Java & JEE Training

Day 3 – Elements of Java programming (Continued..)
Deep dive into Java programming

MindsMapped Consulting

Agenda

- Elements of Java programming language
    - Conditional Statements
    - Loops

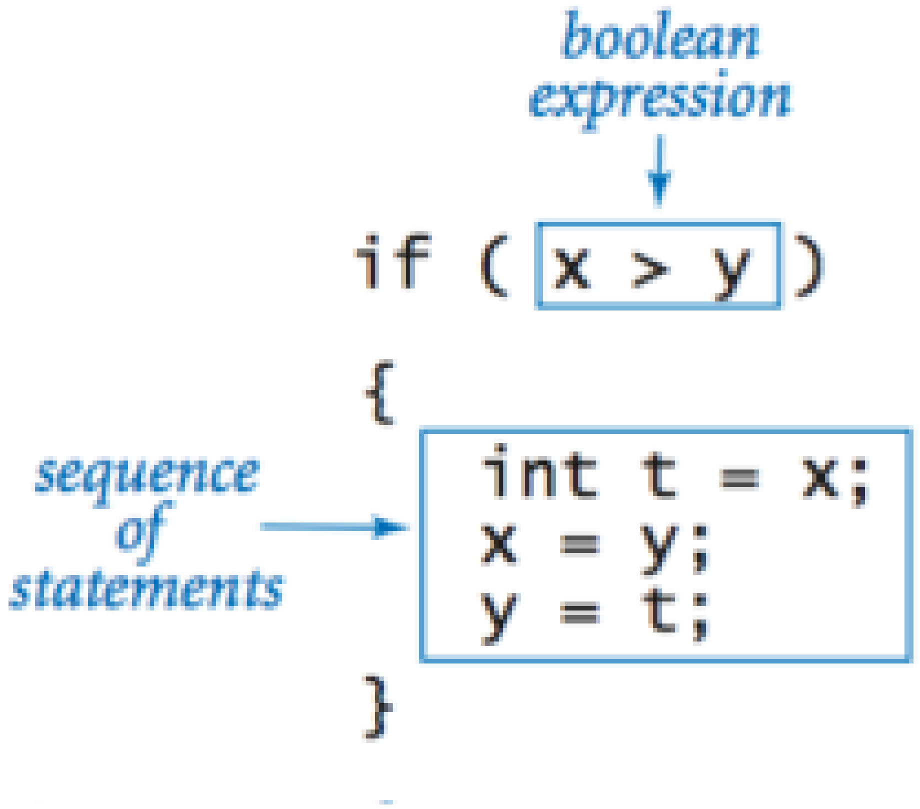
Conditional Statements

MindsMapped Consulting

Conditional Statements

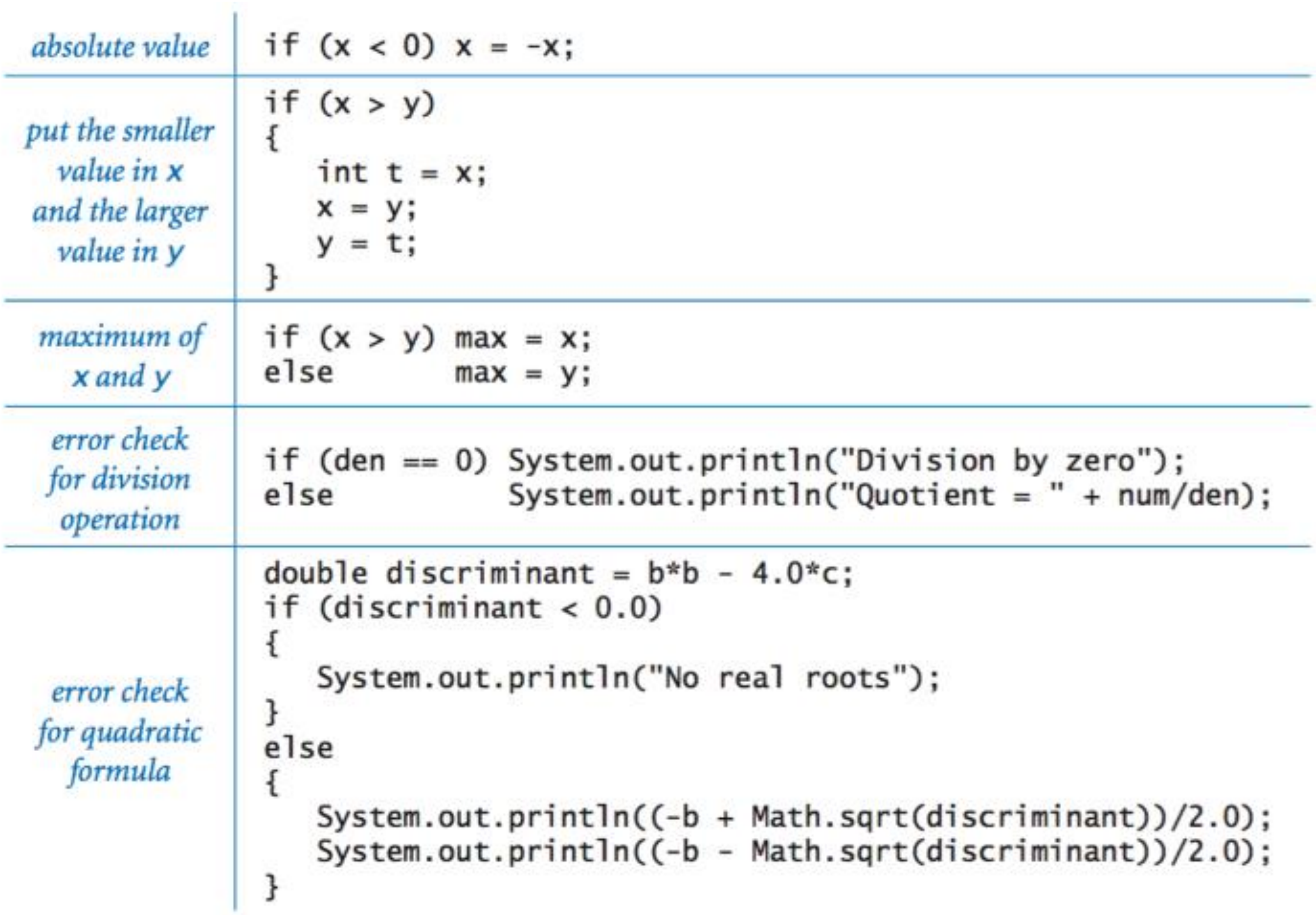
- if
- if-else
- if-else ladder
- switch-case
- Ternary operator

if



```
if ( x > y )
{
   int t = x;
   x = y;
   y = t;
}
```

if and if-else examples



absolute value

```
if (x < 0) x = -x;
```

put the smaller value in x and the larger value in y

```
if (x > y)
{
   int t = x;
   x = y;
   y = t;
}
```

maximum of x and y

```
if (x > y) max = x;
else       max = y;
```

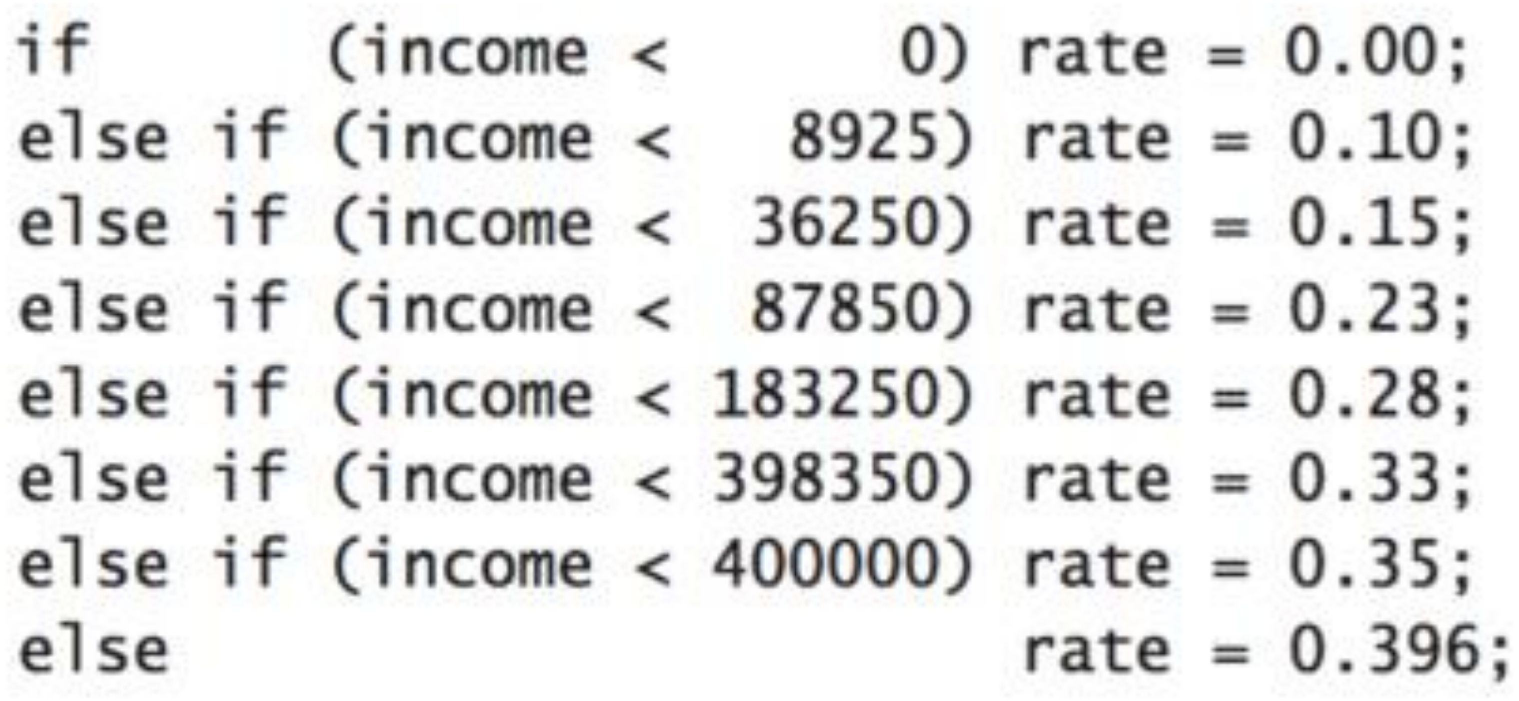
error check for division operation

```
if (den == 0) System.out.println("Division by zero");
else          System.out.println("Quotient = " + num/den);
```

