CSE 1325

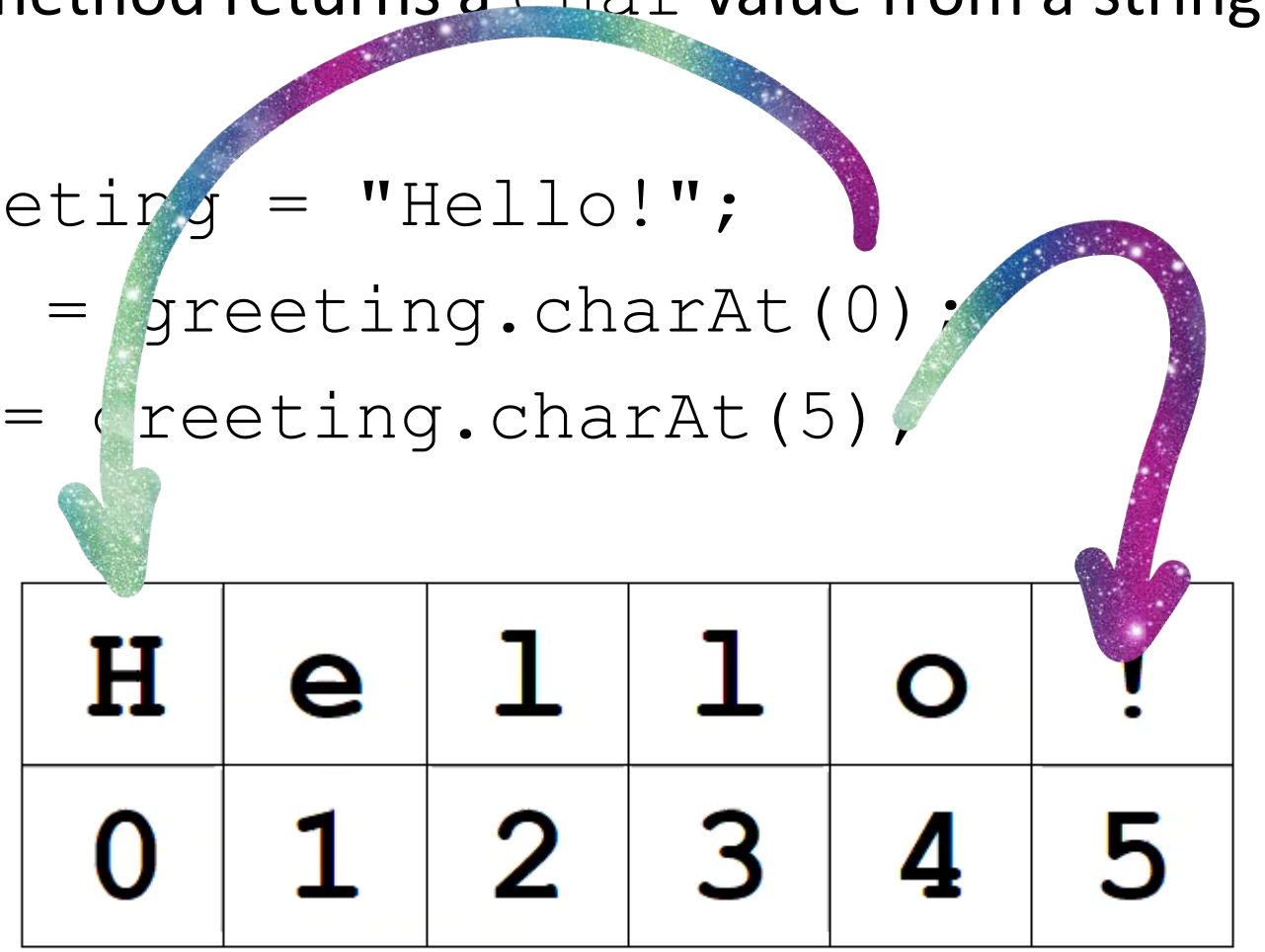
Week of 09/05/2022

Instructor : Donna French

Strings and Characters

The `charAt` method returns a `char` value from a string.

```
String greeting = "Hello!";
char first = greeting.charAt(0);
char last = greeting.charAt(5);
```



Source History

```
 * @author Donna
 */
public class Strings
{

    public static void main(String[] args)
    {
        String greeting = "Hello!";

    }

}

```

strings.Strings > main >

Output - Strings (run)

```
public static void main(String[] args)
{
    String greeting = "Hello!";
    char first = greeting.charAt(0);
    char last = greeting.charAt(5);

    System.out.print(first);
    System.out.print(last);
}
```

Substrings

We can extract a piece of string

Given string

"Hello!"

a piece of it would be

"ello"

That piece of a string is called a "substring"

Substrings

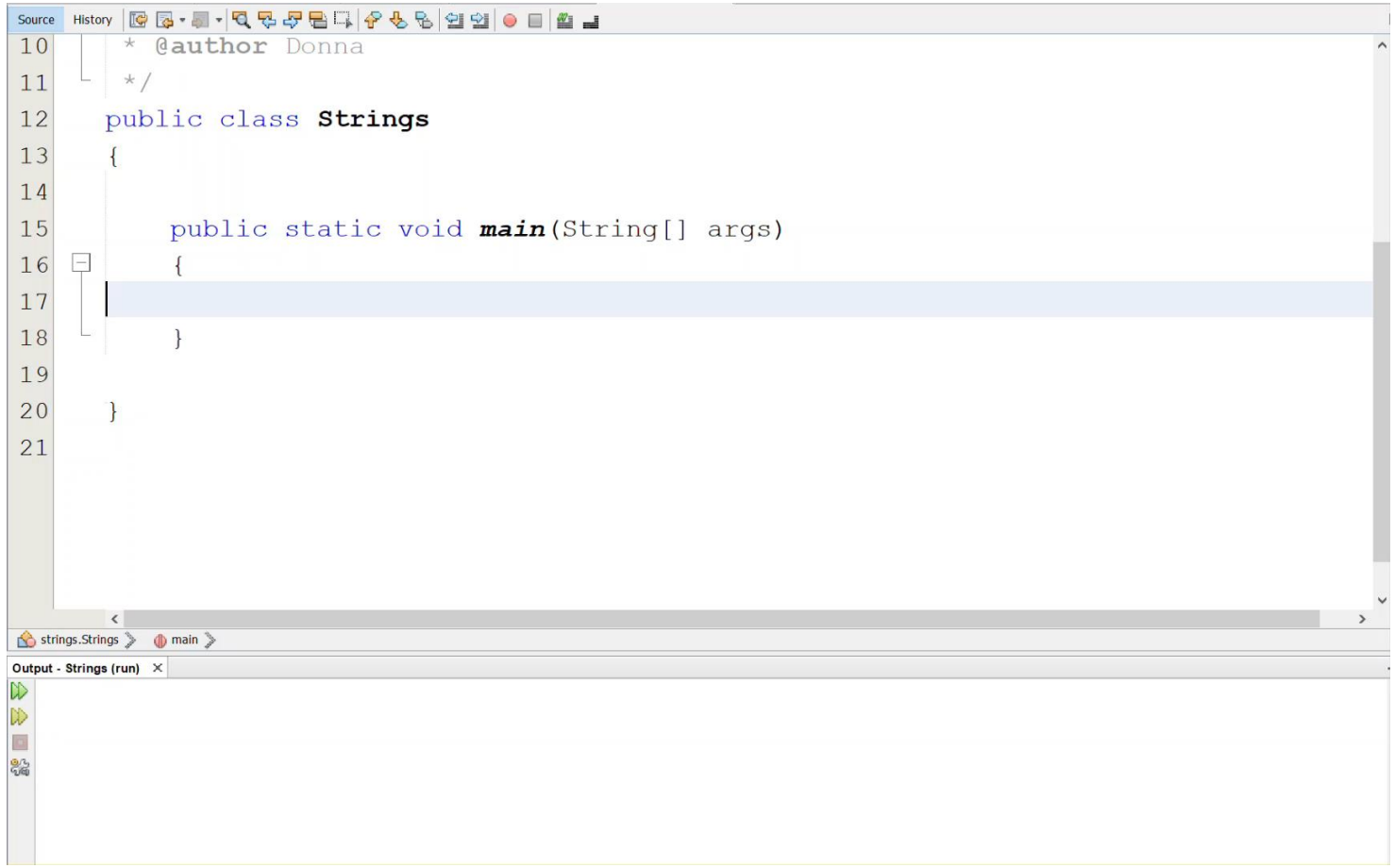
The `substring` method returns a piece/substring of a given string.

```
Yarn.substring(start, pastEnd);
```

`start` is the first character of the substring
`pastEnd` is the position one past the desired end

H	e	l	l	o		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11

H	e	l	l	o		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11



```
public static void main(String[] args)
{
    String greeting = "Hello World!";
    String subgreeting = greeting.substring(6, 12);
    System.out.println(subgreeting);
}
```

Substrings

If the `substring` method is used with only one parameter

```
Yarn.substring(start);
```

then that parameter is treated as the start and everything to the end of the string is copied.

```
String subgreeting = greeting.substring(6);
```

`subgreeting` would be set to

```
World!
```

H	e	l	l	o		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11

H	o	w		a	r	e		y	o	u	?
0	1	2	3	4	5	6	7	8	9	10	11

Code to retrieve "How"

```
greeting.substring(0, 3)
```

What does

```
greeting.substring(4, 8)
```

retrieve?

```
are
```

Code to retrieve " you?"

```
greeting.substring(7, 12)
```

What does

```
greeting.substring(5, 10)
```

retrieve?

```
"re yo"
```

Code to retrieve "How are you?"

```
greeting.substring(0)
greeting.substring(0, 12)
```

