While Loops & Collections

Camille Duquesne
1ère IB

Any questions/difficulties you would like to share about last session’s content ?

B2 Programming

B2.2.2 Construct programs that apply arrays and Lists.

B2.3.3 Construct programs that utilize looping structures to perform repeated actions.

- Types of loops, including counted loops and conditional loops, and appropriate use of each type
- Conditional statements within loops, using Boolean and/or relational operators to govern the loop’s execution

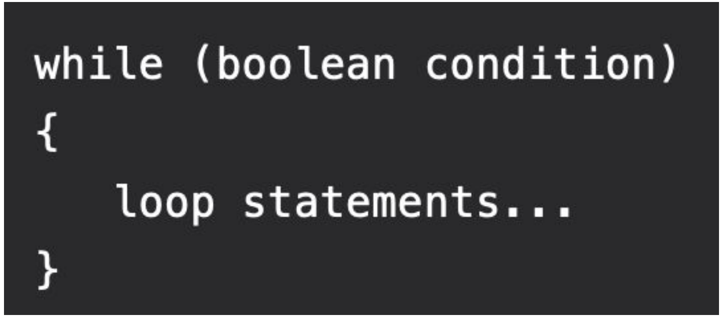
B2.3.1 Construct programs that implement the correct sequence of code instructions to meet program objectives.

- The impact of instruction order on program functionality
- Ways to avoid errors, such as infinite loops, deadlock, incorrect output

While loops

While loops *(or pre condition loop)* are used when the number of iterations/repetitions are not known before the execution of code. They are usually useful for **validation** or **verification** purposes.

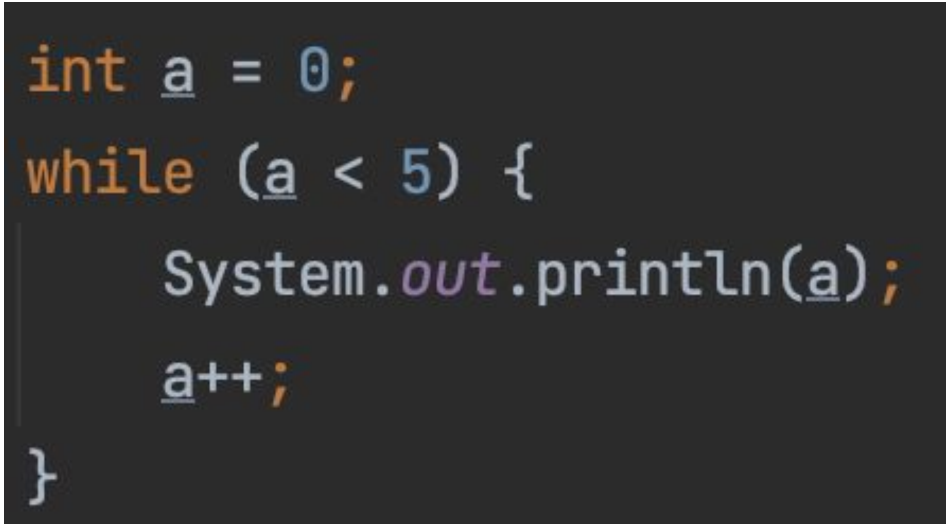
- A while loop starts with the checking of condition (called **termination condition**). If it evaluated to true, then the loop body statements are executed otherwise the statements following the loop are executed.
- Once the condition is evaluated to true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the next iteration.
- When the condition becomes false, the loop terminates.



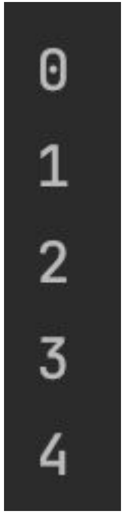
```
while (boolean condition)
{
    loop statements...
}
```

Be careful to have a boolean condition that turns false at some point otherwise you have an infinity loop !!!

While loop example



```java
int a = 0;
while (a < 5) {
    System.out.println(a);
    a++;
}
```

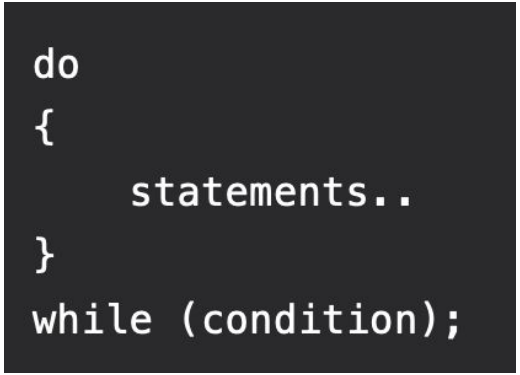


```
0
1
2
3
4
```

What happens if I change my condition to `(a < 0)` ?