error check for quadratic formula

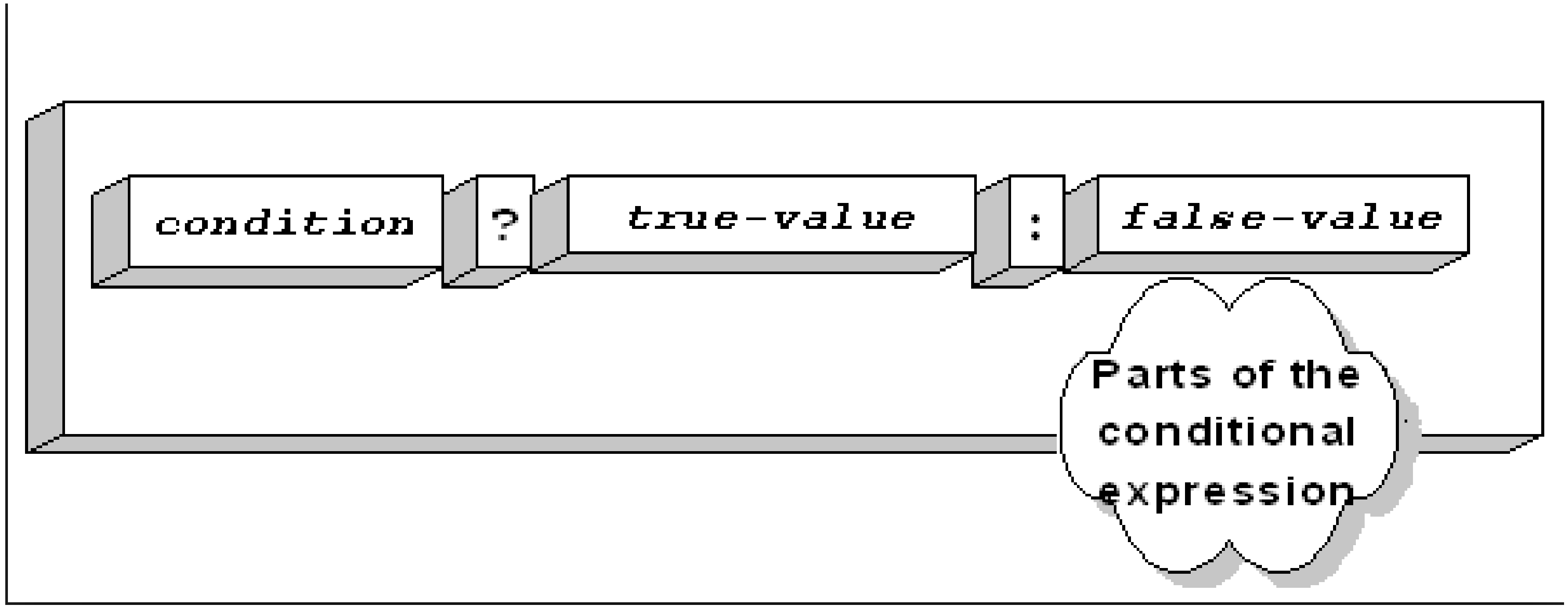
```
double discriminant = b*b - 4.0*c;
if (discriminant < 0.0)
{
   System.out.println("No real roots");
}
else
{
   System.out.println((-b + Math.sqrt(discriminant))/2.0);
   System.out.println((-b - Math.sqrt(discriminant))/2.0);
}
```

if-else ladder



```
if      (income <      0) rate = 0.00;
else if (income <   8925) rate = 0.10;
else if (income <  36250) rate = 0.15;
else if (income <  87850) rate = 0.23;
else if (income < 183250) rate = 0.28;
else if (income < 398350) rate = 0.33;
else if (income < 400000) rate = 0.35;
else                      rate = 0.396;
```

Ternary Operator

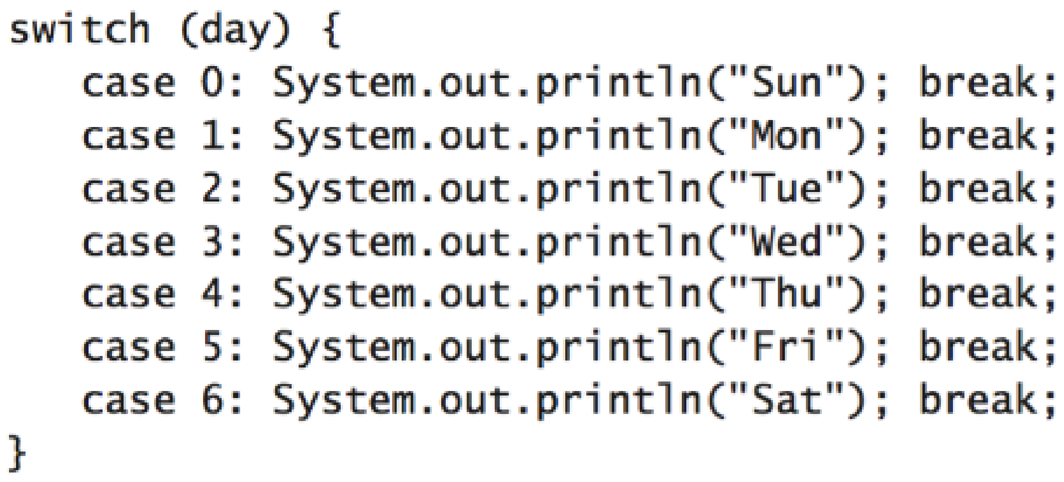


Switch...case statement

Unlike if-then and if-then-else statements, the switch statement can have a number of possible execution paths. A switch works with the byte, short, char, and int primitive data types. It also works with enumerated types (discussed in Enum Types), the String class, and a few special classes that wrap certain primitive types: Character, Byte, Short, and Integer (discussed in Numbers and Strings).

*** Provide default value – it is a good practice.**

*** The break is important. Don't forget it.**



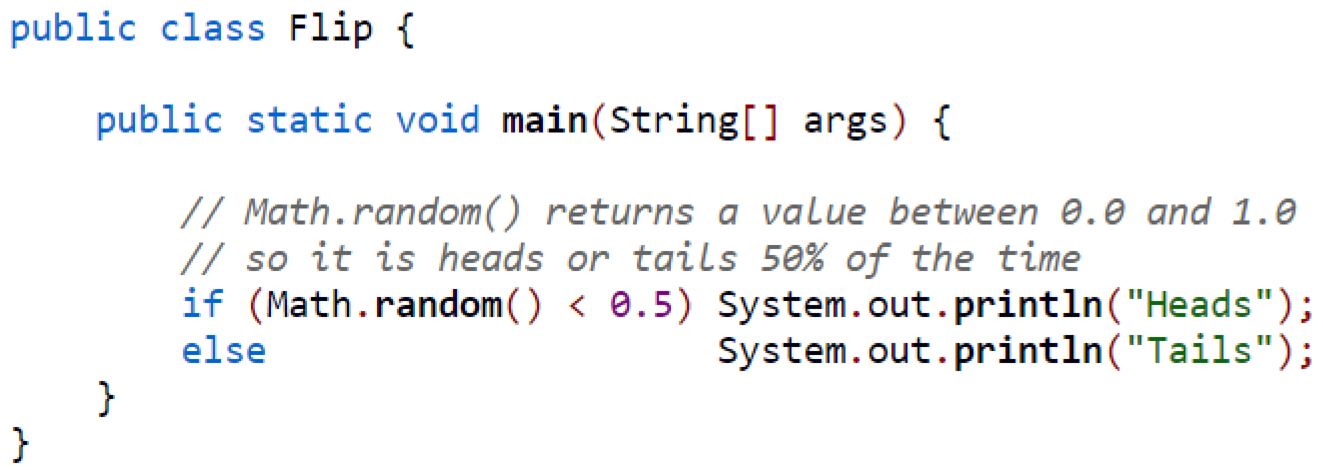
```
switch (day) {
    case 0: System.out.println("Sun"); break;
    case 1: System.out.println("Mon"); break;
    case 2: System.out.println("Tue"); break;
    case 3: System.out.println("Wed"); break;
    case 4: System.out.println("Thu"); break;
    case 5: System.out.println("Fri"); break;
    case 6: System.out.println("Sat"); break;
}
```

Exercise

Use Math.random() and an if-else statement to print the results of a coin flip.