What does

```
greeting.substring(0, 11)
```

retrieve?

```
How are you
```

Code to retrieve "w a"

```
greeting.substring(2, 5)
```

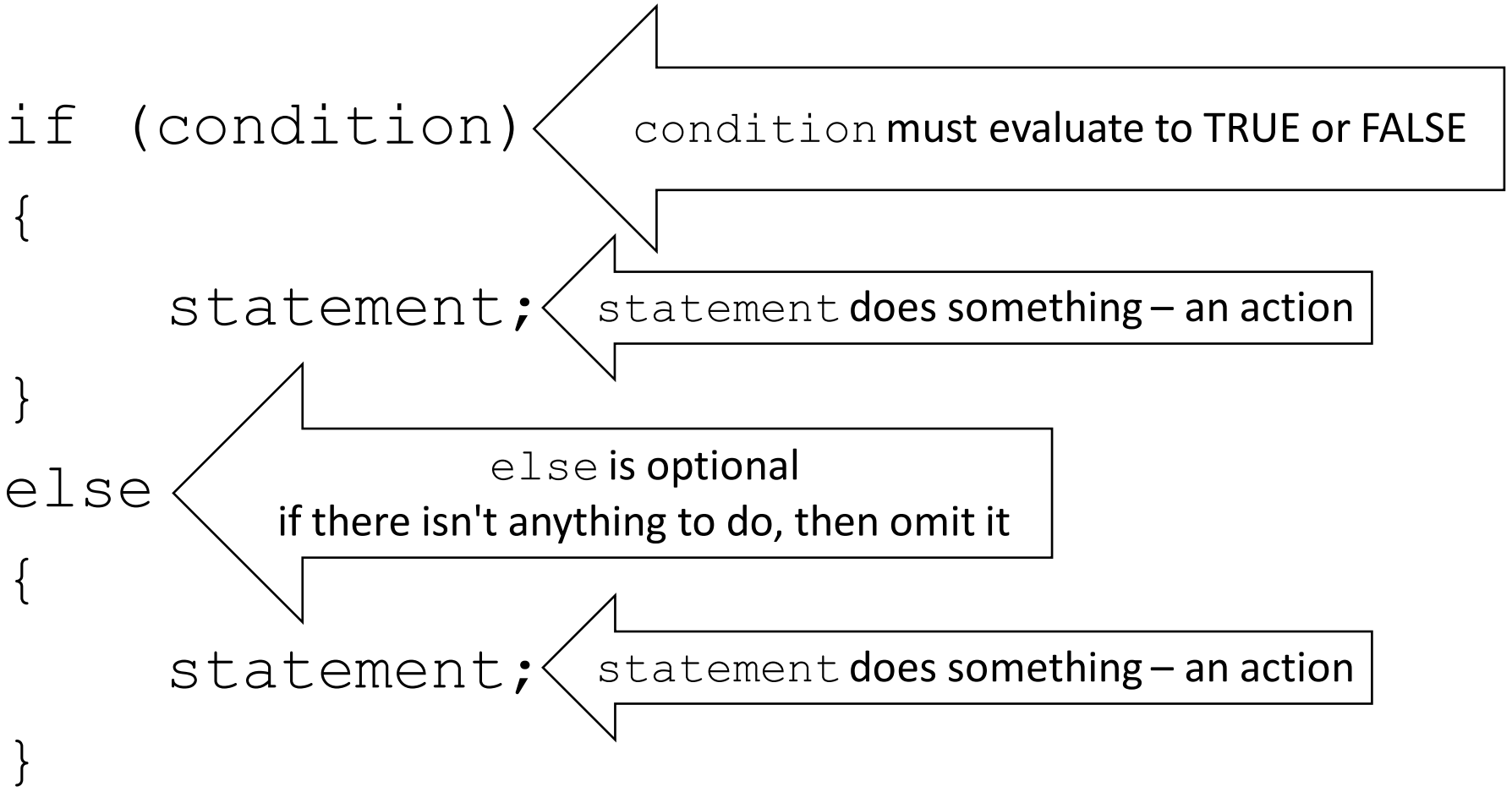
What does

```
greeting.substring(8)
```

retrieve?

```
"you?"
```

The `if` Statement



```
public static void main(String[] args)
{
    int x = 2;

    if (x)
    {
        System.out.println("Yes");
    }
    else
    {
        System.out.println("No");
    }
}
```

In Java, the if condition must evaluate to a Boolean value – true or false
Fix this by changing `(x)` to `(x == 2)`

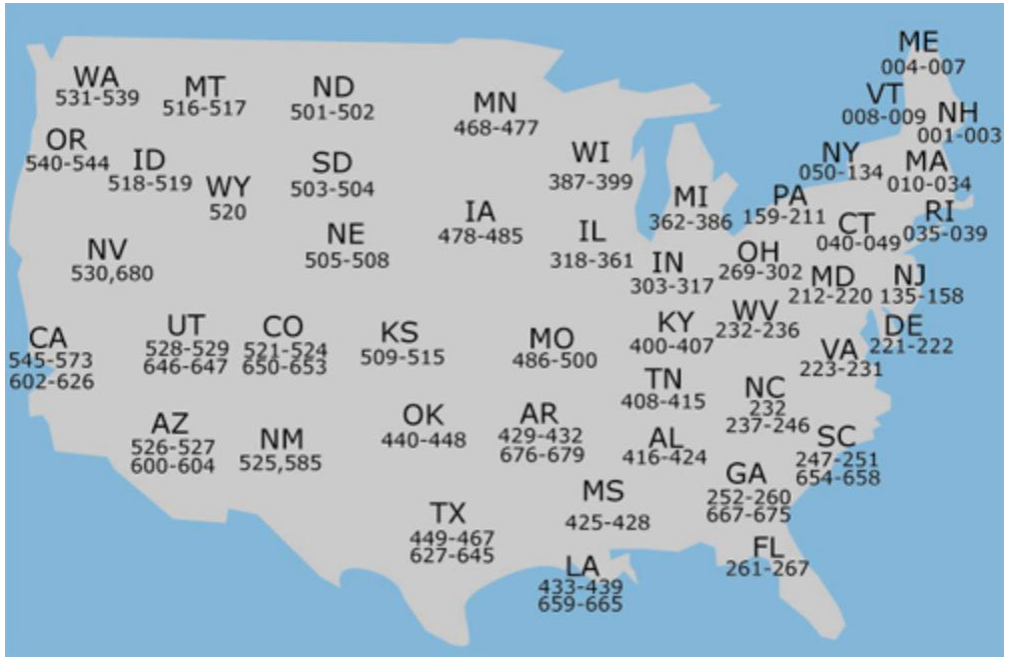
```
C:\Users\frenc\Documents\NetBeansProjects\IfJava\src\ifjava\IfJava.java:1
2: error: incompatible types: int cannot be converted to boolean
        if (x)
1 error
BUILD FAILED (total time: 1 second)
```

Making Decisions

A Social Security Number consists of nine digits - three *area numbers*, two *group numbers*, and four *serial numbers*.

The area number is code for the region of the address appearing on SSN application form.

Write a program to determine which area any given SSN is from.



Algorithm

Write a program to determine which area any given SSN is from.

Step 1	Prompt for SSN
Step 2	Extract area (first 3 characters) from SSN
Step 3	If area is xxx, then that SSN is from state YY (use provided map to fill in area-state associations)

Step 1 Prompt for SSN

```
Scanner in = new Scanner(System.in);

System.out.print("Please enter your SSN ");
String SSN = in.next();
```

Step 2 Extract area (first 3 characters) from SSN

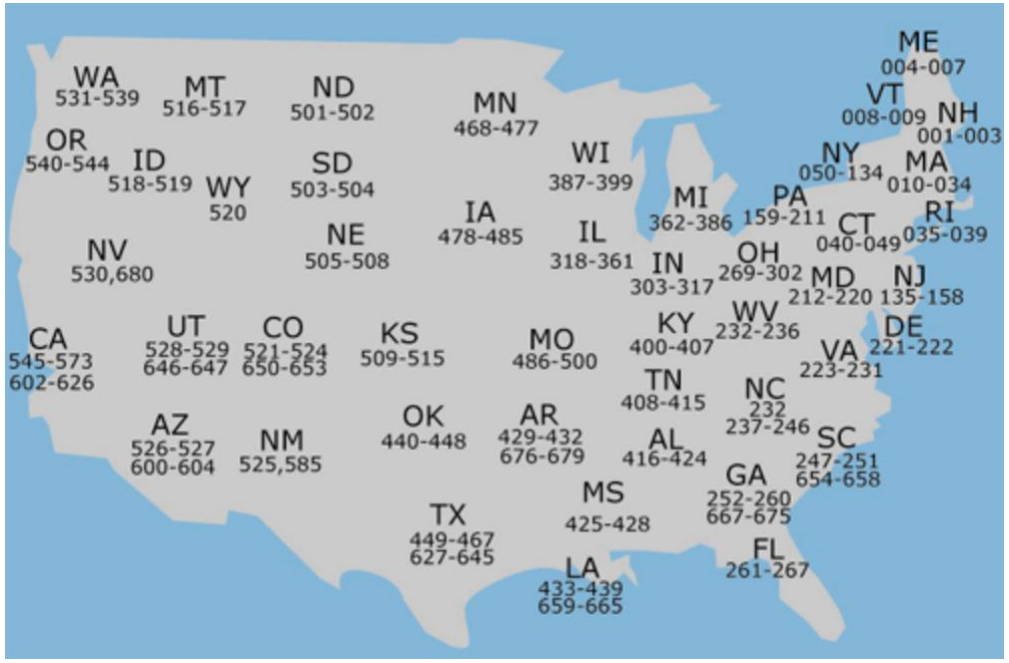
```
String areaNumber = SSN.substring(    );
```

1	2	3	1	2	1	2	3	4
0	1	2	3	4	5	6	7	8

1	2	3	-	1	2	-	1	2	3	4
0	1	2	3	4	5	6	7	9	10	11

Step 3 If area is xxx, then that SSN is from state YY (use provided map to fill in area-state associations)

```
if areaNumber is "501", then this SSN was assigned in ND

if areaNumber is "004", then this SSN was assigned in ME
```



Determining if two Strings are equal

It might be tempting to say

```
if (StringX == StringY)
```

but using an == between two `String`s can have a different meaning in Java depending on how the `String` variables were formed.

The == tests whether two strings are stored in the same location in memory – not if their contents are equivalent. This becomes a problem when you create a `String` using `new`.

