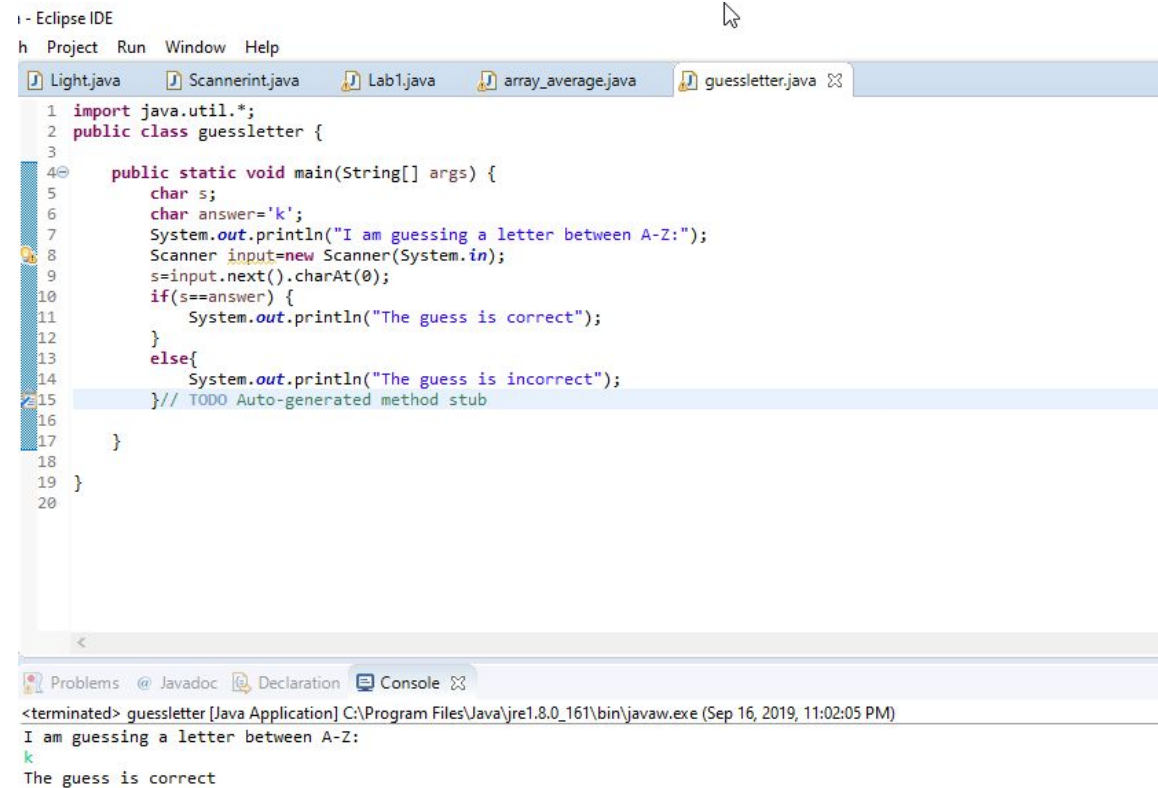
Control Statements

If condition

- The if statement is Java's conditional branch statement. It can be used to route program execution through two different paths. Here is the general form of the if statement:

```
if (condition) statement1;  
else statement2;
```