Exercise

Use Math.random() and an if-else statement to print the results of a coin flip.



```java
public class Flip {

    public static void main(String[] args) {

        // Math.random() returns a value between 0.0 and 1.0
        // so it is heads or tails 50% of the time
        if (Math.random() < 0.5) System.out.println("Heads");
        else                     System.out.println("Tails");
    }
}
```

Exercise

- Write two programs - one using switch case and another using if else ladder to print the full name of a month provided that the user inputs an integer between 1 to 12.
- E.g. if the user inputs 1, print January.

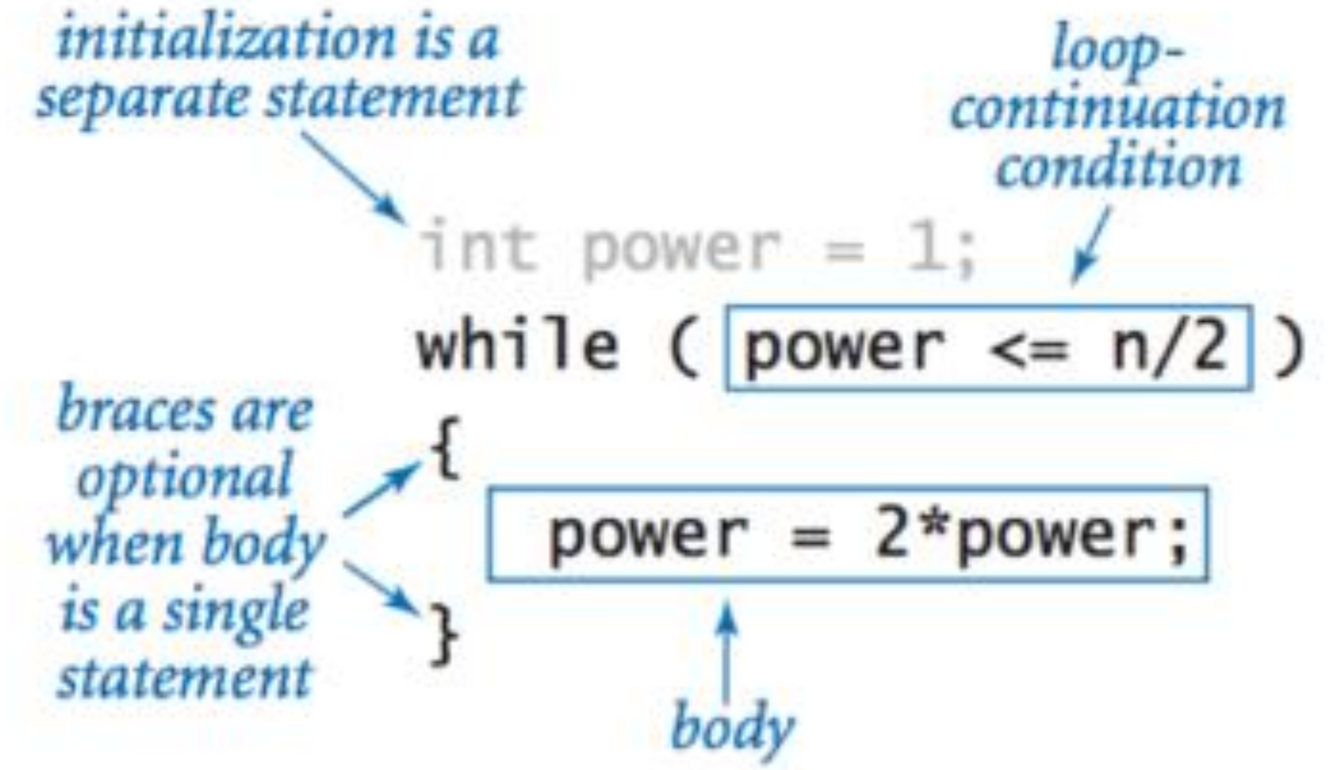
Loops

MindsMapped Consulting

Loops

- while
- do..while
- for

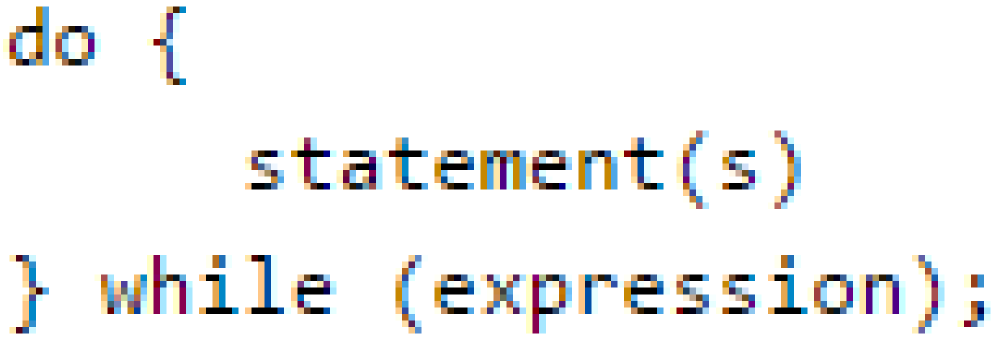
while



```
int power = 1;
while ( power <= n/2 )
{
    power = 2*power;
}
```

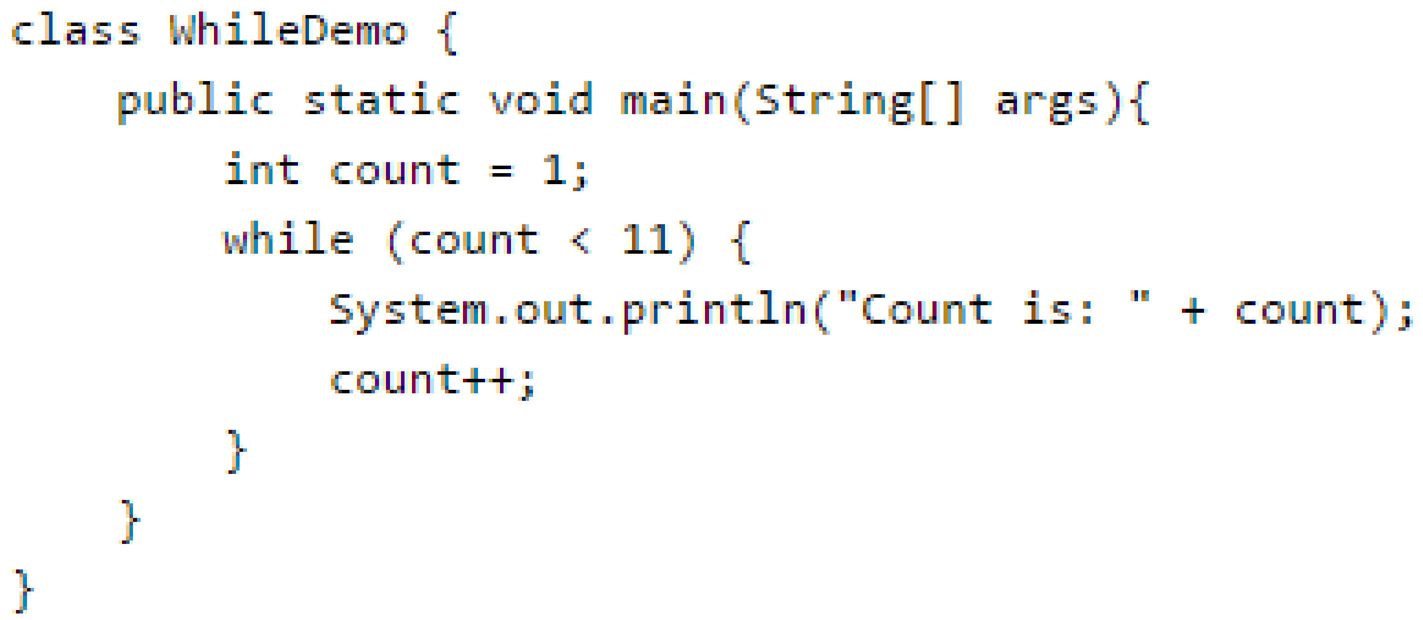
initialization is a separate statement
loop-continuation condition
braces are optional when body is a single statement
body