Determining if two Strings are equal

So how do we safely and consistently determine if the contents of two strings are equal?

```
if (StringX equals StringY)
```

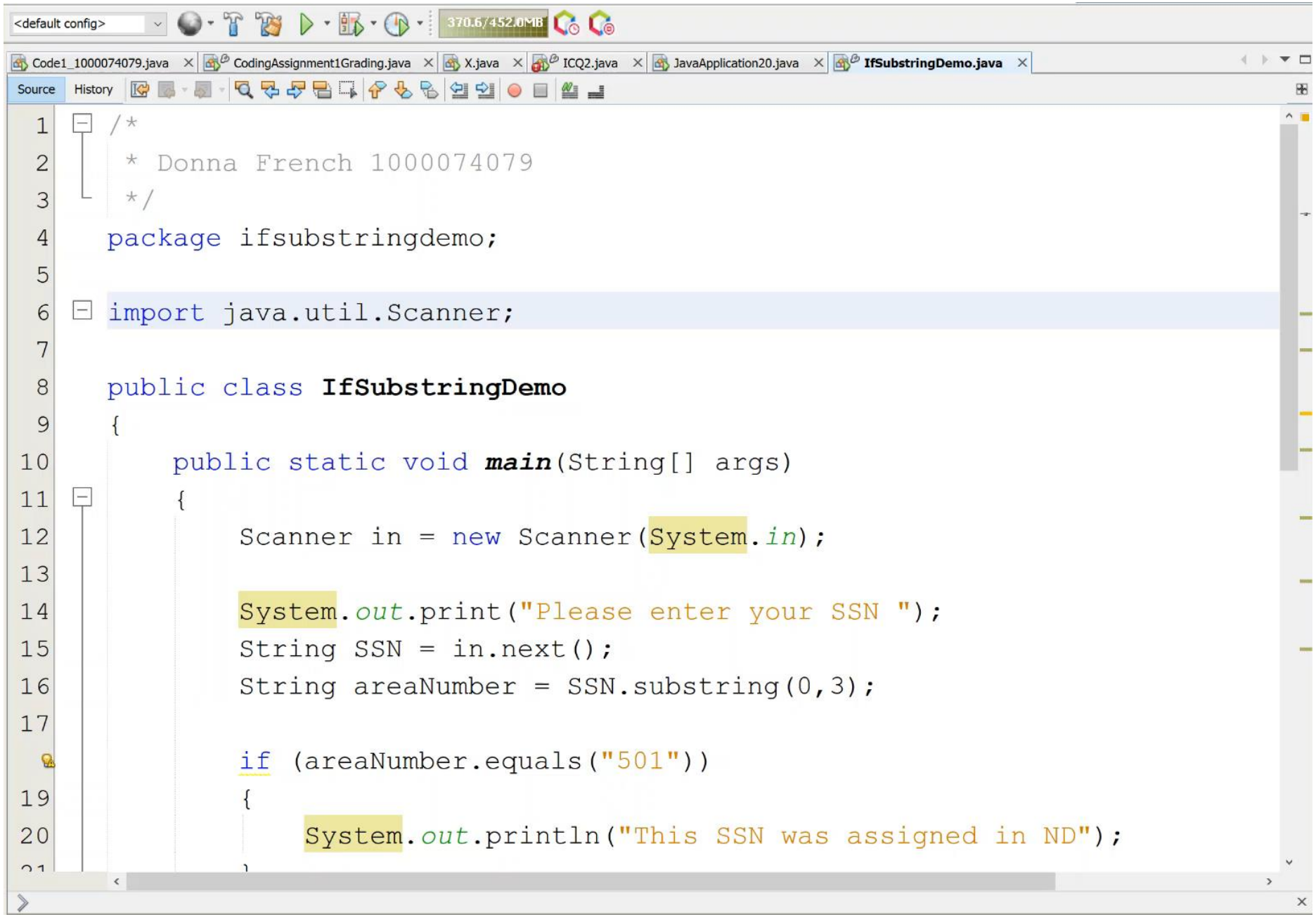
The `String` class has a method called `equals`.

We ask `StringX` to check if `StringY` is equal. We pass `StringY` to `StringX`'s `equals` method.

```
if (StringX.equals(StringY))
```

```
if (StringY.equals(StringX))
```

```
if (areaNumber.equals("501"))
{
      System.out.println("This SSN was assigned in ND");
}
else if (areaNumber.equals("004"))
{
      System.out.println("This SSN was assigned in ME");
}
else
{
      System.out.println("Unknown area number!");
}
```


<default config>
370.6/452.0MB
Code1_1000074079.java
CodingAssignment1Grading.java
X.java
ICQ2.java
JavaApplication20.java
IfSubstringDemo.java
Source
History
/*
* Donna French 1000074079
*/
package ifsubstringdemo;
import java.util.Scanner;
public class IfSubstringDemo
{
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
System.out.print("Please enter your SSN ");
String SSN = in.next();
String areaNumber = SSN.substring(0,3);
if (areaNumber.equals("501"))
{
System.out.println("This SSN was assigned in ND");

```
Scanner in = new Scanner(System.in);

System.out.print("Please enter your SSN ");
String SSN = in.next();
String areaNumber = SSN.substring(0,3);

if (areaNumber.equals("501"))
{
    System.out.println("This SSN was assigned in ND");
}
else if (areaNumber.equals("004"))
{
    System.out.println("This SSN was assigned in ME");
}
else if (areaNumber.equals("449"))
{
    System.out.println("This SSN was assigned in TX");
}
else
{
    System.out.println("Unknown area number");
}
```

Dangling `else` Problem

The Java compiler always associates an else with the immediately preceding if unless told to do otherwise by the placement of braces.

Best Practice – use {} to clear indicate association

```
if (x > 5)
{
      if (y > 5)
            System.out.println("x and y are > 5");
}
else
      System.out.println("x is <= 5");
```

```
Scanner in = new Scanner(System.in);
int x, y;

System.out.print("Please enter a value for x ");
x = in.nextInt();
System.out.print("Please enter a value for y ");
y = in.nextInt();

if (x > 5)
      if (y > 5)
            System.out.println("x and y are > 5");
else
      System.out.println("x is <= 5");
```

```
Please enter a value for x 4
Please enter a value for y 4
BUILD SUCCESSFUL
```

```
Please enter a value for x 6
Please enter a value for y 4
x is <= 5
BUILD SUCCESSFUL
```

```
Please enter a value for x 6
Please enter a value for y 6
x and y are > 5
BUILD SUCCESSFUL
```

```
Please enter a value for x 4
Please enter a value for y 6
BUILD SUCCESSFUL
```

Source History

```
19
20              System.out.print("Do you have time for lunch? ");
21              String lunchTime = in.next();
22              int timeForLunch = 0;
23
24              if (lunchTime.charAt(0) == 'Y');
25                  timeForLunch = 1;   // You have time for lunch
26
27              if (timeForLunch == 1)
28                  System.out.println("Great - Let's go have pizza");
29          }
30      }
31  }
32
```

studentcode.StudentCode > main

Output

StudentCode (run) | StudentCode (run) #2 | EvenOdd (run) | StudentCode (run) #3

```
Updating property file: C:\Users\Donna\Desktop\UTA\CSE1310\Programs\StudentC
Compiling 1 source file to C:\Users\Donna\Desktop\UTA\CSE1310\Programs\Stude
compile:
run:
```

```
System.out.print("Do you have time for lunch? ");

String lunchTime = in.next();

int timeForLunch = 0;

if (lunchTime.charAt(0) == 'Y');

     timeForLunch = 1;    // You have time for lunch

if (timeForLunch == 1)

     System.out.println("Great - Let's go have pizza");
```

Placing a semicolon after the `if` condition creates a logic error that affects how your program **behaves**.

It does NOT create a compiler error.

Boolean Variables and Operators

Java has Boolean data type

A `boolean` can hold either

`true`

or

`false`

```
boolean failed = true;

boolean frog = false;

boolean toad = true;

boolean x = 0;

error: incompatible types: int
cannot be converted to boolean
        boolean x = 0;
```

Boolean Variables and Operators

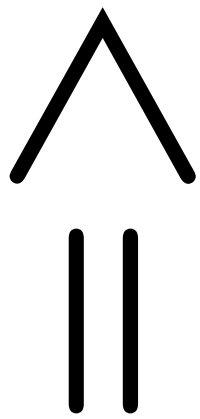
```
public static void main(String[] args)
{
    boolean x = true;

    if (x)
    {
        System.out.println("Yes");
    }
    else
    {
        System.out.println("No");
    }
}
```

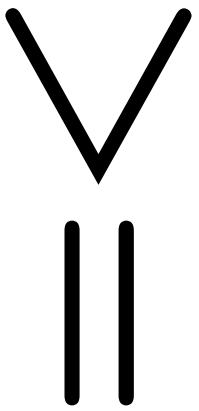
x is a boolean; therefore, the if condition evaluates to a true or false

Relational Operators

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equivalent to
!=	Not equivalent to

The actual value assigned to an expression formed with a relational operator is `1` if the relation is true and `0` if it is false.





Relational Operators

= vs ==

= in coding means assignment

```
x = 1
```

read as x is assigned the value of 1

== in coding means equivalent test

```
x == 1
```

read as the value stored in x equivalent to 1?

Relational and Logical Operators Combined

Is it valid to use

```
if (x < y < 3)
```

```
error: bad operand types
for binary operator '<'
        if (x < y < z)
  first type:  boolean
  second type: int
```

No

```
if (x < y && y < 3)
```

Random Number Generator

- The Java library has a random number generator which produces numbers that appear to be completely random.

- The numbers are not completely random. They are drawn from sequences of numbers that don't repeat for a long time.
  - These types of "random" numbers are sometimes called pseudorandom numbers.

- The provided random number generator works well enough for most purposes requiring a random number.

Random Number Generator

`Math.random()`

Produces a random floating point number that is >= 0 and < 1.

Can be used to set a `double` to a value or can be used in a print statement.

Source History

```
1  package studentcode;
2  import java.util.Scanner;
3  public class StudentCode
4  {
5      public static void main(String[] args)
6      {
7          Scanner in = new Scanner(System.in);
8
9
10     }
11 }
12
```

Random Number Generator

Sometimes we need to generate random integers instead of floating point values.

We can transform the value returned by `Math.random()` into an integer value between two numbers.

Source History

```
5      public static void main(String[] args)
6      {
7          Scanner in = new Scanner(System.in);
8
9          double MyRandomFloat = Math.random();
10
11         System.out.printf("Printing a random floating point number... %f%n",
12             MyRandomFloat);
13
14
15     }
16 }
17
```

studentcode.StudentCode > main >

Output - StudentCode (run) #5

Random Number Generator

Multiple the floating point random number by the max value of the integer range you want to use.