Example



```
i - Eclipse IDE
h Project Run Window Help

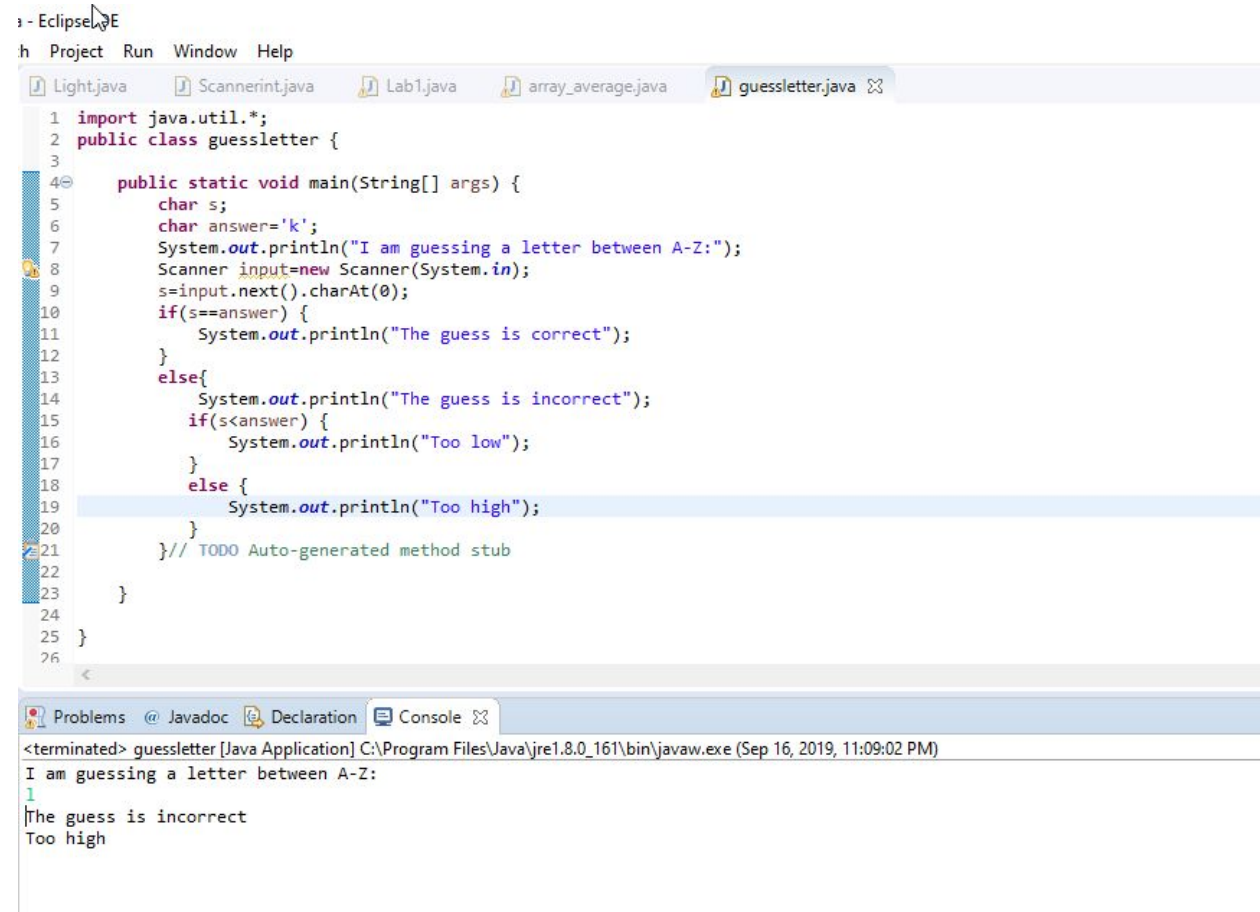
Light.java Scannerint.java Lab1.java array_average.java guessletter.java ✕

1 import java.util.*;
2 public class guessletter {
3
4     public static void main(String[] args) {
5         char s;
6         char answer='k';
7         System.out.println("I am guessing a letter between A-Z:");
8         Scanner input=new Scanner(System.in);
9         s=input.next().charAt(0);
10        if(s==answer) {
11            System.out.println("The guess is correct");
12        }
13        else{
14            System.out.println("The guess is incorrect");
15        }// TODO Auto-generated method stub
16
17    }
18
19 }
20

Problems @ Javadoc Declaration Console ✕

<terminated> guessletter [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 16, 2019, 11:02:05 PM)
I am guessing a letter between A-Z:
k
The guess is correct
```

Example



```
1 import java.util.*;
2 public class guessletter {
3
4     public static void main(String[] args) {
5         char s;
6         char answer='k';
7         System.out.println("I am guessing a letter between A-Z:");
8         Scanner input=new Scanner(System.in);
9         s=input.next().charAt(0);
10        if(s==answer) {
11            System.out.println("The guess is correct");
12        }
13        else{
14            System.out.println("The guess is incorrect");
15            if(s<answer) {
16                System.out.println("Too low");
17            }
18            else {
19                System.out.println("Too high");
20            }
21        } // TODO Auto-generated method stub
22    }
23 }
24
25 }
26
```

<terminated> guessletter [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 16, 2019, 11:09:02 PM)