do-while



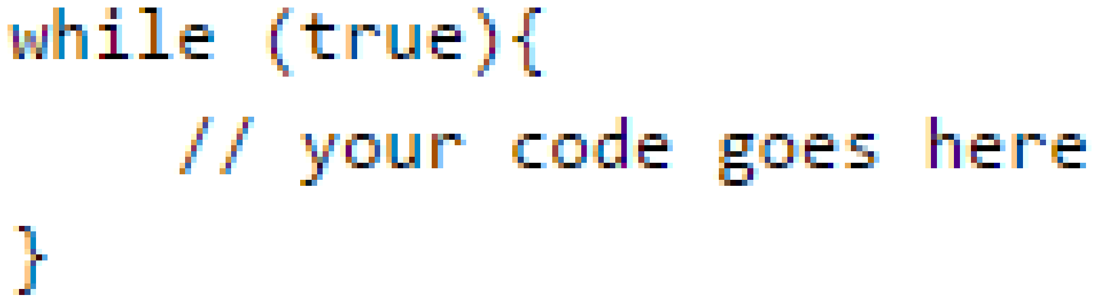
```
do {
      statement(s)
} while (expression);
```

Exercise – what is output?



```
class WhileDemo {
     public static void main(String[] args){
          int count = 1;
          while (count < 11) {
               System.out.println("Count is: " + count);
               count++;
          }
     }
}
```

Exercise – what is the output?

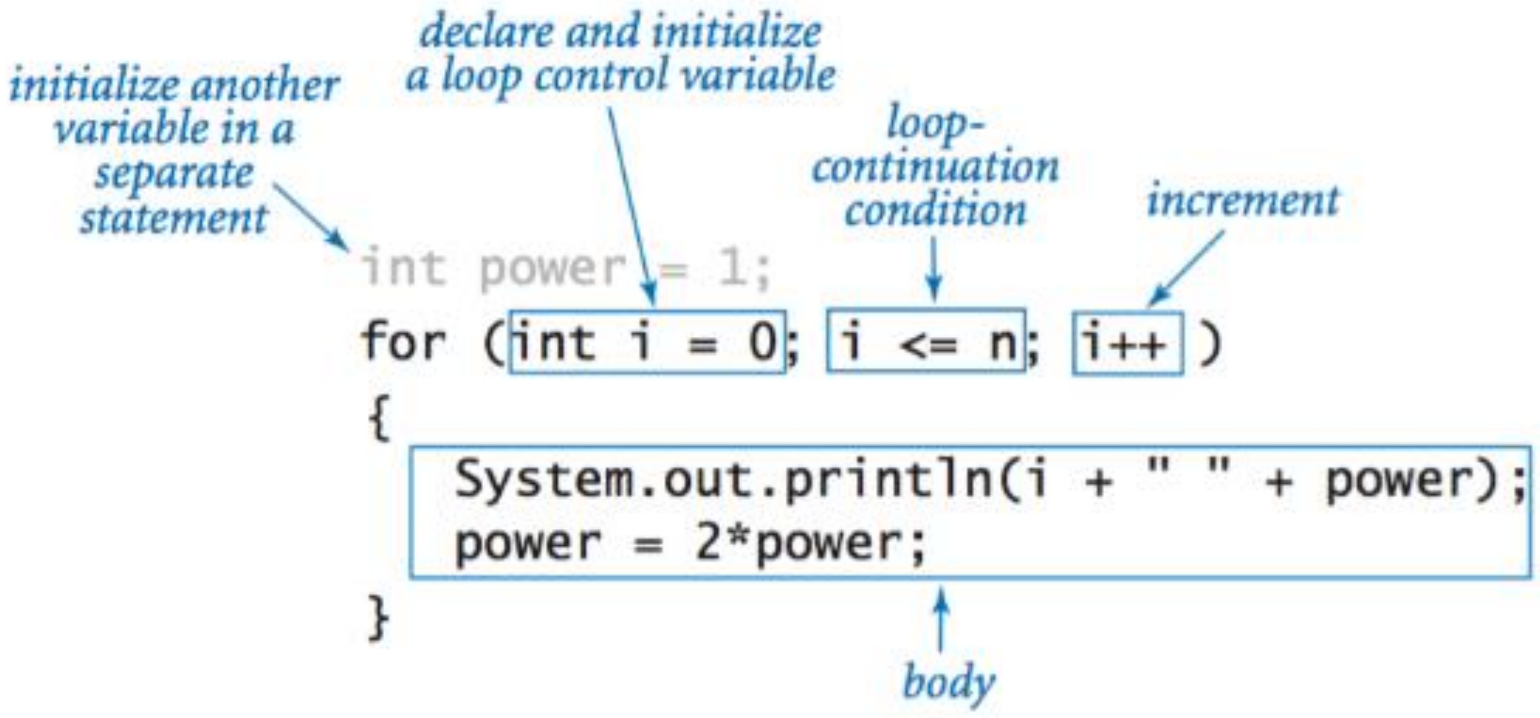


```
while (true){
    // your code goes here
}
```

Difference between while and do-while?

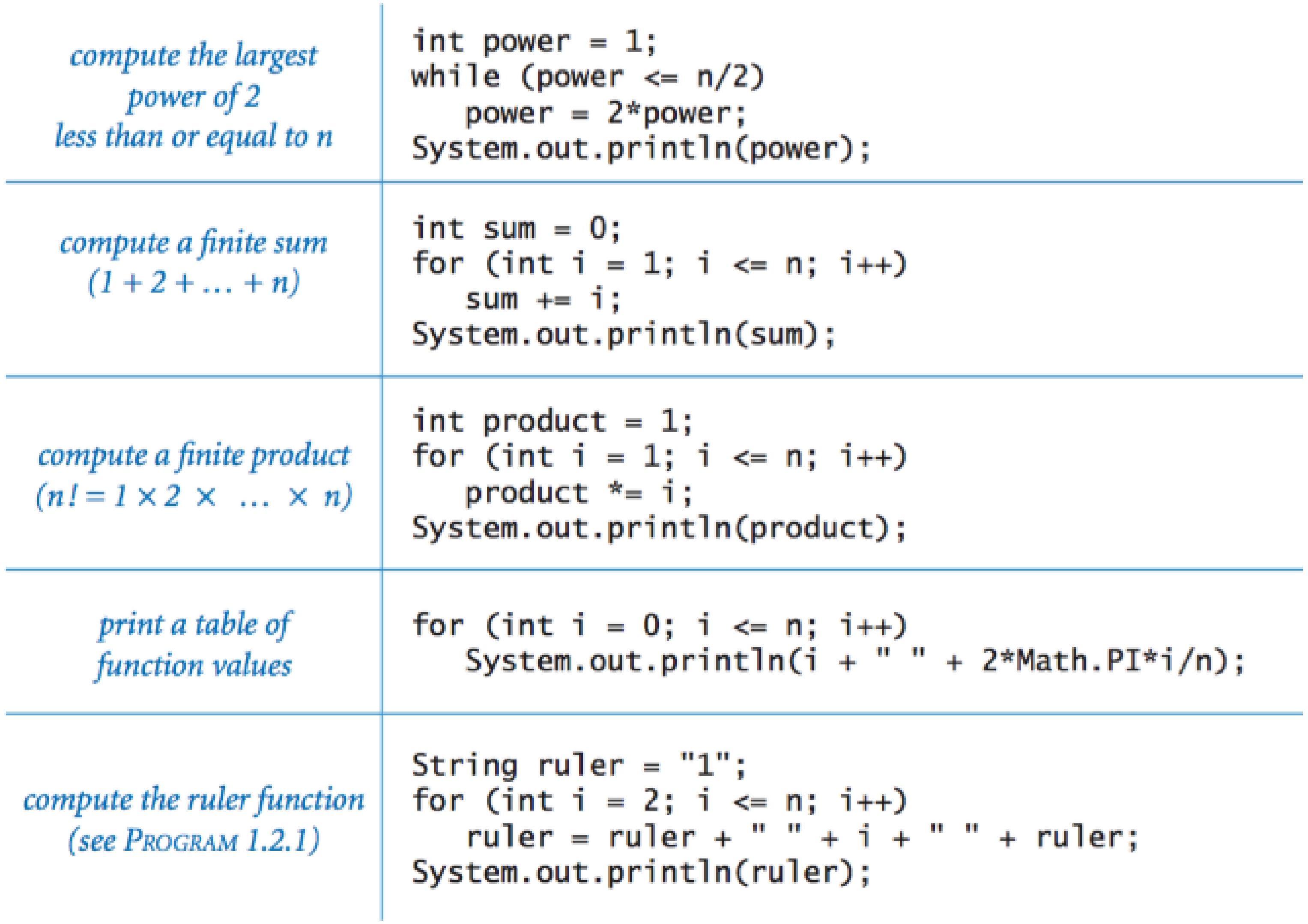
Do-while loop is always executed at least once, as the condition is checked at the end of the loop.