If we want a random integer between 1 and 6, then

```
MyRandomFloat * 6
```

Add 1

Cast to an `int` to remove the decimal portion

Random Number Generator

```
double MyRandomFloat = Math.random();

System.out.printf("Printing a random floating point number... %f%n",
    MyRandomFloat);

int MyRandomInt = (int)(MyRandomFloat * 6) + 1;

System.out.printf("Printing random integer... %d%n", MyRandomInt);
```

Random Number Generator

Another method of generating an integer random number is to use the `Random` class (like we use the `Scanner` class).

Step 1 - import `Random`

```
import java.util.Scanner;
import java.util.Random;
```

Step 2 – Create an instance of `Random`

```
Scanner in = new Scanner(System.in);
Random rn = new Random();
```

Random Number Generator

Step 3 – call method `nextInt()` from `Random` to get a random number between 0 and the value you pass into `nextInt()`.

```
int MyRandomInt = rn.nextInt(20);
```

This will return random integer values between 0 and 19. You will NOT get the number you passed into `nextInt()`. If you want all random numbers between 0 and 20, then use 21 with `nextInt()`. If you want random numbers between 1 and 20, then use 20 with `nextInt()` and add 1 to the result.

```
int MyRandomInt = rn.nextInt(20) + 1;
```

Source History

```java
package studentcode;
import java.util.Scanner;
public class StudentCode
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);


    }
}

```

Output - StudentCode (run) #5

Source History

```java
package studentcode;
import java.util.Scanner;
import java.util.Random;
public class StudentCode
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        Random rn = new Random();

        int MyRandomInt = rn.nextInt(20);
        System.out.printf("Random Integer = %d%n", MyRandomInt);
    }
}
```

studentcode.StudentCode > main > MyRandomInt

Output - StudentCode (run) #5

`while` loop

Loop statements repeatedly execute instructions until a goal has been reached.

while a condition is true

    execute one or more statements

```
while (condition)
{
    statements
}
```

`while` loop

The condition of a while loop is about how long will the loop keep going - not about the goal.

We want to stop looping when counter is 5 but our condition is to keep looping until counter reaches 5.

```
int counter = 0;

while (counter < 5)
{
   counter++;
   System.out.printf("Counter = %d\n", counter);
}

System.out.printf("Final value of counter = %d", counter);
```

`while` loop

Infinite loops

We need to be careful to not get our loop into a situation where the loop cannot quit.

If the loop condition is never satisfied, the loop will continue to execute.

```java
public class StudentCode
{
    public static void main(String[] args)
    {
        int counter = 0;

        while (counter < 5)
        {
            counter++;
            System.out.printf("Counter = %d\n", counter);
        }
        System.out.printf("Final value of counter = %d", counter);
```

studentcode.StudentCode > main > while (counter < 5)

Output - StudentCode (run)

12:22 INS

`hasNextInt`

- method from the `Scanner` class
- returns TRUE if it is possible to read an item of type `int`
- the next item on the input buffer is tested but not used

```
Scanner in = new Scanner(System.in);

int aValue = 0;

System.out.print("Enter a value ");

while (in.hasNextInt())
{
    aValue = in.nextInt();
    System.out.printf("You entered %d\n", aValue);
    System.out.print("Enter a value ");
}
```

```
7       {
8           Scanner in = new Scanner(System.in);
9
10          int aValue = 0;
11
12          System.out.print("Enter a value ");
13
14          while (in.hasNextInt())
15          {
16              aValue = in.nextInt();
17              System.out.printf("You entered %d\n", aValue);
18              System.out.print("Enter a value ");
19          }
20
```

studentcode.StudentCode > main > while (in.hasNextInt())

Output

StudentCode (run) | StudentCode (run) #2

StudentCode (run) running... 19:10 INS

`hasNextDouble`

- method from the `Scanner` class
- returns TRUE if it is possible to read an item of type `double`
- the next item on the input buffer is tested but not used

`while` loop

Algorithm

Ask the user for a list of grades like 82.4 and 99.5. Enter grades until a non double is entered. Find the average.

Print a prompt

while (input is a double)

read input and sum it

count how many grades have been entered

prompt for next double

Print the average which is the sum of the grades divided by the number of grades entered.

```
double gradeSum = 0;
int numberOfGradesEntered = 0;

System.out.print("Enter a grade ");

while (in.hasNextDouble())
{
    gradeSum += in.nextDouble();
    numberOfGradesEntered++;
    System.out.print("Enter a grade ");
}

System.out.printf("The average is %.2f\n",
    gradeSum/numberOfGradesEntered);
```

methods

A method is a sequence of instructions with a name.

We have already seen several methods.

`Math.pow` is a method.

`main` is a method

methods

You call a method in order to execute its instructions.

```
public static void main(String[] args)
{
      double result = Math.pow(2,3);

}
```

By using the expression `Math.pow(2,3)`,`main` calls the `Math.pow` method, asking it to compute 2^3.

The instructions of the `Math.pow` method execute and compute the result.

The `Math.pow` method returns its result back to `main` and the `main` method resumes execution.

methods

The inputs we pass to the method

```
Math.pow(2,3)
```

are called **arguments**. 2 is the first argument and 3 is the second argument.

The output of the function

```
double result = Math.pow(2,3);
```

is called the **return value**.

methods

Methods can receive multiple arguments but can only return one value.

It is also possible to have methods with no arguments.

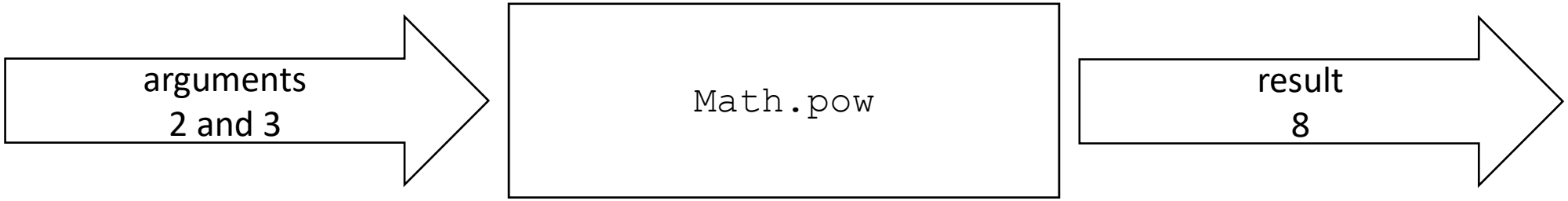
```
double MyRandomFloat = Math.random();
```

`Math.random()` does not take any arguments but it returns a value.

Print statements will be needed to print the return value.

methods

```
double result = Math.pow(2,3);
```



How does `Math.pow` work?

Does it multiple 2x2x2?

Does it use logarithms?


DON'T
KNOW
DON'T
CARE

methods

As a user of the method, you *don't need to know* how the method works.

You just need to know the *specification* of the method

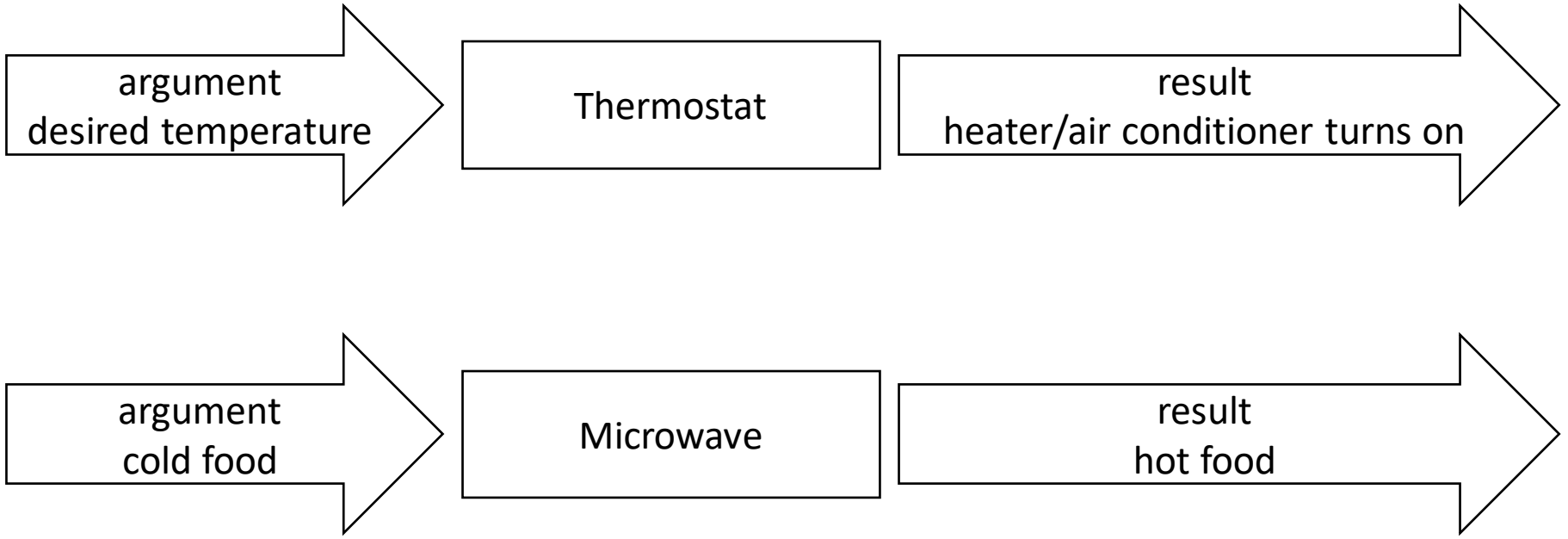
what arguments does it need?

what result does it give?

A term used to describe this is "black box".

methods

Lots of things in our lives are "black boxes".



methods

When we create our own methods, you should strive to make them appear as black boxes to others.

Anyone using your method should only need to know

what goes in

what comes out

Given certain inputs, what is the expected output?

methods

By default, parameters in Java methods are passed by value.

Pass by value means a copy of the variable was made and handed to the method to use.

No matter what that method does to that copy, the original remains unchanged.

When I hand out copies of a quiz and you write on your copy, does that change my original copy in anyway?

methods

Every branch of a method needs to return a value.