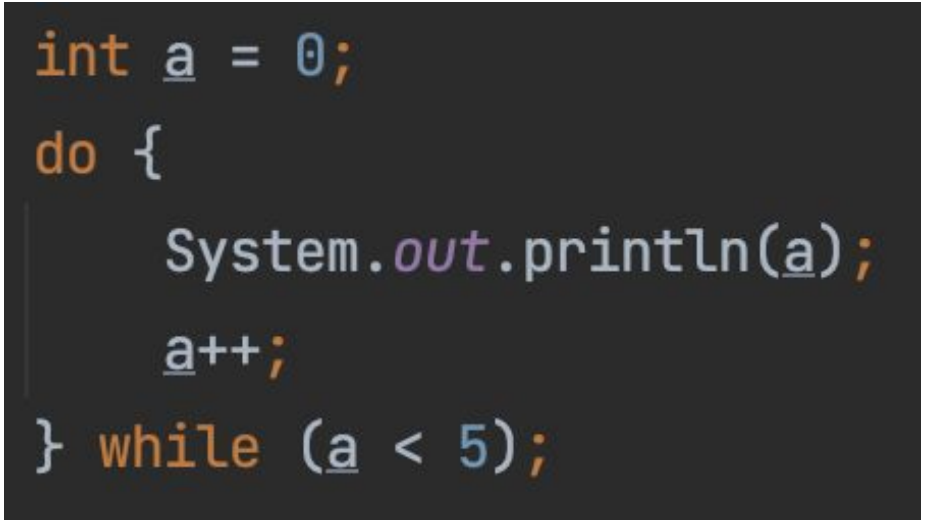
Do while loop

- A do while loop *(post condition loop)* starts with the execution of the statement(s). There is no checking of any condition for the first iteration.
- After the execution of the statements (and any updates to variable values), the termination condition is checked. If it is evaluated to true, the next iteration of the loop starts.
- When the condition becomes false, the loop terminates.
- It is important to note that the do-while loop will execute its statements at least once before any condition is checked !

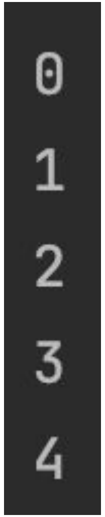


```
do
{
    statements..
}
while (condition);
```

Do while loop example



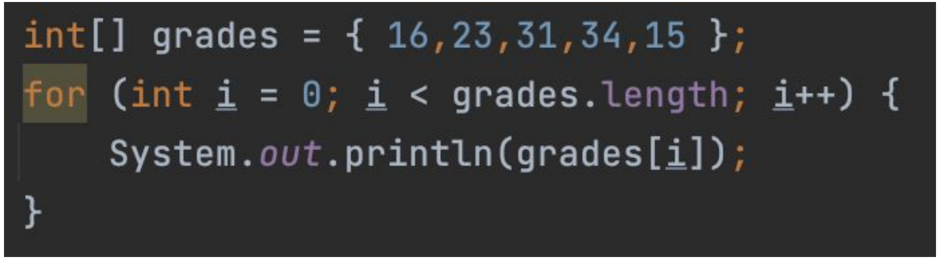
```java
int a = 0;
do {
    System.out.println(a);
    a++;
} while (a < 5);
```



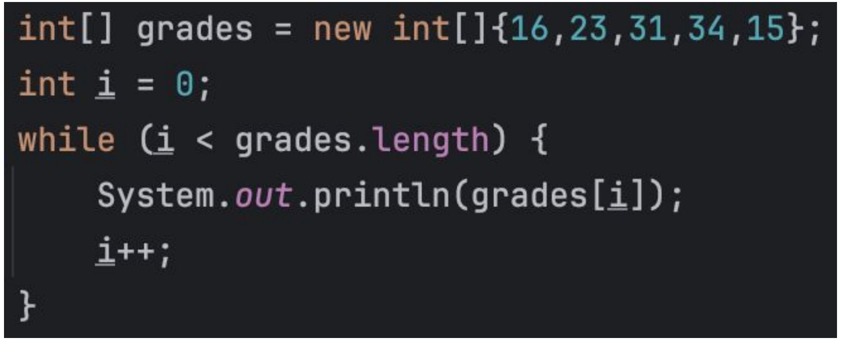
```
0
1
2
3
4
```

What happens if I change my condition to `(a < 0)` ?

Can I rewrite a for loop as a while loop ?



```
int[] grades = { 16,23,31,34,15 };
for (int i = 0; i < grades.length; i++) {
    System.out.println(grades[i]);
}
```



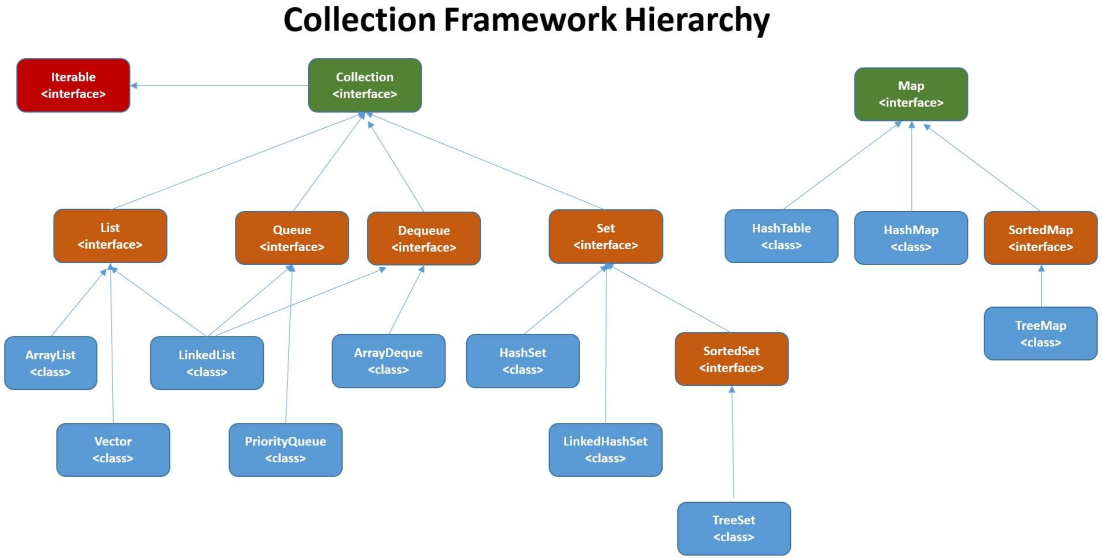
```
int[] grades = new int[]{16,23,31,34,15};
int i = 0;
while (i < grades.length) {
    System.out.println(grades[i]);
    i++;
}
```

Can I rewrite all while loops as for loops as well ?