for statement


initialize another variable in a separate statement
declare and initialize a loop control variable
loop-continuation condition
increment
body

```
int power = 1;
for (int i = 0; i <= n; i++ )
{
   System.out.println(i + " " + power);
   power = 2*power;
}
```

Loop Examples



compute the largest power of 2 less than or equal to n

```
int power = 1;
while (power <= n/2)
   power = 2*power;
System.out.println(power);
```

compute a finite sum $(1 + 2 + \ldots + n)$

```
int sum = 0;
for (int i = 1; i <= n; i++)
   sum += i;
System.out.println(sum);
```

compute a finite product $(n! = 1 \times 2 \times \ldots \times n)$

```
int product = 1;
for (int i = 1; i <= n; i++)
   product *= i;
System.out.println(product);
```

print a table of function values

```
for (int i = 0; i <= n; i++)
   System.out.println(i + " " + 2*Math.PI*i/n);
```

compute the ruler function (see P*ROGRAM* *1.2.1)*

```
String ruler = "1";
for (int i = 2; i <= n; i++)
   ruler = ruler + " " + i + " " + ruler;
System.out.println(ruler);
```

Branching Statements: Break, Continue and Return…

Break:

The break statement has two forms: labeled and unlabeled. You saw the unlabeled form in the previous discussion of the switch statement. You can also use an unlabeled break to terminate a for, while, or do-while loop

An unlabeled break statement terminates the innermost switch, for, while, or do-while statement, but a labeled break terminates an outer statement.

Break statement: Example (Without Label)

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

        int[] arrayOfInts =
            { 32, 87, 3, 589,
              12, 1076, 2000,
              8, 622, 127 };
        int searchfor = 12;

        int i;
        boolean foundIt = false;

        for (i = 0; i < arrayOfInts.length; i++) {
            if (arrayOfInts[i] == searchfor) {
                foundIt = true;
                break;
            }
        }

        if (foundIt) {
            System.out.println("Found " + searchfor + " at index " + i);
        } else {
            System.out.println(searchfor + " not in the array");
        }
    }
}
```

Break statement: Example (With Label)

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

        int[][] arrayOfInts = {
            { 32, 87, 3, 589 },
            { 12, 1076, 2000, 8 },
            { 622, 127, 77, 955 }
        };
        int searchfor = 12;

        int i;
        int j = 0;
        boolean foundIt = false;

    search:
        for (i = 0; i < arrayOfInts.length; i++) {
            for (j = 0; j < arrayOfInts[i].length;
                 j++) {
                if (arrayOfInts[i][j] == searchfor) {
                    foundIt = true;
                    break search;
                }
            }
        }

        if (foundIt) {
            System.out.println("Found " + searchfor + " at " + i + ", " + j);
        } else {
            System.out.println(searchfor + " not in the array");
        }
    }
}
```

Branching Statements: Break, Continue and Return…

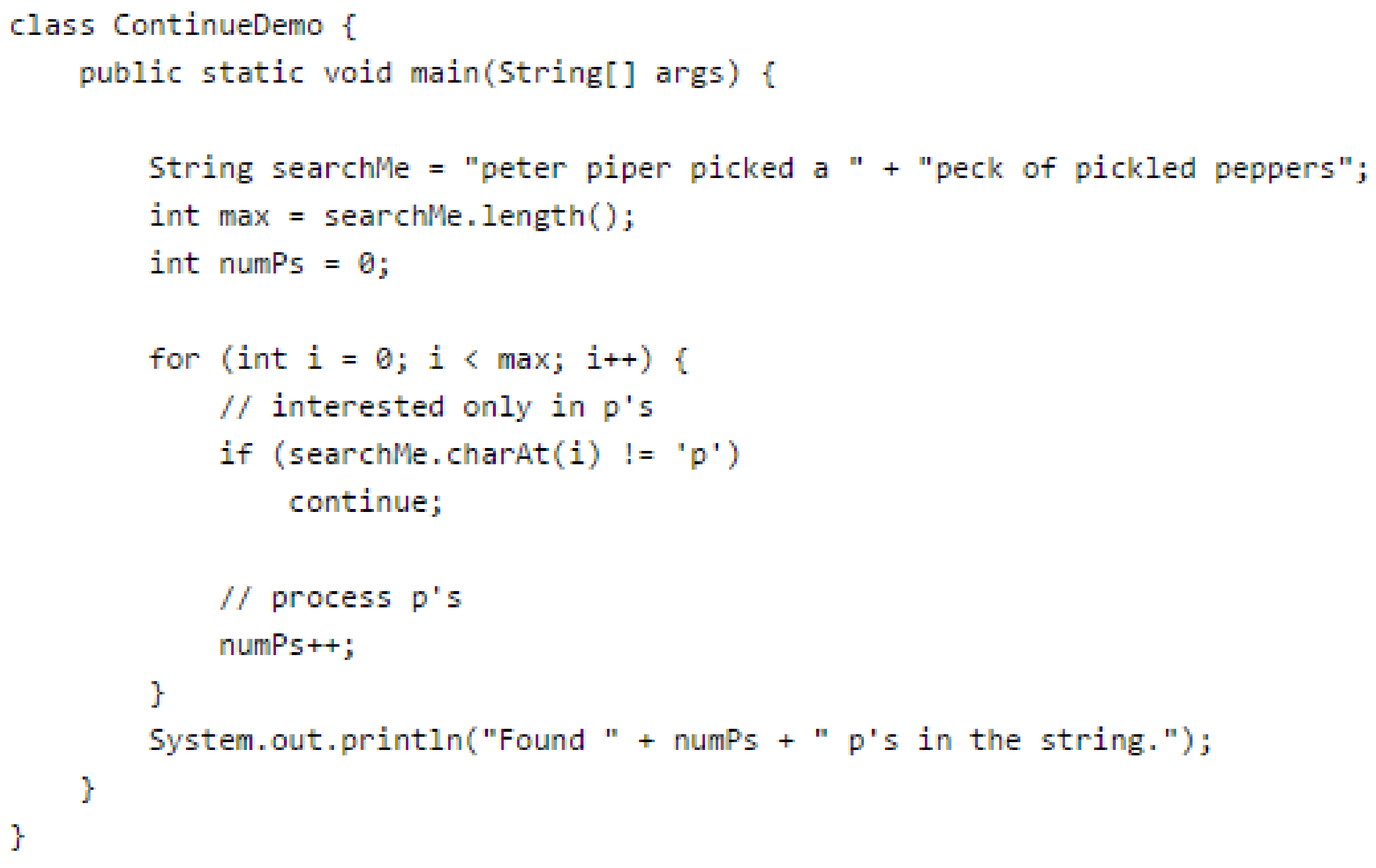
Continue:

The continue statement skips the current iteration of a for, while , or do-while loop.

The unlabeled form skips to the end of the innermost loop's body and evaluates the boolean expression that controls the loop.