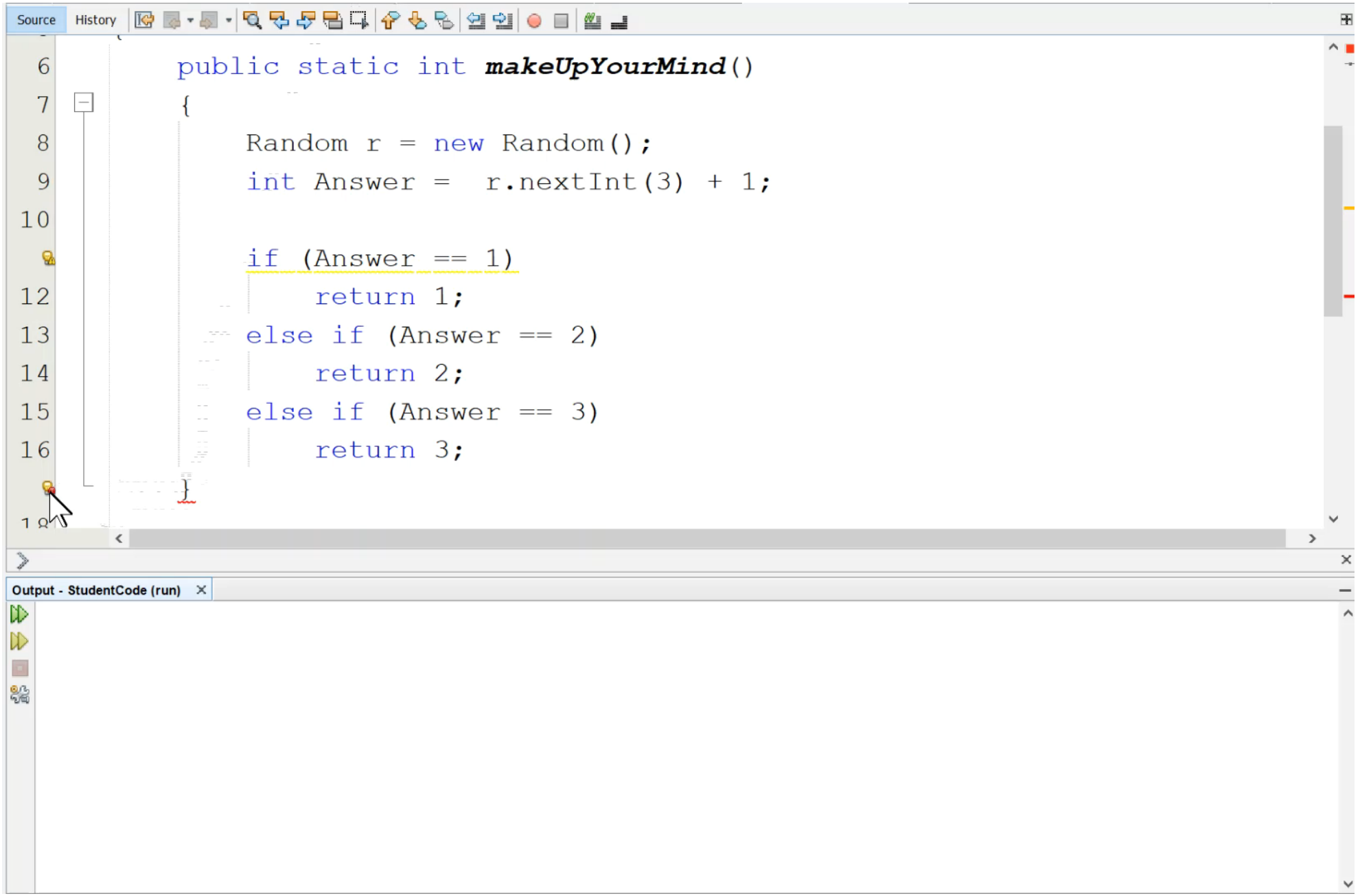
If a method contains logic that has multiple paths – `if-else` or `if-else if-else` or `switch`, then each path through the method needs to have a `return` associated with it.

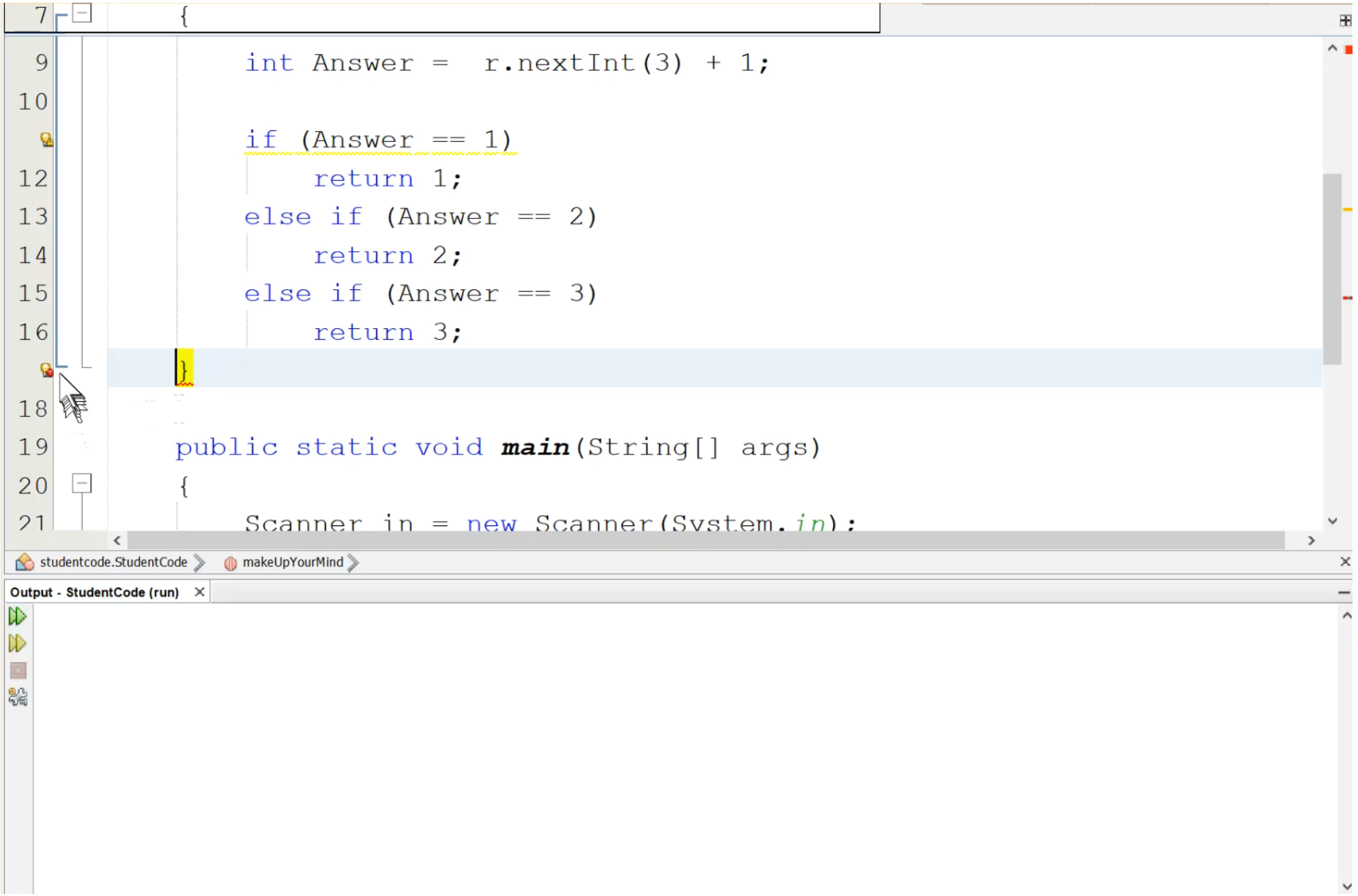
The compiler will report

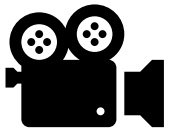
```
missing return statement
```

at the end of the method if you are missing a `return` for any path

```
public static int makeUpYourMind()
{
    Random r = new Random();
    int Answer =  r.nextInt(3) + 1;

    if (Answer == 1)
        return 1;
    else if (Answer == 2)
        return 2;
    else if (Answer == 3)
        return 3;
}

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

    System.out.println(makeUpYourMind());
}
```


Source
History
public static int makeUpYourMind()
{
Random r = new Random();
int Answer = r.nextInt(3) + 1;
if (Answer == 1)
return 1;
else if (Answer == 2)
return 2;
else if (Answer == 3)
return 3;
}
Output - StudentCode (run)


{
int Answer = r.nextInt(3) + 1;
if (Answer == 1)
return 1;
else if (Answer == 2)
return 2;
else if (Answer == 3)
return 3;
}
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
studentcode.StudentCode
makeUpYourMind
Output - StudentCode (run)



Source History

```
6      public static int makeUpYourMind()
7      {
8          Random r = new Random();
9          int Answer =  r.nextInt(3) + 1;
10
           if (Answer == 1)
12             return 1;
13         else if (Answer == 2)
14             return 2;
15         else if (Answer == 3)
16             return 3;
17
18         return 0;
```