The collections framework

The collections framework defines different data types in order to work with collections of elements (= a group of elements together)

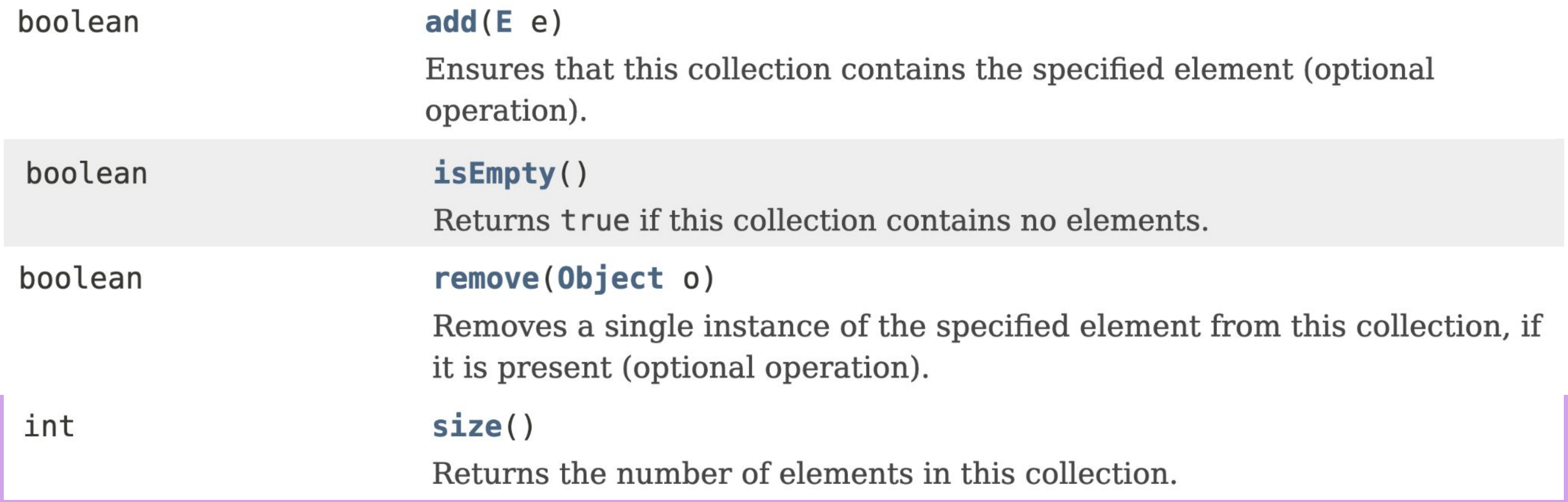
In HL we'll get to understand most of these collections and their specificities.



The collections interface

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

If we look at the documentation of the collection interface we can sport a few methods that will be very useful:

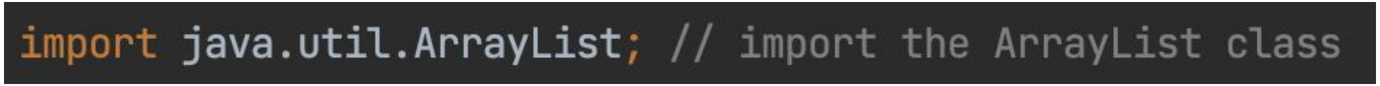


boolean	add(E e) Ensures that this collection contains the specified element (optional operation).
boolean	isEmpty() Returns true if this collection contains no elements.
boolean	remove(Object o) Removes a single instance of the specified element from this collection, if it is present (optional operation).
int	size() Returns the number of elements in this collection.

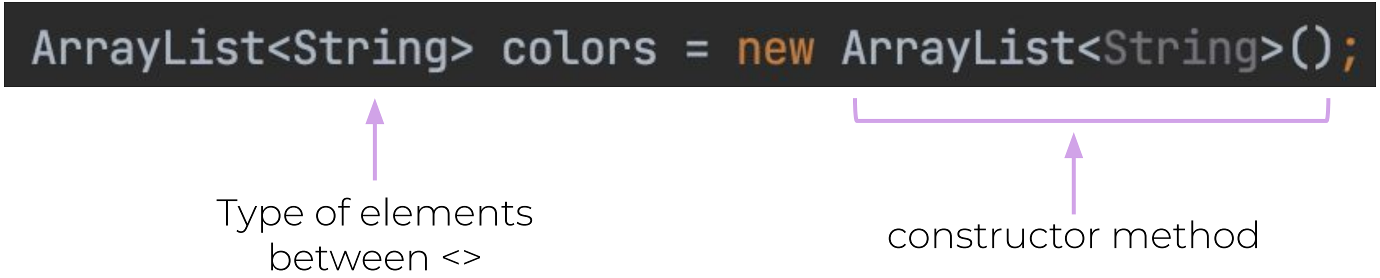
What are ArrayList ?