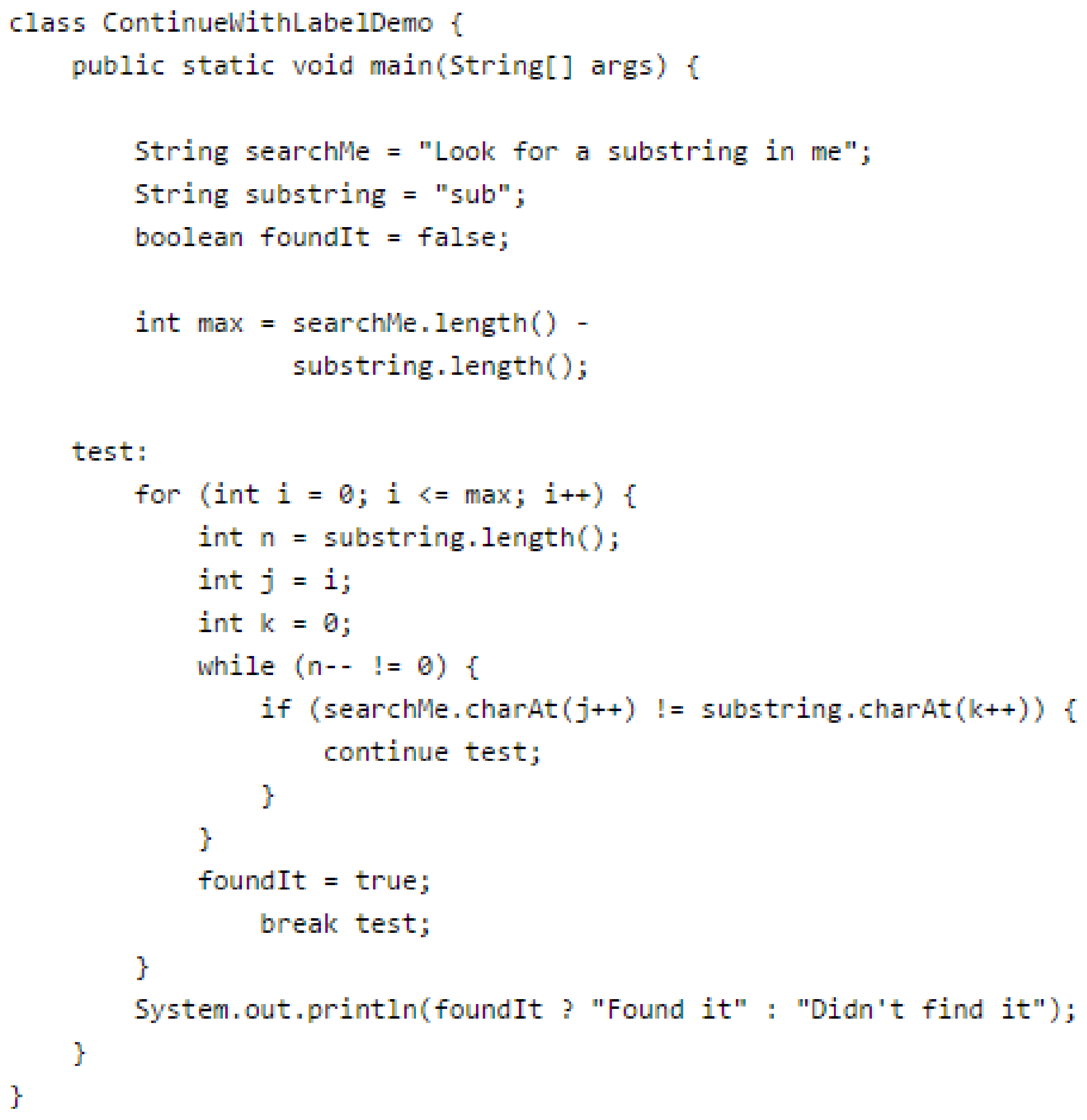
The following program, ContinueDemo , steps through a String, counting the occurences of the letter "p". If the current character is not a p, the continue statement skips the rest of the loop and proceeds to the next character. If it is a "p", the program increments the letter count.

Continue Statement: Example (Without Label)



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

        String searchMe = "peter piper picked a " + "peck of pickled peppers";
        int max = searchMe.length();
        int numPs = 0;

        for (int i = 0; i < max; i++) {
            // interested only in p's
            if (searchMe.charAt(i) != 'p')
                continue;

            // process p's
            numPs++;
        }
        System.out.println("Found " + numPs + " p's in the string.");
    }
}
```

Continue Statement: Example (Without Label)



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

        String searchMe = "Look for a substring in me";
        String substring = "sub";
        boolean foundIt = false;

        int max = searchMe.length() -
                  substring.length();

    test:
        for (int i = 0; i <= max; i++) {
            int n = substring.length();
            int j = i;
            int k = 0;
            while (n-- != 0) {
                if (searchMe.charAt(j++) != substring.charAt(k++)) {
                    continue test;
                }
            }
            foundIt = true;
                break test;
        }
        System.out.println(foundIt ? "Found it" : "Didn't find it");
    }
}
```

Branching Statements: Break, Continue and Return…

Return:

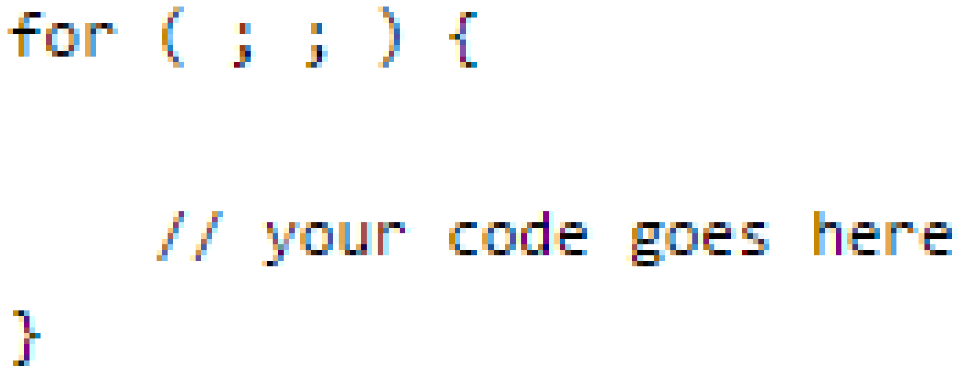
The last of the branching statements is the return statement. The return statement exits from the current method, and control flow returns to where the method was invoked. The return statement has two forms: one that returns a value, and one that doesn't. To return a value, simply put the value (or an expression that calculates the value) after the return keyword.

```
return ++count;
```

The data type of the returned value must match the type of the method's declared return value. When a method is declared void, use the form of return that doesn't return a value.

```
return;
```

Exercise: What is the output?



```
for ( ; ; ) {

    // your code goes here
}
```

Exercise

Consider the following code snippet.

```
if (aNumber >= 0)
    if (aNumber == 0)
        System.out.println("first string");
else System.out.println("second string");
System.out.println("third string");
```

- What output do you think the code will produce if aNumber is 3?
- Write a test program containing the previous code snippet; make aNumber 3. What is the output of the program? Is it what you predicted? Explain why the output is what it is; in other words, what is the control flow for the code snippet?
- Using only spaces and line breaks, reformat the code snippet to make the control flow easier to understand.
- Use braces, { and }, to further clarify the code.

Exercise

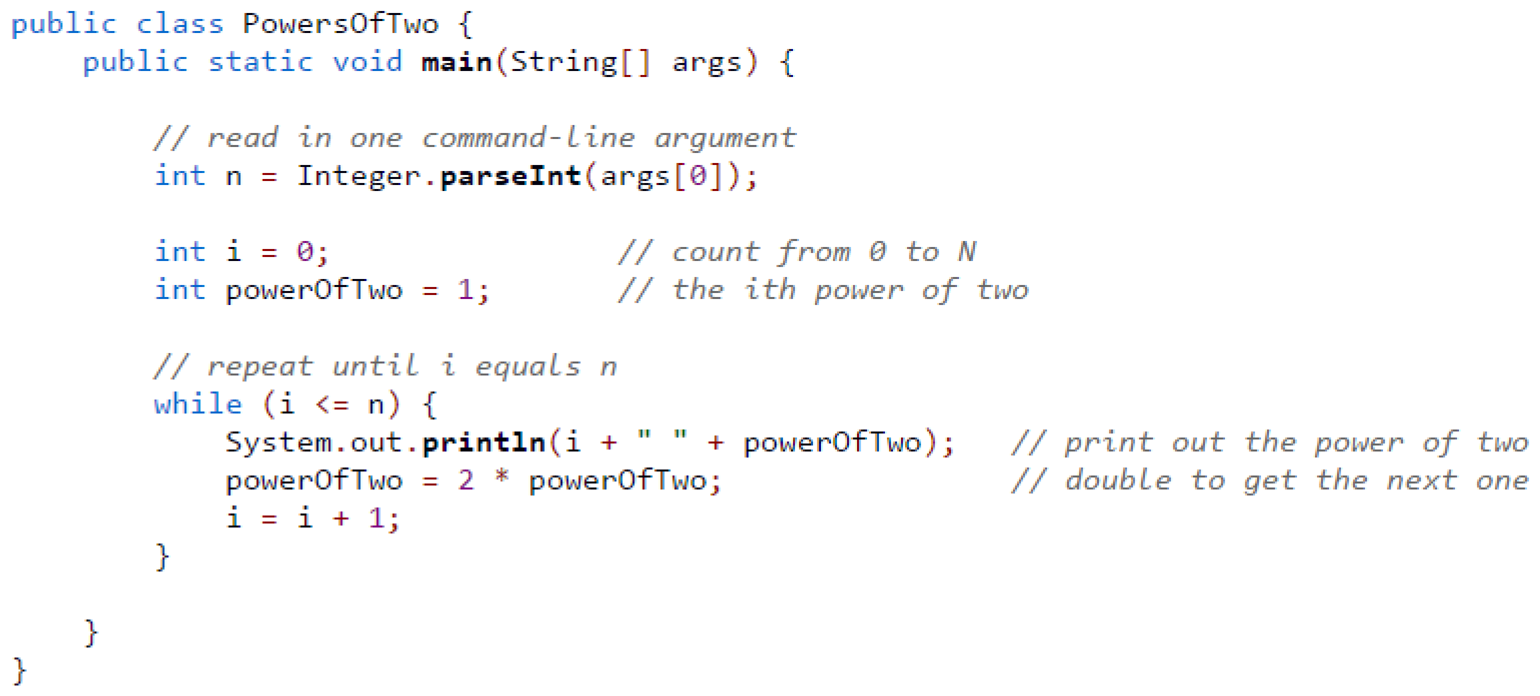
- Print 10 HelloWorld using different kinds of loops

Exercise

- Write a program to print all powers of 2 less than or equal to n.