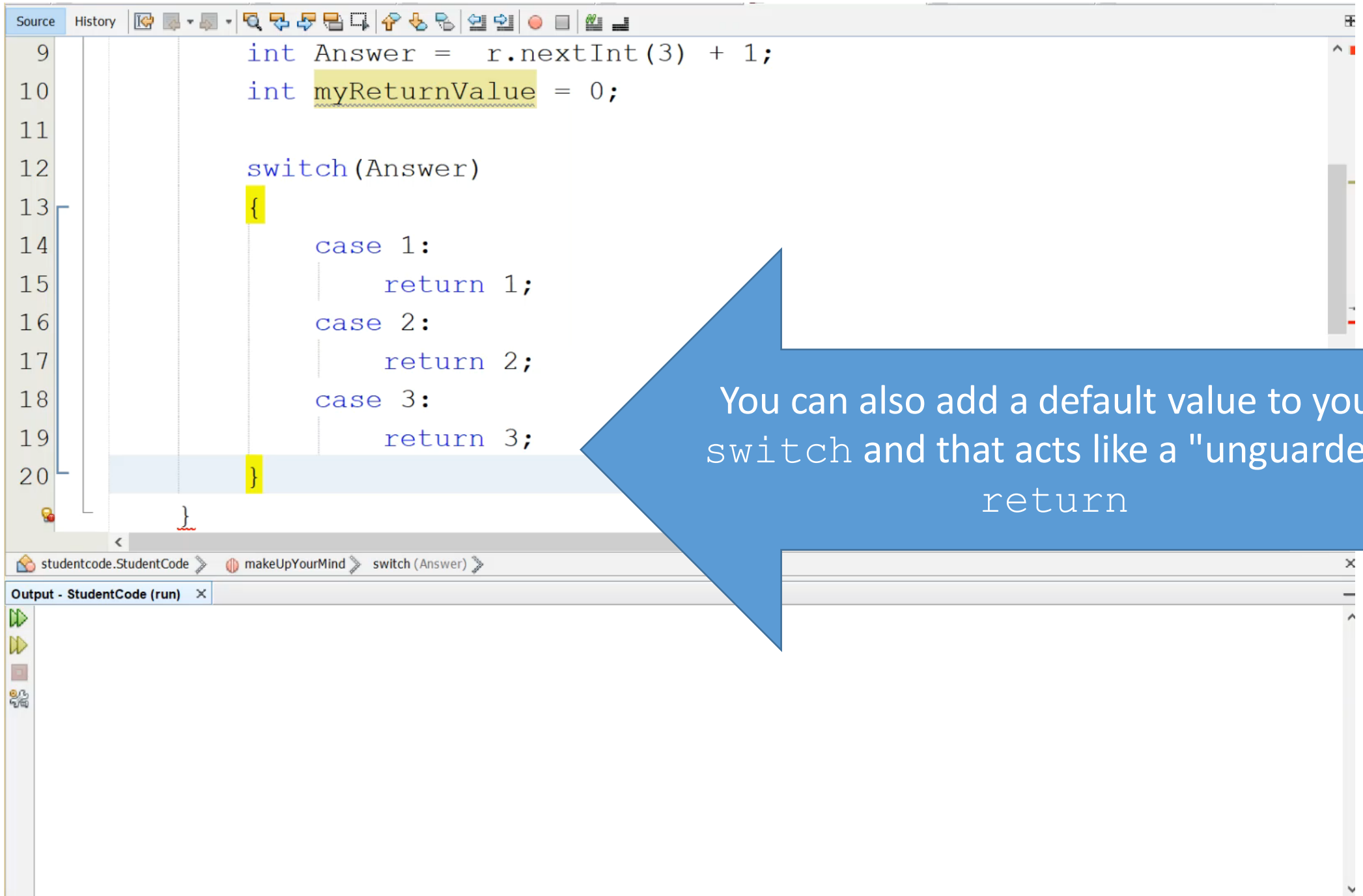
studentcode.StudentCode > makeUpYourMind >

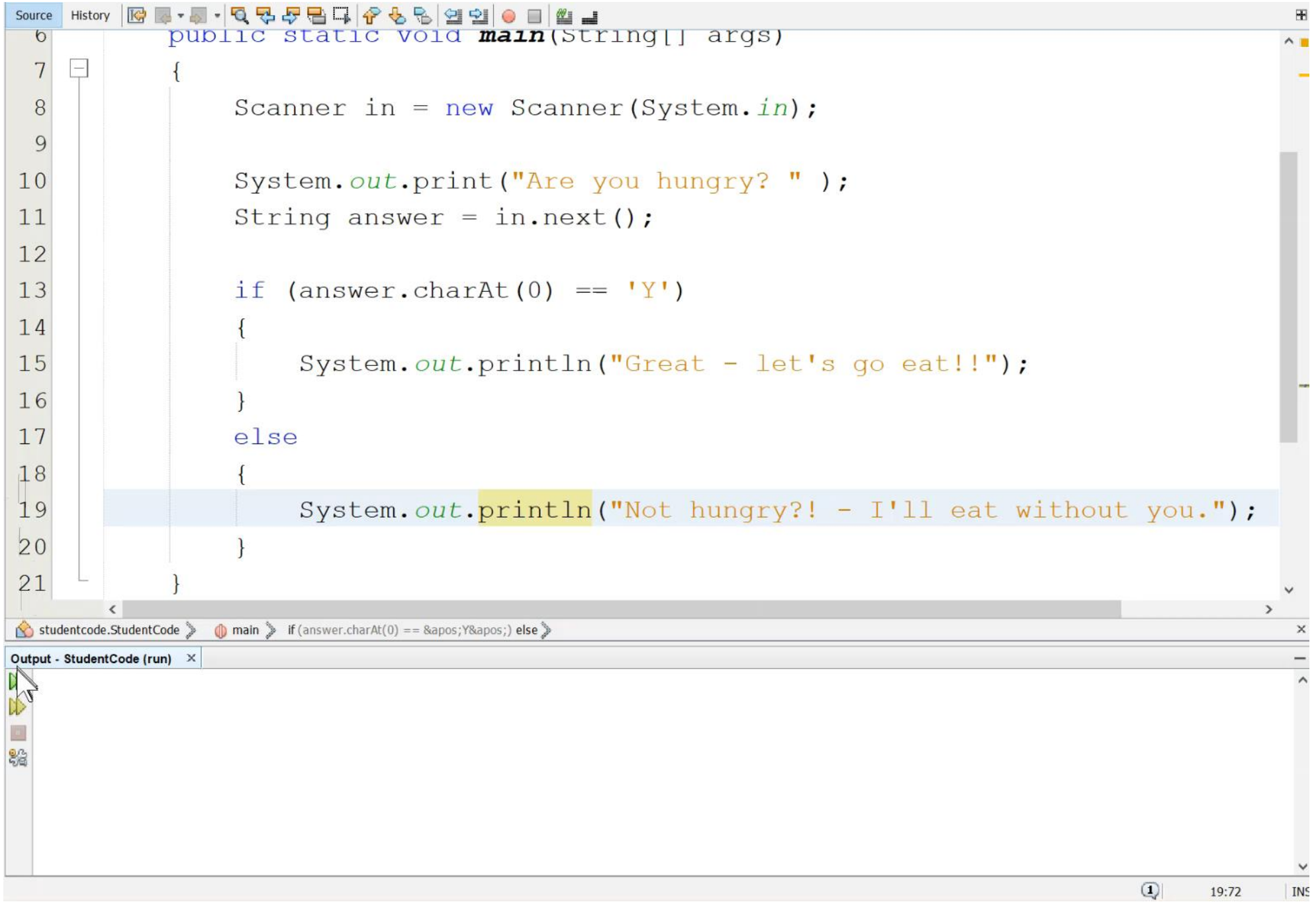
Output - StudentCode (run)

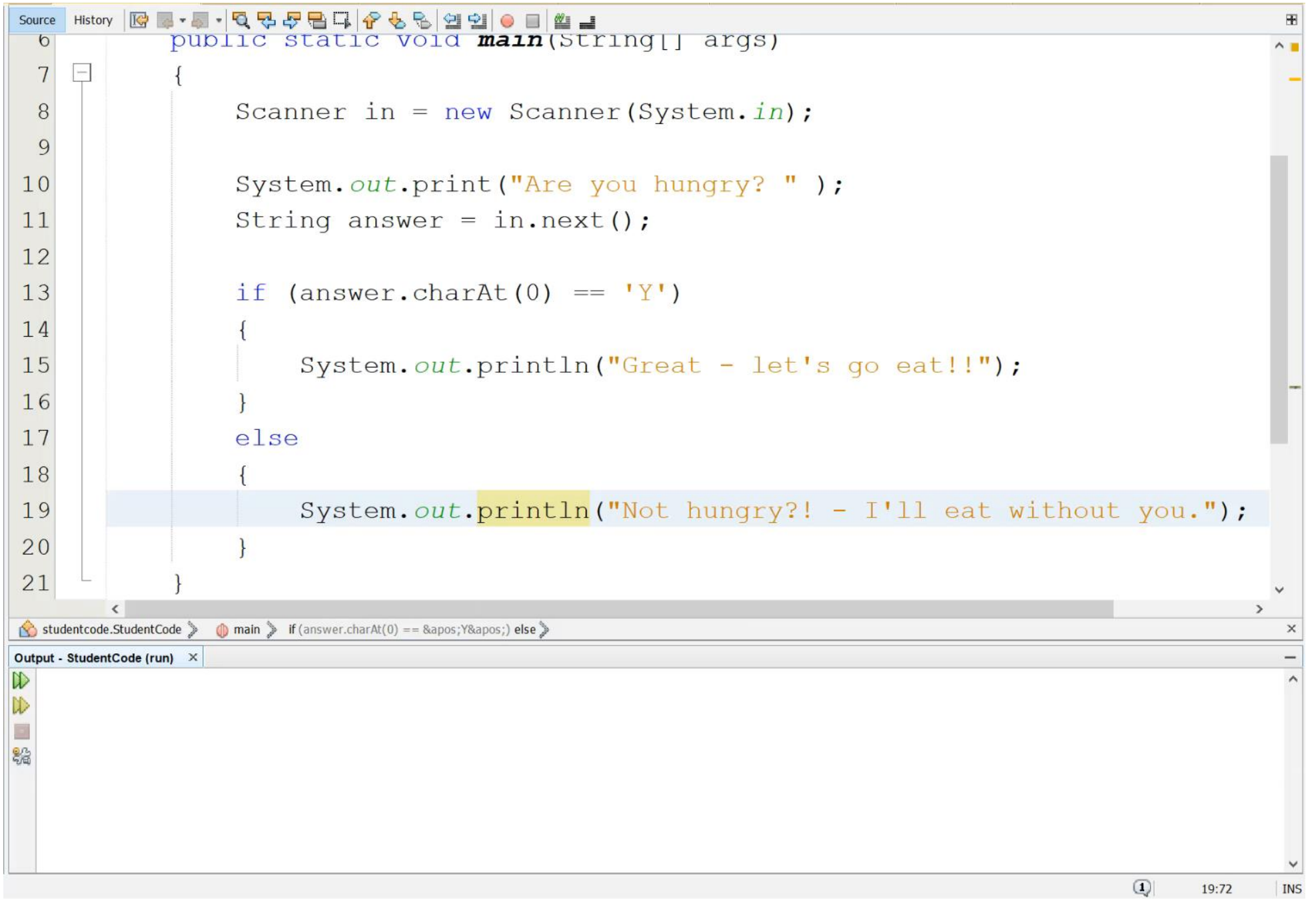
methods

```
public static int makeUpYourMind()
{
    Random r = new Random();
    int Answer =  r.nextInt(3) + 1;

    if (Answer == 1)
        return 1;
    else if (Answer == 2)
        return 2;
    else if (Answer == 3)
        return 3;

    return 0;
}
```

```
public static int makeUpYourMind()
{
    Random r = new Random();
    int Answer =  r.nextInt(3) + 1;
    int myReturnValue = 0;

    if (Answer == 1)
        myReturnValue = 1;
    else if (Answer == 2)
        myReturnValue = 2;
    else if (Answer == 3)
        myReturnValue = 3;

    return myReturnValue;
}
```