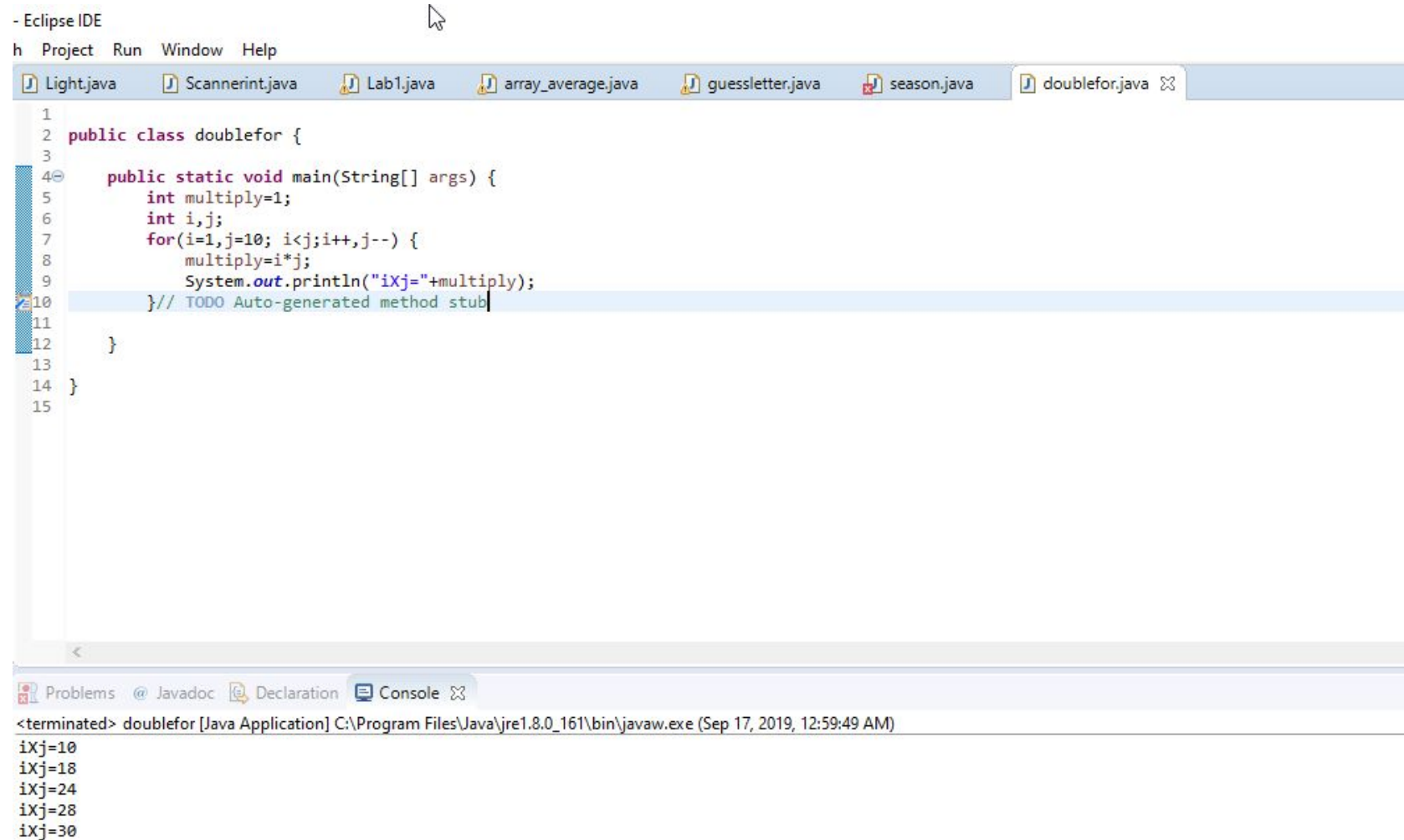
I am guessing a letter between A-Z:

1

The guess is incorrect

Too high

Example



The screenshot displays the Eclipse IDE interface. The top menu bar includes 'File', 'Edit', 'Project', 'Run', 'Window', and 'Help'. Below the menu is a tabbed editor showing several Java files: 'Light.java', 'Scannerint.java', 'Lab1.java', 'array_average.java', 'guessletter.java', 'season.java', and 'doublefor.java'. The 'doublefor.java' file is active, showing the following code:

```
1
2 public class doublefor {
3
4     public static void main(String[] args) {
5         int multiply=1;
6         int i,j;
7         for(i=1,j=10; i<j;i++,j--) {
8             multiply=i*j;
9             System.out.println("ixj="+multiply);
10        }// TODO Auto-generated method stub
11
12    }
13
14 }
15
```

At the bottom of the IDE is the 'Console' view, which shows the output of the program:

```
<terminated> doublefor [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 17, 2019, 12:59:49 AM)
ixj=10
ixj=18
ixj=24
ixj=28
ixj=30
```

for each

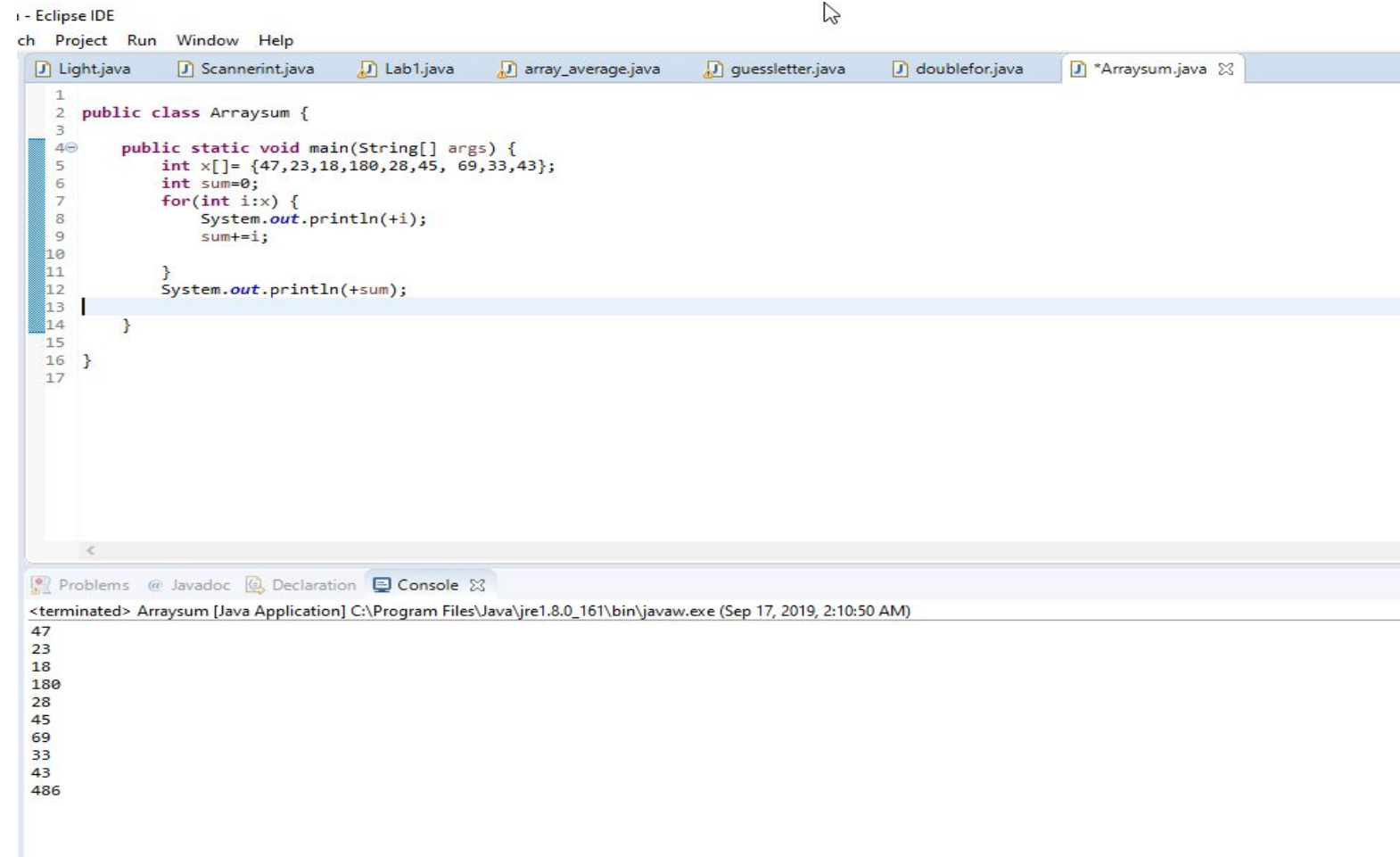
- The for-each style for automates the preceding loop. Specifically, it eliminates the need to establish a loop counter, specify a starting and ending value, and manually index the array. Instead, it automatically cycles through the entire array, obtaining one element at a time, in sequence, from beginning to end. For example, here is the preceding fragment rewritten using a for-each version of the for:

```
int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