Exercise

- Write a program to print all powers of 2 less than or equal to n.



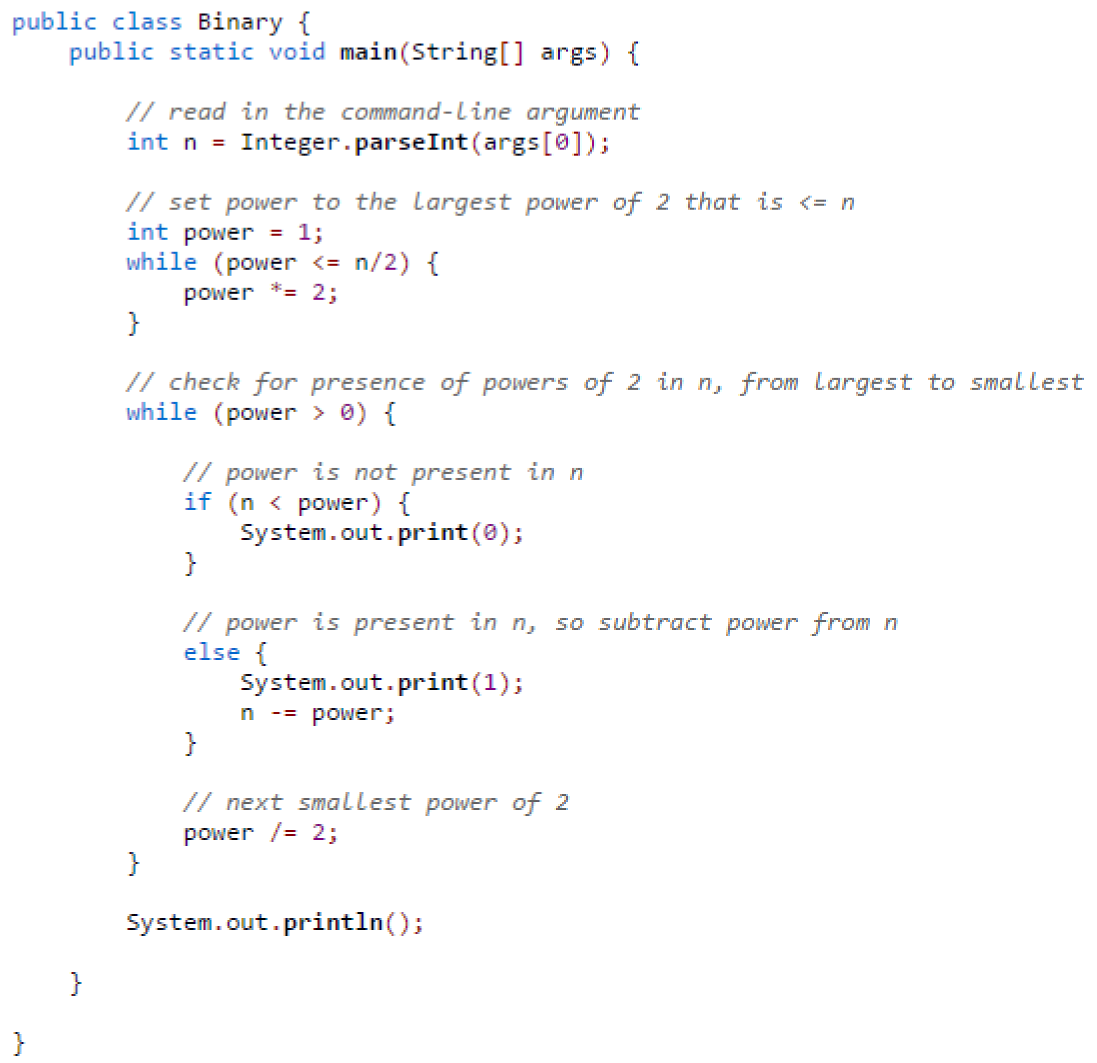
```java
public class PowersOfTwo {
    public static void main(String[] args) {

        // read in one command-line argument
        int n = Integer.parseInt(args[0]);

        int i = 0;                // count from 0 to N
        int powerOfTwo = 1;       // the ith power of two

        // repeat until i equals n
        while (i <= n) {
            System.out.println(i + " " + powerOfTwo);   // print out the power of two
            powerOfTwo = 2 * powerOfTwo;                // double to get the next one
            i = i + 1;
        }

    }
}
```

Exercise

Write a program to print the binary equivalent of a decimal integer input by the user.

Exercise

Write a program to print the binary equivalent of a decimal integer input by the user.

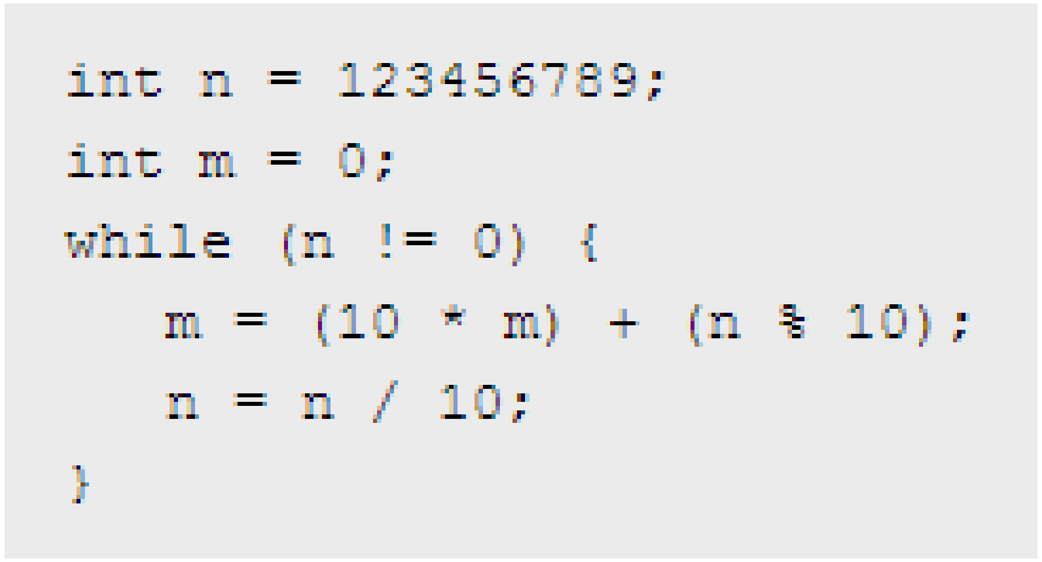


```java
public class Binary {
    public static void main(String[] args) {

        // read in the command-line argument
        int n = Integer.parseInt(args[0]);

        // set power to the largest power of 2 that is <= n
        int power = 1;
        while (power <= n/2) {
            power *= 2;
        }

        // check for presence of powers of 2 in n, from largest to smallest
        while (power > 0) {

            // power is not present in n
            if (n < power) {
                System.out.print(0);
            }

            // power is present in n, so subtract power from n
            else {
                System.out.print(1);
                n -= power;
            }

            // next smallest power of 2
            power /= 2;
        }

        System.out.println();

    }
}
```

Exercise

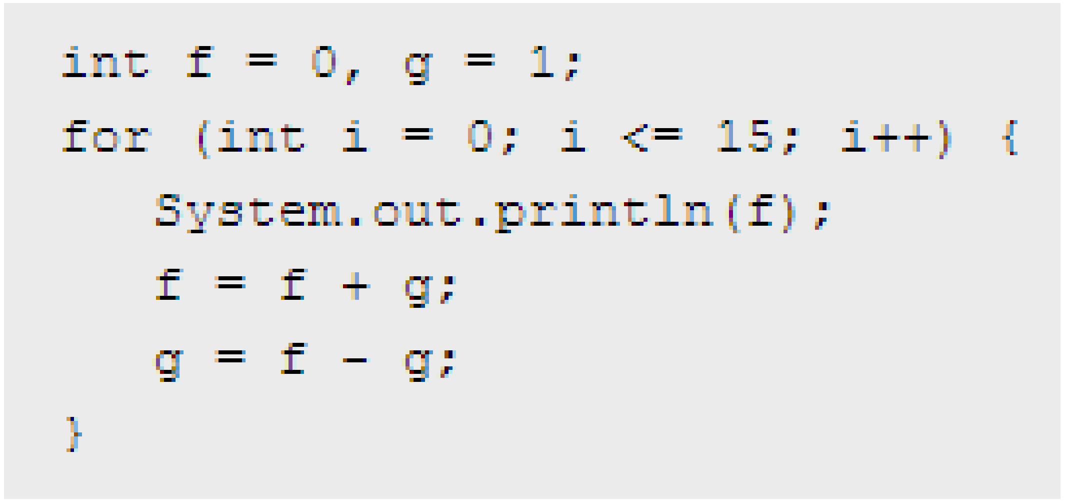
Use conditional operator (ternary operator) to find the least of 3 numbers input by the user.