Source
History
int Answer = r.nextInt(3) + 1;
int myReturnValue = 0;
switch(Answer)
{
case 1:
return 1;
case 2:
return 2;
case 3:
return 3;
}
}
studentcode.StudentCode
makeUpYourMind
switch (Answer)
Output - StudentCode (run)
You can also add a default value to your switch and that acts like a "unguarded" return


Source
History
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
System.out.print("Are you hungry? ");
String answer = in.next();
if (answer.charAt(0) == 'Y')
{
System.out.println("Great - let's go eat!!");
}
else
{
System.out.println("Not hungry?! - I'll eat without you.");
}
}
studentcode.StudentCode
main
if (answer.charAt(0) == 'Y') else
Output - StudentCode (run)
19:72
INS


Source
History
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
System.out.print("Are you hungry? ");
String answer = in.next();
if (answer.charAt(0) == 'Y')
{
System.out.println("Great - let's go eat!!");
}
else
{
System.out.println("Not hungry?! - I'll eat without you.");
}
}
studentcode.StudentCode
main
if (answer.charAt(0) == 'Y') else
Output - StudentCode (run)
19:72
INS

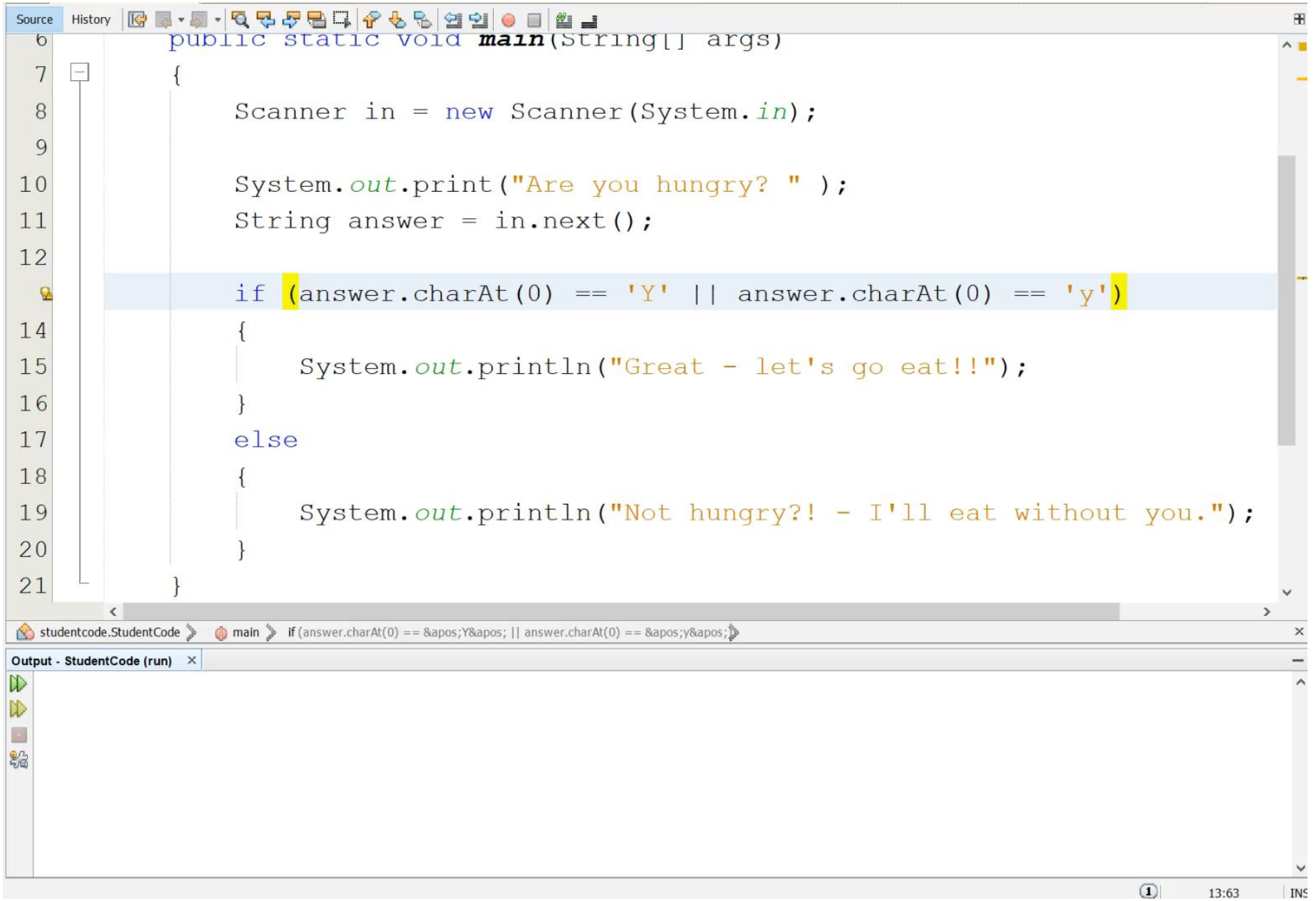
`toUpperCase` and `toLowerCase`

`toUpperCase`

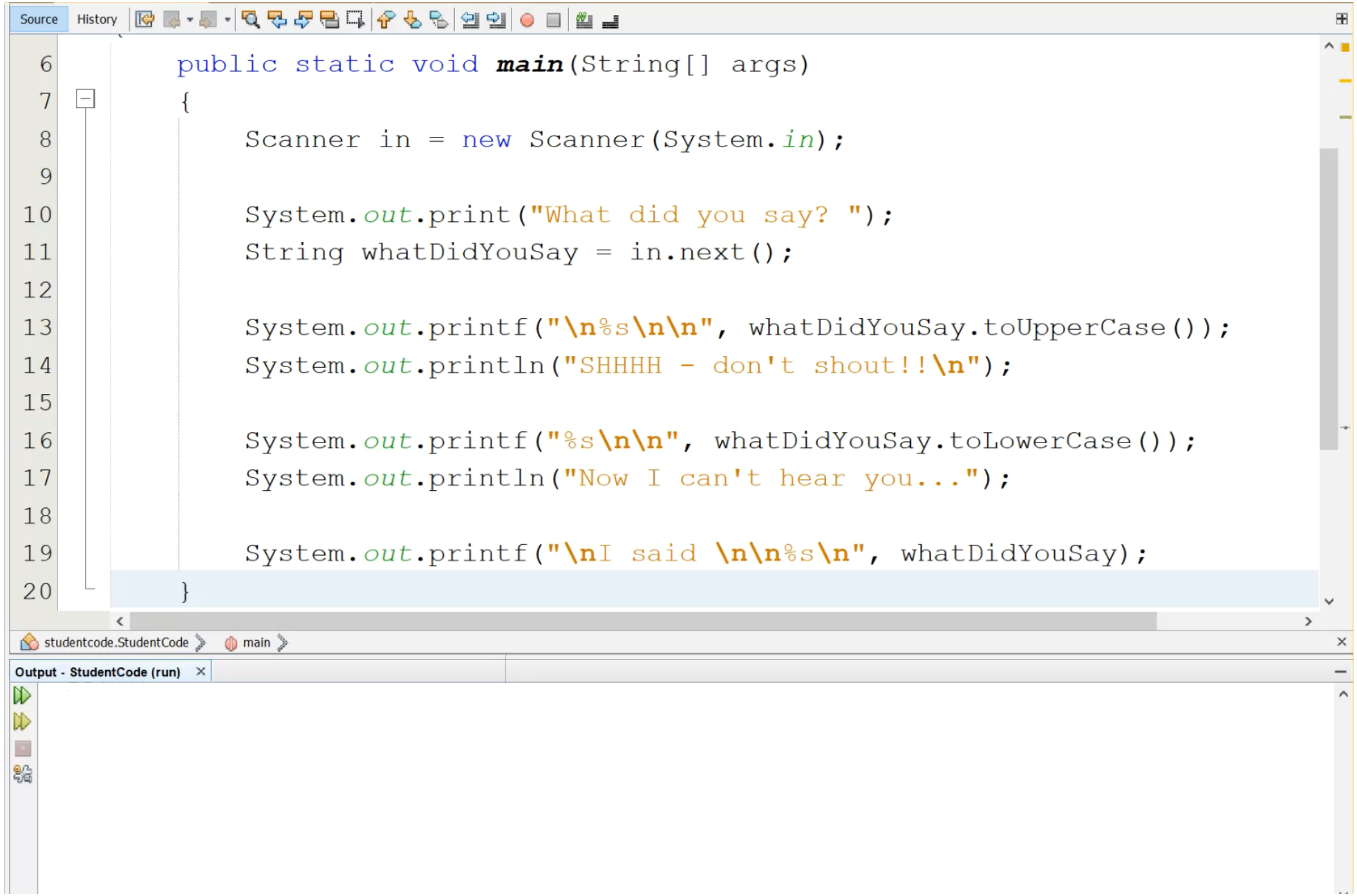
This method returns a new string that consists of all characters in the string converted to uppercase.

`toLowerCase`

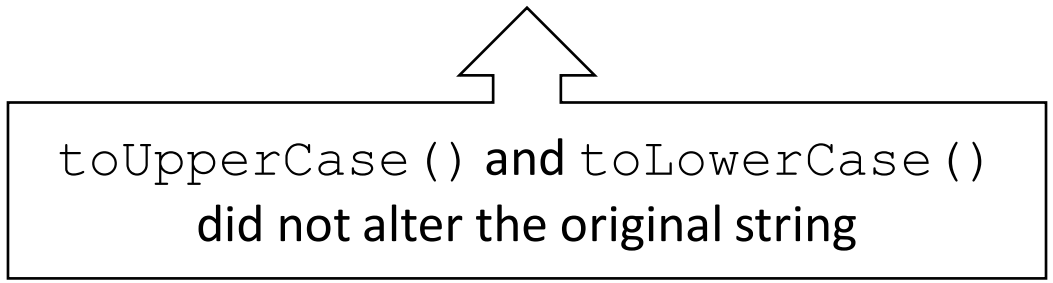
This method returns a new string that consists of all characters in the string converted to lowercase.