```
int sum = 0;
```

```
for(int x: nums) sum += x;
```

for each example



The screenshot shows the Eclipse IDE interface. The top menu bar includes 'File', 'Edit', 'Project', 'Run', 'Window', and 'Help'. The project explorer on the left lists several files: 'Light.java', 'Scannerint.java', 'Lab1.java', 'array_average.java', 'guessletter.java', 'doublefor.java', and '*Arraysum.java'. The main editor window displays the code for 'Arraysum.java'.

```
1 public class Arraysum {
2
3
4     public static void main(String[] args) {
5         int x[] = {47, 23, 18, 180, 28, 45, 69, 33, 43};
6         int sum = 0;
7         for(int i: x) {
8             System.out.println(i);
9             sum += i;
10        }
11        System.out.println(+sum);
12    }
13 }
14
15
16
17
```

Below the editor, the 'Console' tab is active, showing the output of the program:

```
<terminated> Arraysum [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 17, 2019, 2:10:50 AM)
47
23
18
180
28
45
69
33
43
486
```

for each example

nal.java - Eclipse IDE

File Project Run Window Help

Light.java Scannerint.java Lab1.java array_average.java guessletter.java doublefor.java Arraysun.java

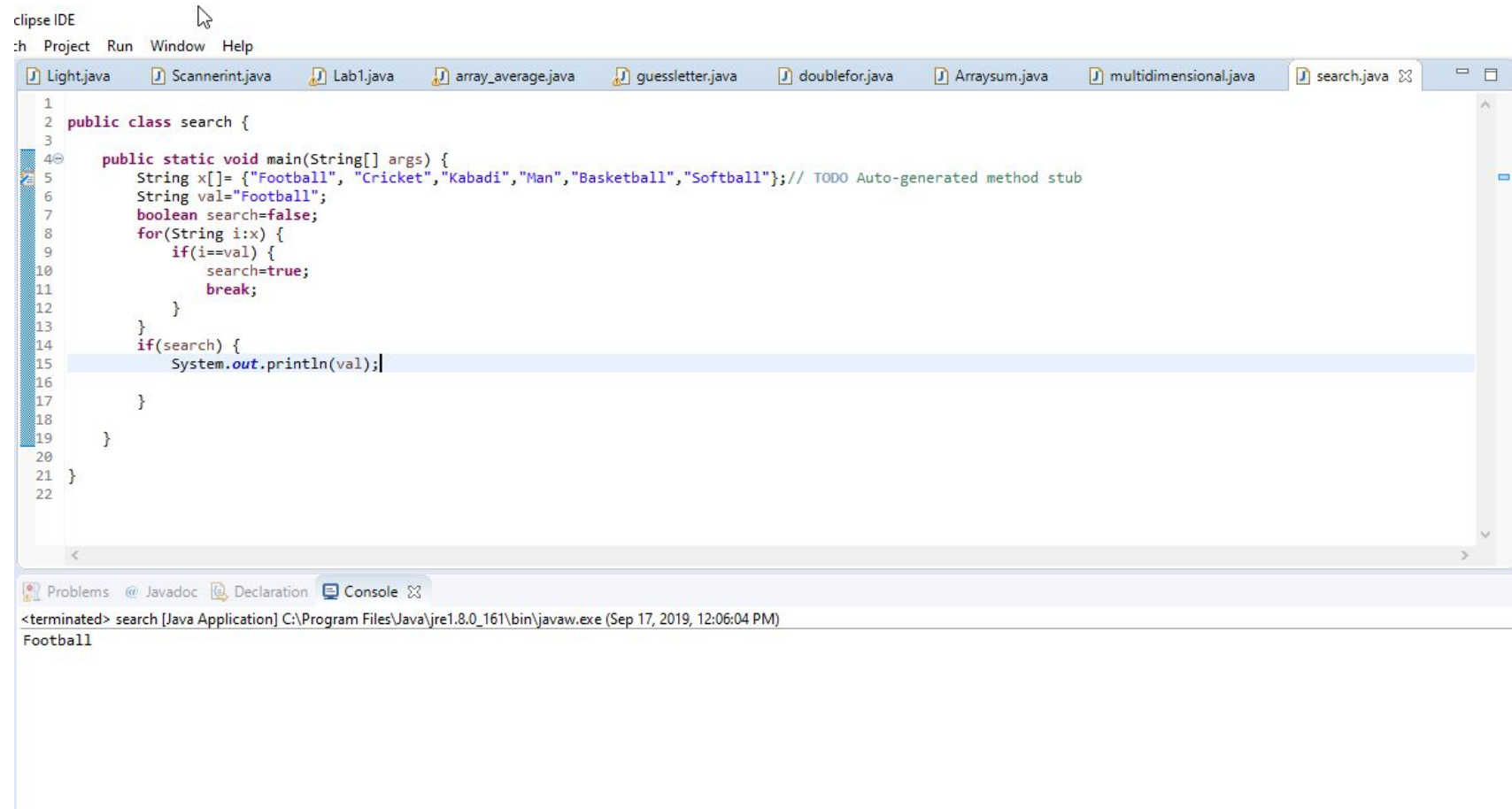
```
1
2 public class multidimensional {
3 public static void main(String[] args) {
4     int arr[][]= {{45,67,78},{23,89,32}};
5     int sum=0;
6     for(int i[:arr) {
7         for(int y:i) {
8             sum+=y;
9         }
10    }