An ArrayList is a **resizable collection** of elements of the same type. They are defined in the `java.util` package.

To construct an empty ArrayList we do the following:



```java
import java.util.ArrayList; // import the ArrayList class
```



```java
ArrayList<String> colors = new ArrayList<String>();
```

Type of elements between <>

constructor method

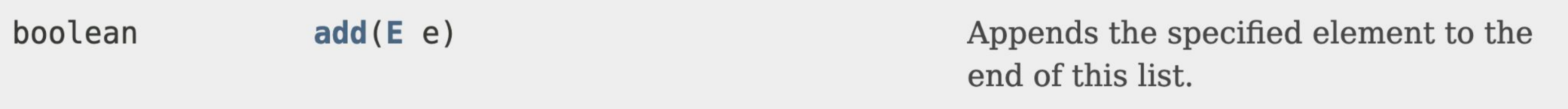
What are useful ArrayList methods ?

Let’s check the documentation to see what methods are available:

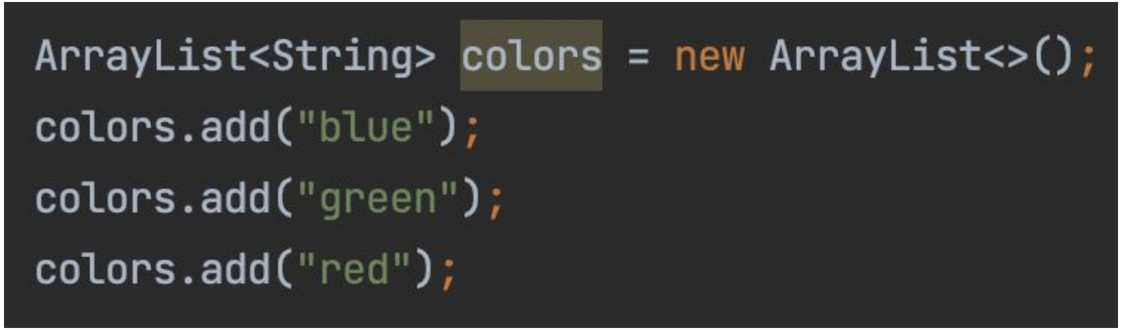
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ArrayList.html

Adding elements to my ArrayList

After reading the documentation we find the add() method to populate our ArrayList:



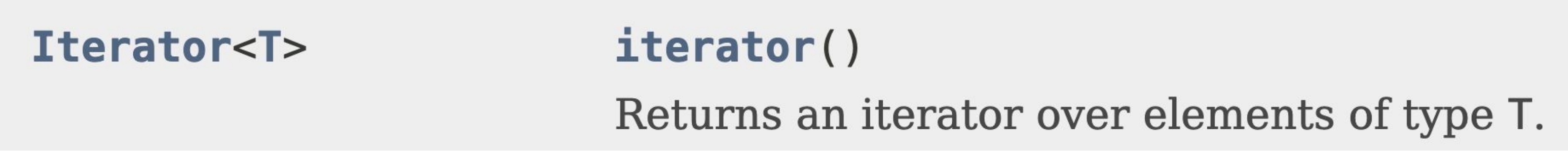
boolean	add(E e)	Appends the specified element to the end of this list.



```java
ArrayList<String> colors = new ArrayList<>();
colors.add("blue");
colors.add("green");
colors.add("red");
```

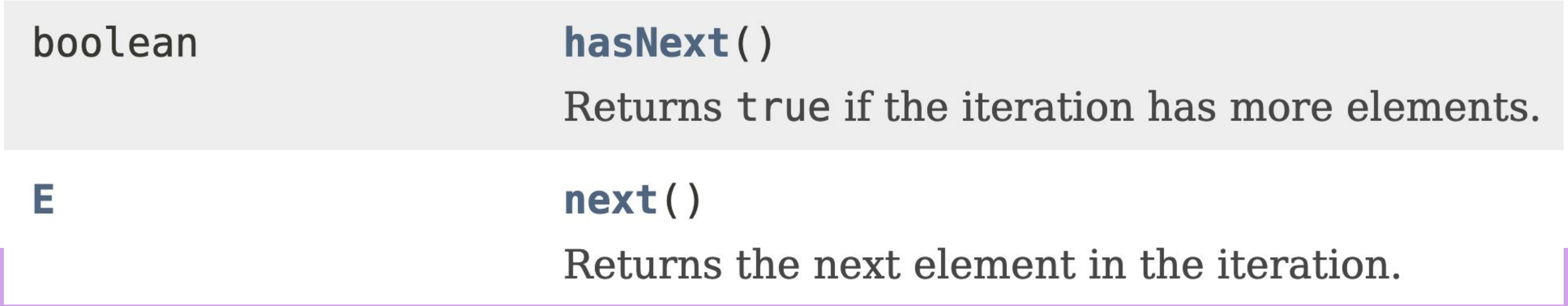
How do I loop over my collection ?

All collections extend the Iterable Interface, which allows these collections to be iterated on, for example in a for loop.