Source
History
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
System.out.print("Are you hungry? ");
String answer = in.next();
if (answer.charAt(0) == 'Y' || answer.charAt(0) == 'y')
{
System.out.println("Great - let's go eat!!");
}
else
{
System.out.println("Not hungry?! - I'll eat without you.");
}
}
studentcode.StudentCode
main
if (answer.charAt(0) == 'Y' || answer.charAt(0) == 'y')
Output - StudentCode (run)
13:63
INS

```
System.out.print("Are you hungry? " );
String answer = in.next();
answer = answer.toUpperCase();

if (answer.charAt(0) == 'Y')
{
    System.out.println("Great - let's go eat!!");
}
else
{
    System.out.println("Not hungry?! - I'll eat without you.");
}
```


Source
History
6 public static void main(String[] args)
7 {
8 Scanner in = new Scanner(System.in);
9
10 System.out.print("What did you say? ");
11 String whatDidYouSay = in.next();
12
13 System.out.printf("\n%s\n\n", whatDidYouSay.toUpperCase());
14 System.out.println("SHHHH - don't shout!!\n");
15
16 System.out.printf("%s\n\n", whatDidYouSay.toLowerCase());
17 System.out.println("Now I can't hear you...");
18
19 System.out.printf("\nI said \n\n%s\n", whatDidYouSay);
20 }
studentcode.StudentCode
main
Output - StudentCode (run)

```
System.out.print("What did you say? ");
String whatDidYouSay = in.next();

System.out.printf("\n%s\n\n", whatDidYouSay.toUpperCase());
System.out.println("SHHHH - don't shout!!\n");

System.out.printf("%s\n\n", whatDidYouSay.toLowerCase());
System.out.println("Now I can't hear you...");

System.out.printf("\nI said \n\n%s\n", whatDidYouSay);
```



`toUpperCase` and `toLowerCase`

The definitions of both `toUpperCase()` and `toLowerCase()` state

"this method returns a new string that consists of all characters in the string converted to …"

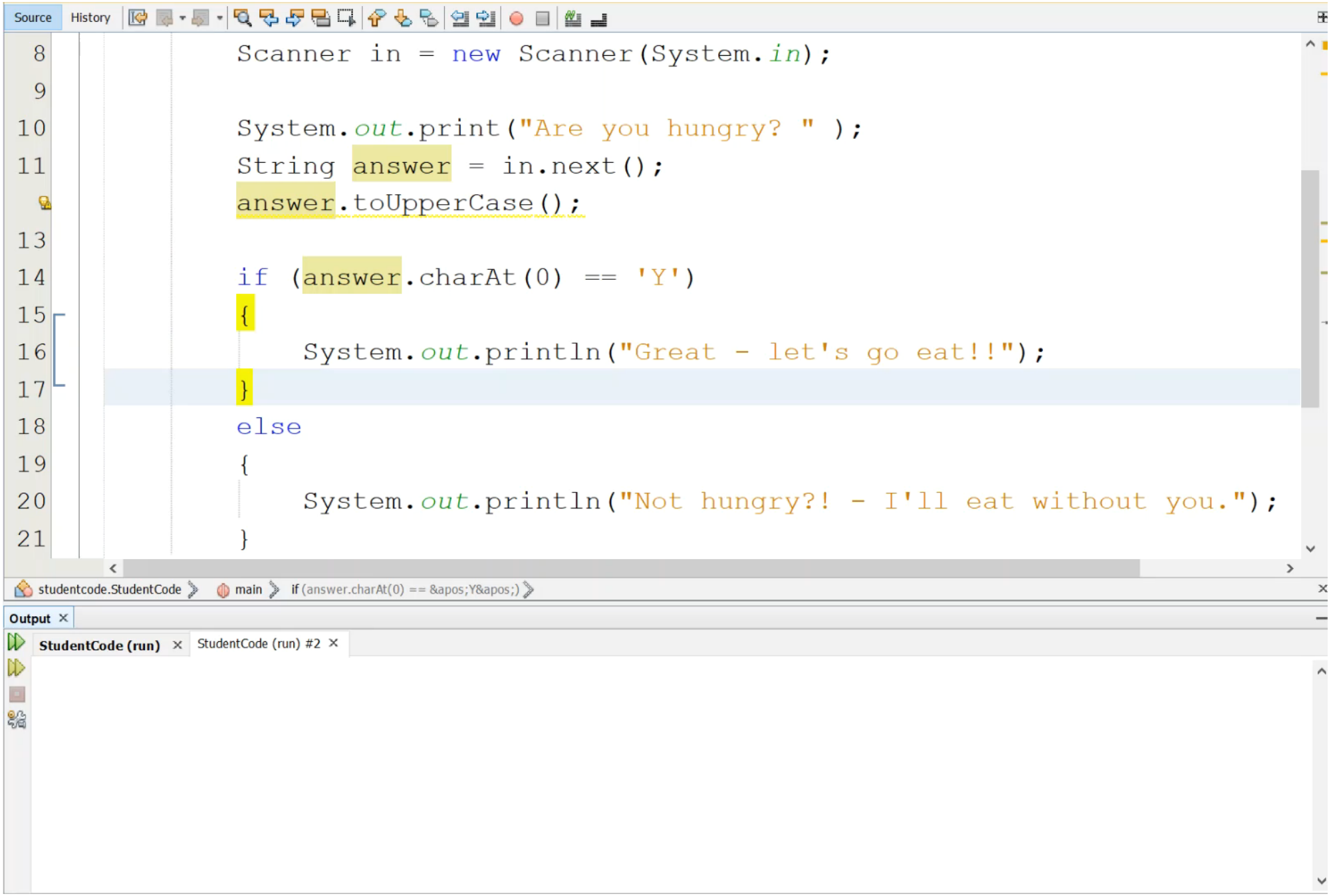
These methods do not alter the string – they return a new string. If you want to alter your string, then set your string equal to the return value of the method.

```
myString = toUpperCase(myString);
```

`toUpperCase` and `toLowerCase`

#1 Error when using `toUpperCase()` and `toLowerCase()`

```
System.out.print("Are you hungry? " );
String answer = in.next();
answer.toUpperCase();

if (answer.charAt(0) == 'Y')
{
    System.out.println("Great - let's go eat!!");
}
else
{
    System.out.println("Not hungry?! - I'll eat without you.");
}
```


Source
History
8 Scanner in = new Scanner(System.in);
9
10 System.out.print("Are you hungry? ");
11 String answer = in.next();
answer.toUpperCase();
13
14 if (answer.charAt(0) == 'Y')
15 {
16 System.out.println("Great - let's go eat!!");
17 }
18 else
19 {
20 System.out.println("Not hungry?! - I'll eat without you.");
21 }
studentcode.StudentCode
main
if (answer.charAt(0) == 'Y')
Output
StudentCode (run)
StudentCode (run) #2

`toUpperCase` and `toLowerCase`

#1 Error when using `toUpperCase()` and `toLowerCase()`

```
System.out.print("Are you hungry? " );
String answer = in.next();
answer.toUpperCase();

if (answer.charAt(0) == 'Y')
{
    System.out.println("Great - let's go eat!!");
}
else
{
    System.out.println("Not hungry?! - I'll eat without you.");
}
```

must be an assignment

```
answer = answer.toUpperCase();
```

`toUpperCase` and `toLowerCase`

```
String answer = in.next();
answer.toUpperCase();

if (answer.charAt(0) == 'Y')
```

`toUpperCase()` returns the uppercased version of the string but nothing catches it and it is lost

VS

```
System.out.printf("\n%s\n\n", whatDidYouSay.toUpperCase());
```

return value is given to the `%s` of `System.out.printf()` and used to print the string in uppercase

How to read a single character from the input

```
Scanner Cat = new Scanner(System.in);

System.out.print("Enter a string ");
String Yarn = Cat.next();

System.out.printf("The first letter is %c\n", Yarn.charAt(0));
```

```
Enter a string Kitty
The first letter is K
```

This code will print the first letter of the string but what if we don't want the whole string – what if we only want to store the first letter?

How to read a single character from the input

```
Scanner Cat = new Scanner(System.in);


System.out.print("Enter a string ");
String Yarn = Cat.next();
```

```
C:\Users\frenc\Documents\NetBeansProjects\ReadSingleChar\src\readsinglechar\
ReadSingleChar.java:14: error: incompatible types: String cannot be
converted to char
        char Yarn = Cat.next();
```

How to read a single character from the input

```
Scanner Cat = new Scanner(System.in);

System.out.print("Enter a string ");
char Yarn = Cat.next().charAt(0);

System.out.printf("The first letter is %c\n", Yarn);

Enter a string Kitty
The first letter is K
```

How to read a single character from the input

```
Scanner Cat = new Scanner(System.in);

System.out.print("Enter a string ");
char Yarn = Cat.next().toUpperCase().charAt(0);

System.out.printf("The first letter is %c\n", Yarn);

Enter a string kitty
The first letter is K
```

```
Scanner Cat = new Scanner(System.in);
System.out.print("Which character do you want? ");
int pick = Cat.nextInt();

System.out.print("Enter a string ");
char Yarn = Cat.next().toUpperCase().charAt(pick);

System.out.printf("Your letter is %c\n", Yarn);
```

```
Which character do you want? 12
Enter a string supercalifragilisticexpialidocious
Your letter is G
```

supercalifragilisticexpialidocious

```
char Yarn = Cat.next().toUpperCase().charAt(pick);
```

Question…

Can we write this as

```
char Yarn = Cat.next().charAt(pick).toUpperCase();
```

That would say to get the input, get only the first letter and then uppercase it. Right?

Slide - Apache NetBeans IDE 12.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

423.5/572.0MB

FinalDemo1325.java | HelloWorld1325.java | NewLineLeftBehind.java | Slide.java

Source History

Projects Files Services

```
10      public static void main(String[] args)
11      {
12          Scanner Cat = new Scanner(System.in);
13          System.out.print("Which character do you want? ");
14          int pick = Cat.nextInt();
15
16          System.out.print("Enter a string ");
17          char Yarn = Cat.next().toUpperCase().charAt(pick);
18          //char Yarn = Cat.next().charAt(pick).toUpperCase();
19
20          System.out.printf("Your letter is %c\n", Yarn);
21      }
```

Output - Slide (run)

24:1 INS

`toUpperCase` and `toLowerCase`

`toUpperCase`

This method returns a new string that consists of all characters in the string converted to uppercase.

`toLowerCase`

This method returns a new string that consists of all characters in the string converted to lowercase.