11    System.out.println(+sum);|
12
13
14
15 }
16 }
17
```

Problems Javadoc Declaration Console

<terminated> multidimensional [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 17, 2019, 11:57:12 AM)

334

For each example



The screenshot shows the Eclipse IDE interface. The top menu bar includes 'File', 'Edit', 'Project', 'Run', 'Window', and 'Help'. The top toolbar contains icons for 'File', 'Run', 'Debug', 'Window', and 'Help'. The main editor window displays the code for 'search.java'. The code is as follows:

```
1 public class search {
2
3
4     public static void main(String[] args) {
5         String x[] = {"Football", "Cricket", "Kabadi", "Man", "Basketball", "Softball"}; // TODO Auto-generated method stub
6         String val = "Football";
7         boolean search = false;
8         for (String i : x) {
9             if (i == val) {
10                 search = true;
11                 break;
12             }
13         }
14         if (search) {
15             System.out.println(val);
16         }
17     }
18 }
19
20
21
22
```

The bottom of the IDE shows the 'Console' tab, which displays the output of the program:

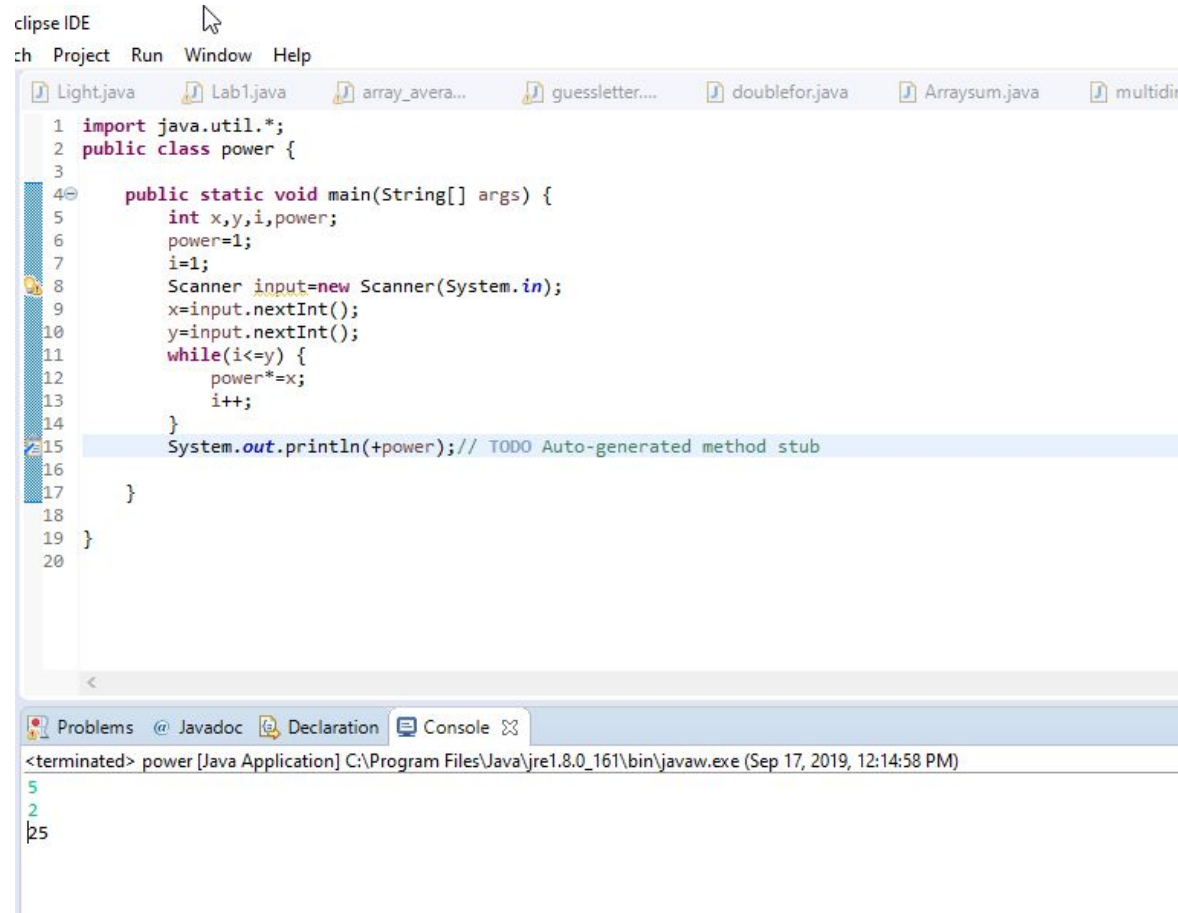
```
<terminated> search [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 17, 2019, 12:06:04 PM)
Football
```

while loop

- The while loop is Java's most fundamental loop statement. It repeats a statement or block while its controlling expression is true. Here is its general form:

```
while(condition) {  
    // body of loop  
}
```

while loop example



The screenshot shows the Eclipse IDE interface. The top menu bar includes 'File', 'Edit', 'Project', 'Run', 'Window', and 'Help'. The project explorer on the left lists several Java files: 'Light.java', 'Lab1.java', 'array_avera...', 'guessletter....', 'doublefor.java', 'Arraysum.java', and 'multidir...'. The main editor window displays the code for 'Lab1.java'. The code is as follows:

```
1 import java.util.*;
2 public class power {
3
4     public static void main(String[] args) {
5         int x,y,i,power;
6         power=1;
7         i=1;
8         Scanner input=new Scanner(System.in);
9         x=input.nextInt();
10        y=input.nextInt();
11        while(i<=y) {
12            power*=x;
13            i++;
14        }
15        System.out.println(+power);// TODO Auto-generated method stub
16
17    }
18
19 }
20
```

The bottom of the IDE shows the 'Console' tab with the following output:

```
<terminated> power [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 17, 2019, 12:14:58 PM)
5
2
25
```

Switch

- The switch statement is Java's multiway branch statement. It provides an easy way to dispatch execution to different parts of your code based on the value of an expression. As such, it often provides a better alternative than a large series of if-else-if statements. Here is the general form of a switch statement:

```
switch (expression) {  
    case value1:  
        // statement sequence  
        break;  
}
```

Switch example

Project Run Window Help

```
switch_1.java
1 import java.util.*;
2 public class switch_1 {
3
4     public static void main(String[] args) {
5         int ch;
6         Scanner input=new Scanner(System.in);
7         ch= input.nextInt();
8         switch(ch) {
9             case 1:
10                System.out.println("January");
11                break;
12             case 2:
13                System.out.println("February");
14                break;
15             case 3:
16                System.out.println("March");
17                break;
18             case 4:
19                System.out.println("April");
20                break;
21             case 5:
22                System.out.println("May");
23                break;
24             case 6:
25                System.out.println("June");
26                break;
27             case 7:
28                System.out.println("July");
29                break;
30             case 8:
31                System.out.println("August");
32                break;
33             case 9:
```

Console

<terminated> switch_1 [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Sep 17, 2019, 3:26:45 PM)

4
April

Switch example

```
33     case 9:
34         System.out.println("September");
35         break;
36     case 10:
37         System.out.println("October");
38         break;
39     case 11:
40         System.out.println("November");
41         break;
42     case 12:
43         System.out.println("December");
44         break;
45     default:
46         System.out.println("Invalid");
47         break;
48     }
49 }
50 }
51
```

Console

<terminated> switch_1 [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Sep 17, 2019, 3:26:45)

4
April

Switch example

```
switch_1.java *switchseason.java
1 import java.util.Scanner;
2 public class switchseason {
3
4     public static void main(String[] args) {
5         String season;
6         Scanner input=new Scanner(System.in);
7         season=input.next();
8         switch(season) {
9             case "December":
10             case "January":
11             case "February":
12                 System.out.println("Winter");
13                 break;
14             case "March":
15             case "April":
16             case "May":
17                 System.out.println("Autumn");
18                 break;
19             case "June":
20             case "July":
21             case "August":
22                 System.out.println("Summer");
23                 break;
24             case "September":
25             case "Octobor":
26             case "November":
27                 System.out.println("Spring");
28                 break;
29         }
```

Switch example