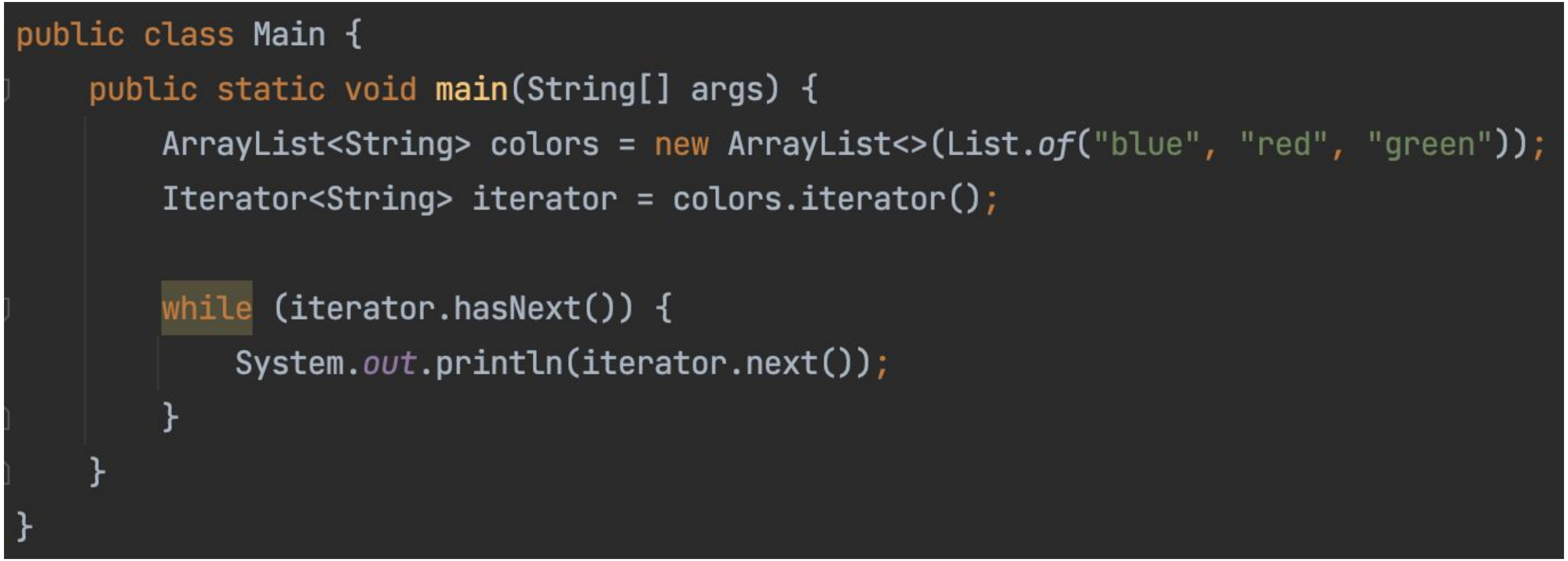
`Iterator<T>`	`iterator()` Returns an iterator over elements of type T.

The iterator object in turn has two methods that will be extremely important for us:

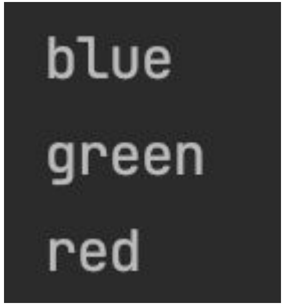


`boolean`	`hasNext()` Returns `true` if the iteration has more elements.
`E`	`next()` Returns the next element in the iteration.