Exercise: What is the value of m and n?



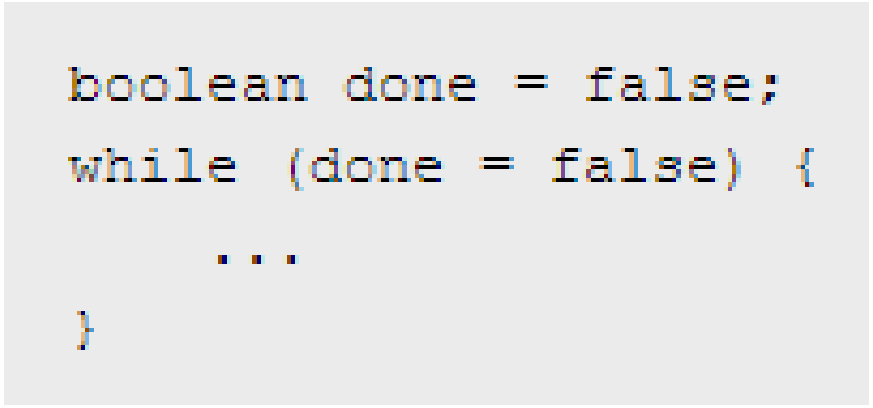
```
int n = 123456789;
int m = 0;
while (n != 0) {
    m = (10 * m) + (n % 10);
    n = n / 10;
}
```

Exercise: What is the output?



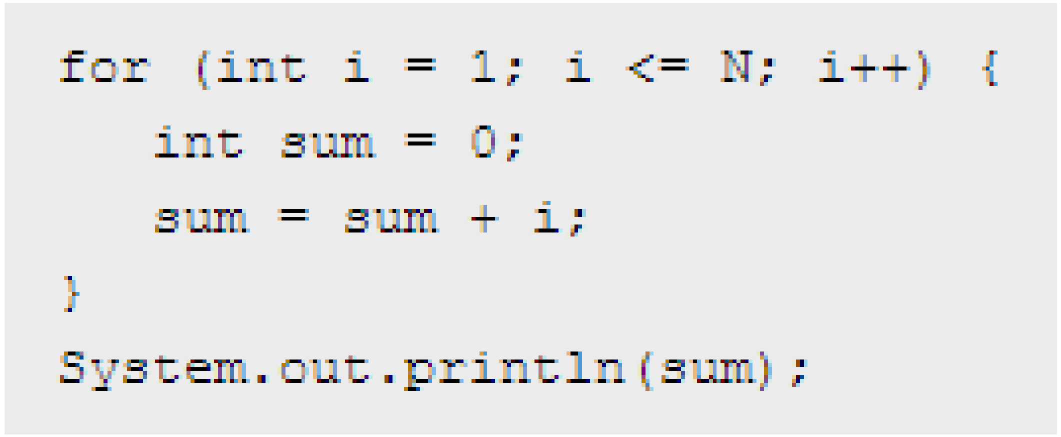
```
int f = 0, g = 1;
for (int i = 0; i <= 15; i++) {
   System.out.println(f);
   f = f + g;
   g = f - g;
}
```

Exercise: What’s wrong with this?



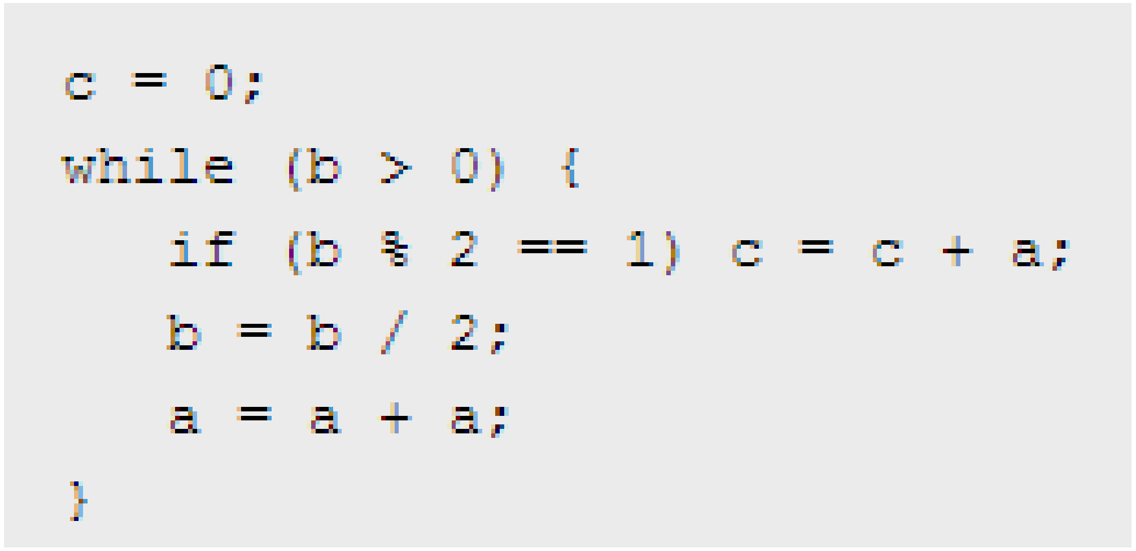
```
boolean done = false;
while (done = false) {
      ...
}
```

Exercise: What is the output?



```
for (int i = 1; i <= N; i++) {
    int sum = 0;
    sum = sum + i;
}
System.out.println(sum);
```

Exercise: What is the output?



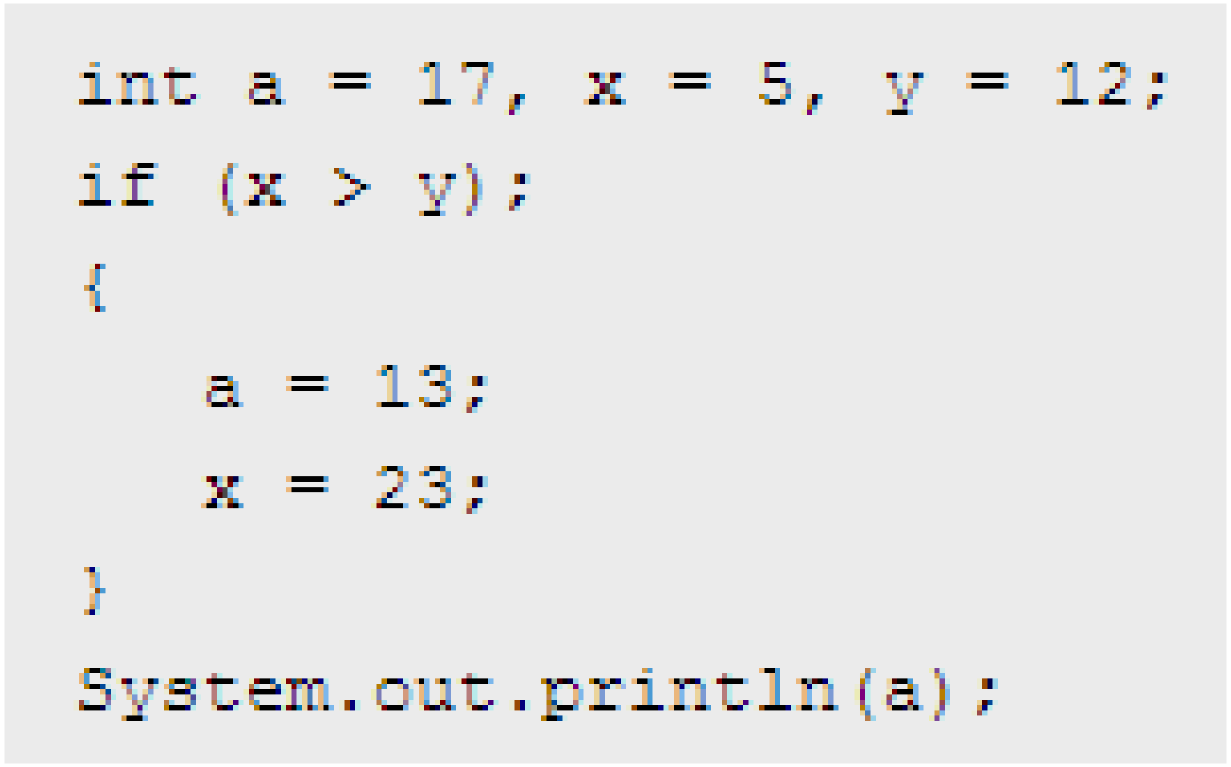
```
c = 0;
while (b > 0) {
    if (b % 2 == 1) c = c + a;
    b = b / 2;
    a = a + a;
}
```

Exercise: What is the output?

```
c = 0;
while (b > 0) {
    if (b % 2 == 1) c = c + a;
    b = b / 2;
    a = a + a;
}
```

Answer: a*b

Exercise: What is the output?



```
int a = 17, x = 5, y = 12;
if (x > y);
{
    a = 13;
    x = 23;
}
System.out.println(a);
```