```
29         default:
30             System.out.println("Invalid season");
31             break;
32     }
33     // TODO Auto-generated method stub
34
35 }
36
37 }
38
```

Console

<terminated> switchseason [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Sep 1

April

Autumn

break

- In Java, the break statement has three uses.
- First, as you have seen, it terminates a statement sequence in a switch statement.
- Second, it can be used to exit a loop.
- Third, it can be used as a “civilized” form of goto

break example

eclipse-workspace - ICE107/src/breakexample.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

switch_1.java switchseason.java breakexample.java

```
1 public class breakexample {
2
3
4     public static void main(String[] args) {
5         int i;
6         for(i=1; i<=10; i++) {
7             if(i==9) {
8                 break;
9             }
10            System.out.println(i);
11        } // TODO Auto-generated method stub
12
13    }
14
15 }
16
```

Console

<terminated> breakexample [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Sep 17, 2019, 4:07:25 PM)

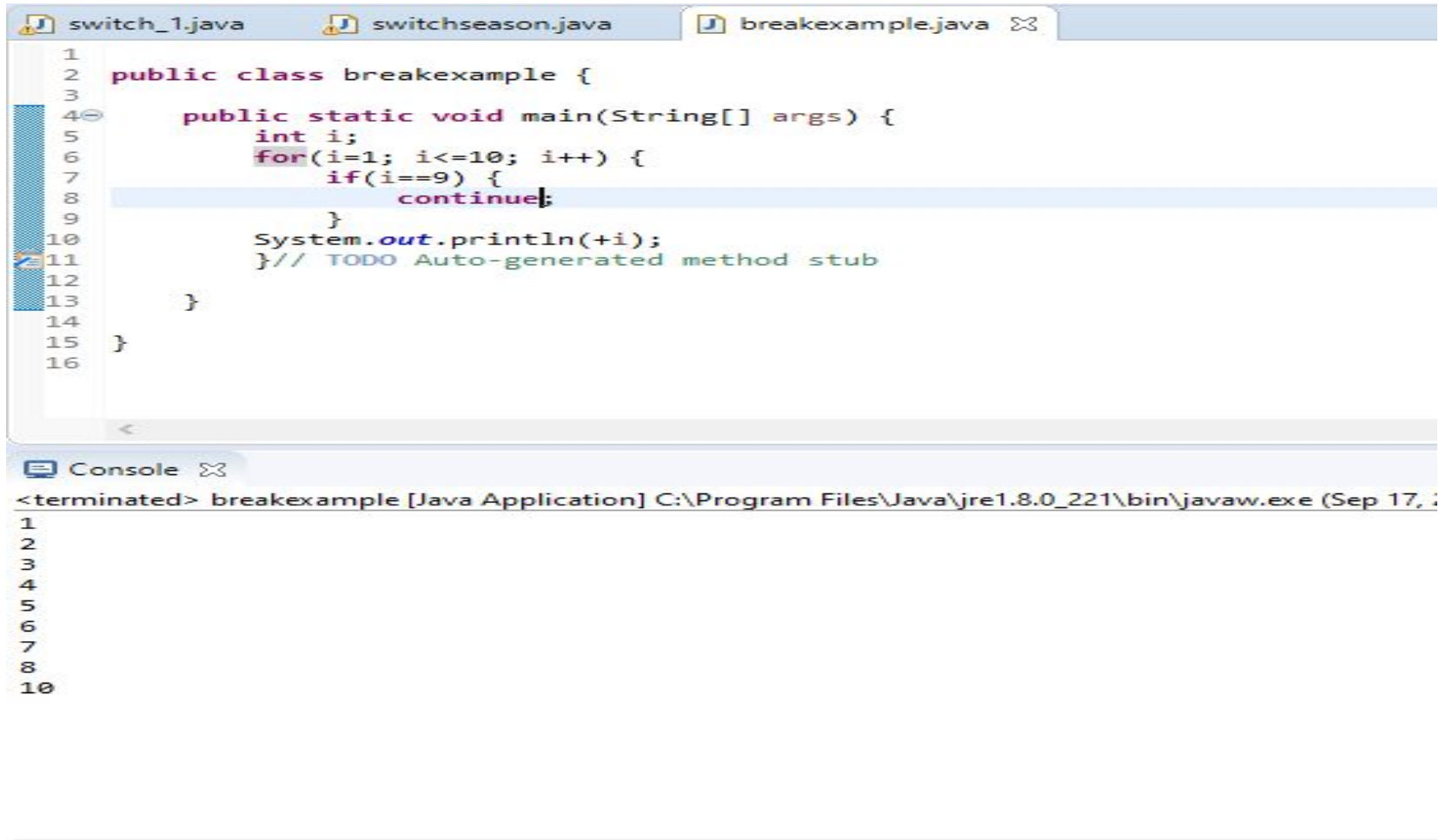
```
1
2
3
4
5
6
7
8
```

Writable Smart Insert 10:32

Continue

- In while and do-while loops, a continue statement causes control to be transferred directly to the conditional expression that controls the loop. In a for loop, control goes first to the iteration portion of the for statement and then to the conditional expression. For all three loops, any intermediate code is bypassed.

Continue example