Iterator Example

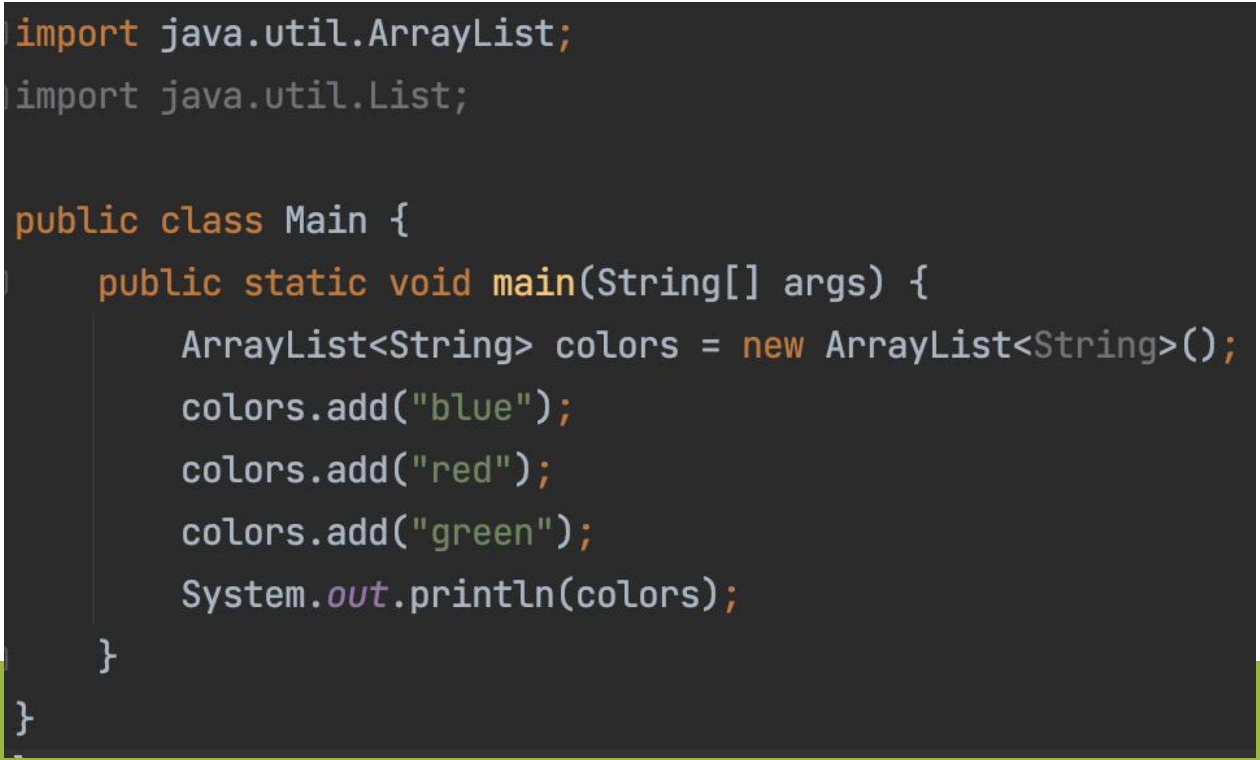


```java
public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<>(List.of("blue", "red", "green"));
        Iterator<String> iterator = colors.iterator();

        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```



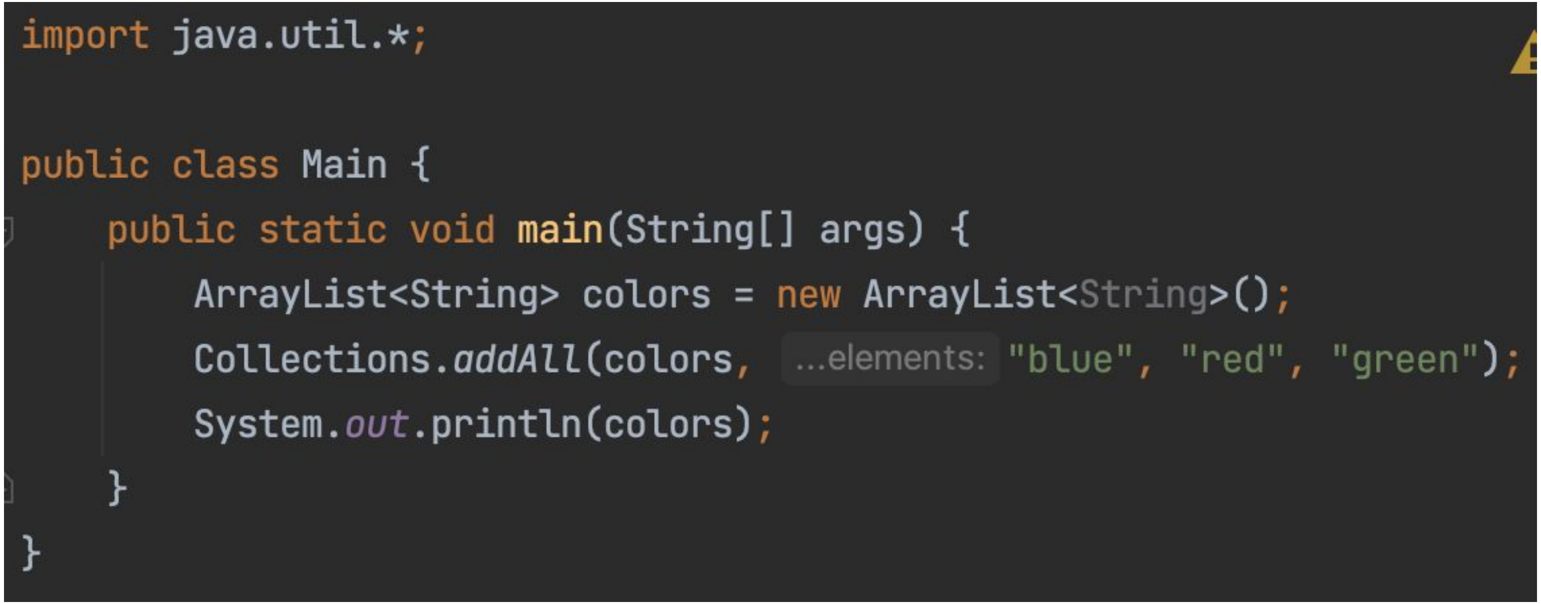
```
blue
green
red
```

What does this code output ?



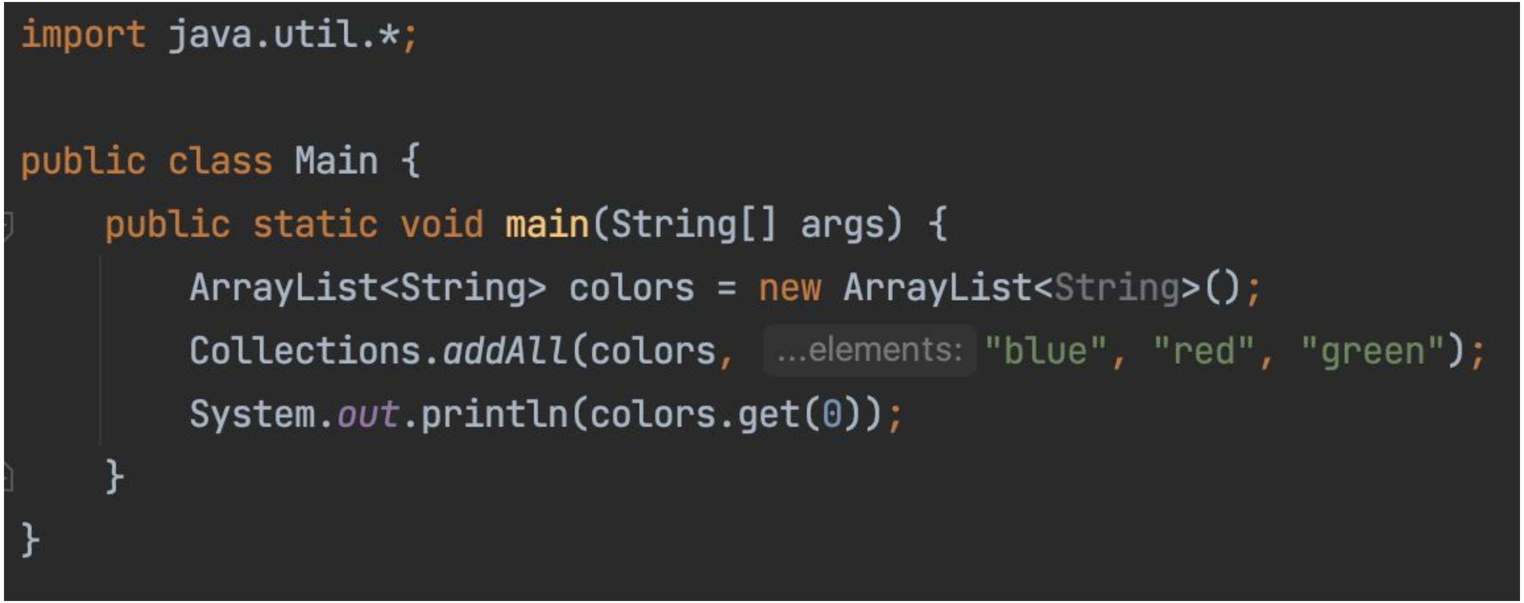
```java
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<String>();