```
1 public class breakexample {
2
3
4     public static void main(String[] args) {
5         int i;
6         for(i=1; i<=10; i++) {
7             if(i==9) {
8                 continue;
9             }
10            System.out.println(+i);
11        } // TODO Auto-generated method stub
12
13    }
14
15 }
16
```

<terminated> breakexample [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Sep 17, :)

```
1
2
3
4
5
6
7
8
9
10
```

do while

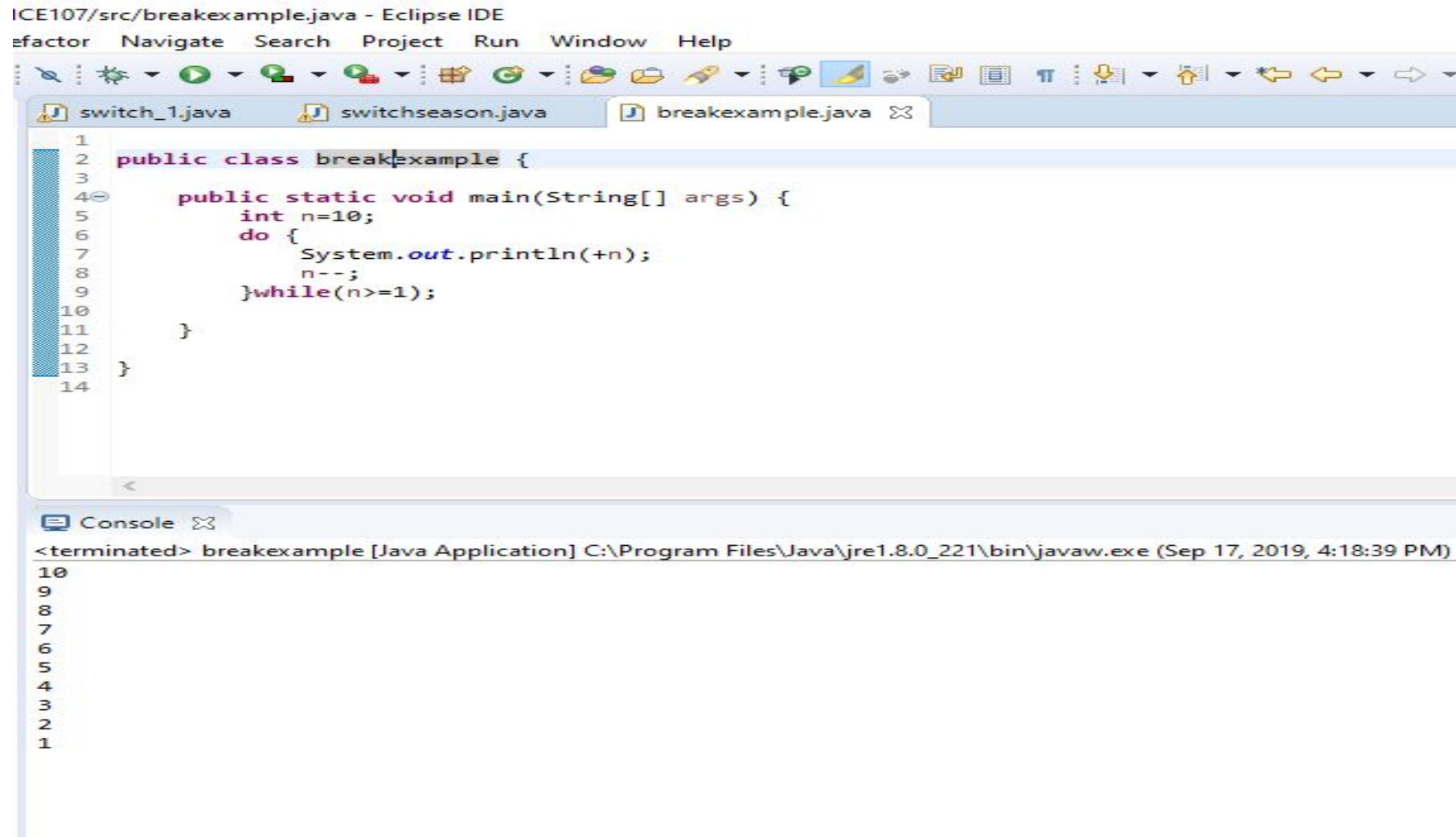
- the do-while. The do-while loop always executes its body at least once, because its conditional expression is at the bottom of the loop. Its general form is

```
do {
```

```
// body of loop
```

```
} while (condition);
```

do...while loop example



The screenshot shows the Eclipse IDE interface. The top menu bar includes 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. Below the menu is a toolbar with various icons. The editor window displays the file 'breakexample.java' with the following code:

```
1
2 public class breakexample {
3
4     public static void main(String[] args) {
5         int n=10;
6         do {
7             System.out.println(+n);
8             n--;
9         }while(n>=1);
10
11     }
12
13 }
14
```

The bottom panel shows the 'Console' output, which displays the numbers 10 through 1 in descending order, indicating the successful execution of the do...while loop.

```
<terminated> breakexample [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Sep 17, 2019, 4:18:39 PM)
10
9
8
7
6
5
4
3
2
1
```