        colors.add("blue");
        colors.add("red");
        colors.add("green");
        System.out.println(colors);
    }
}
```

What does this code output ?



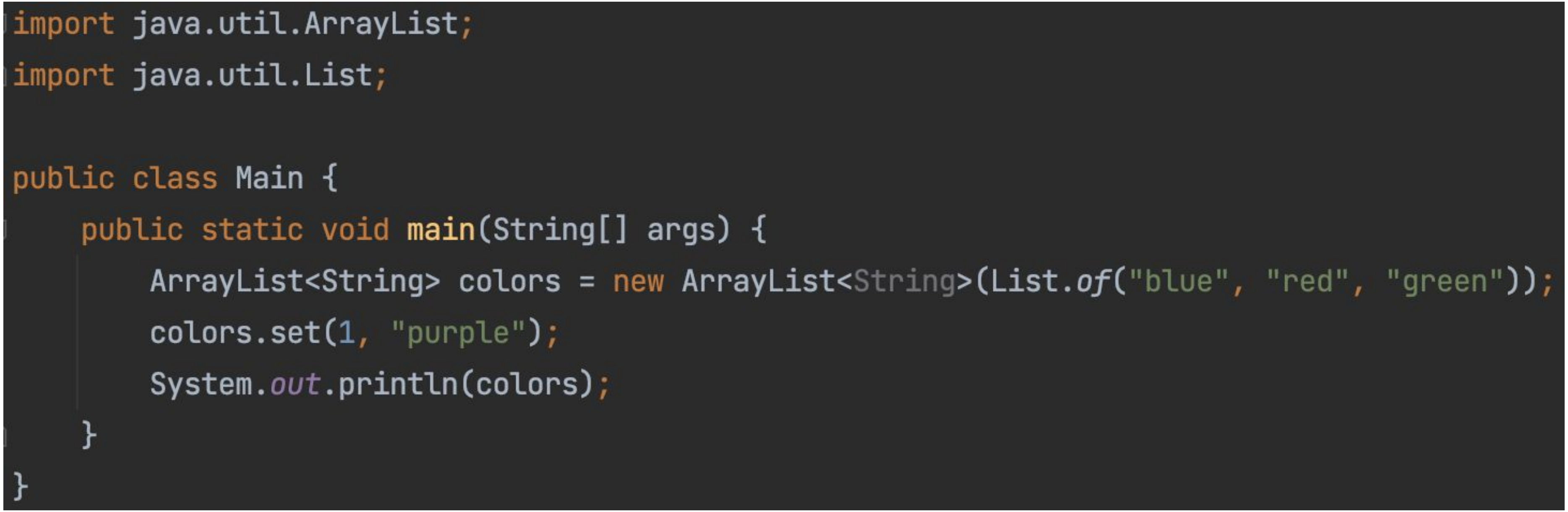
```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<String>();
        Collections.addAll(colors, ...elements: "blue", "red", "green");
        System.out.println(colors);
    }
}
```

What does this code output ?



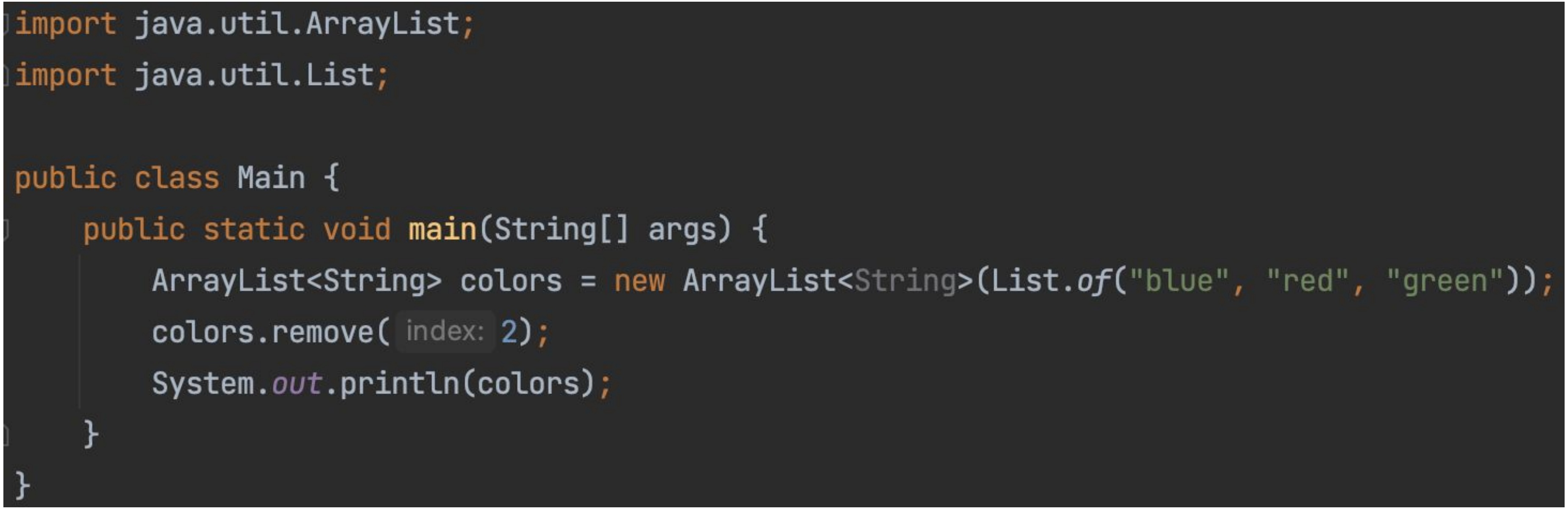
```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<String>();
        Collections.addAll(colors, ...elements: "blue", "red", "green");
        System.out.println(colors.get(0));
    }
}
```

What does this code output ?



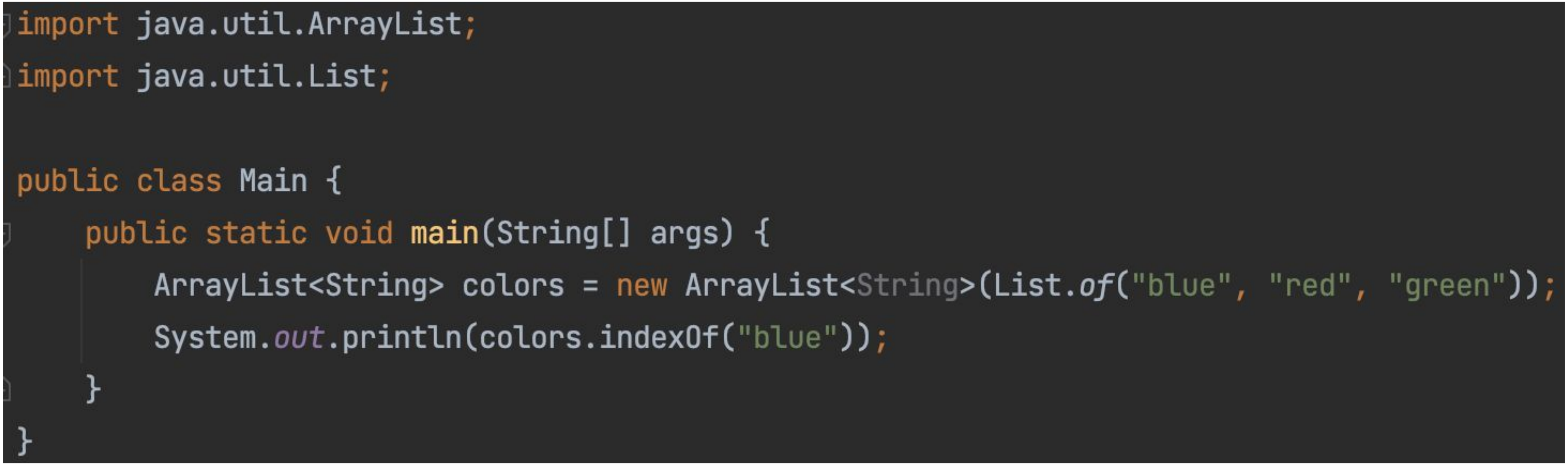
```java
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<String>(List.of("blue", "red", "green"));
        colors.set(1, "purple");
        System.out.println(colors);
    }
}
```

What does this code output ?



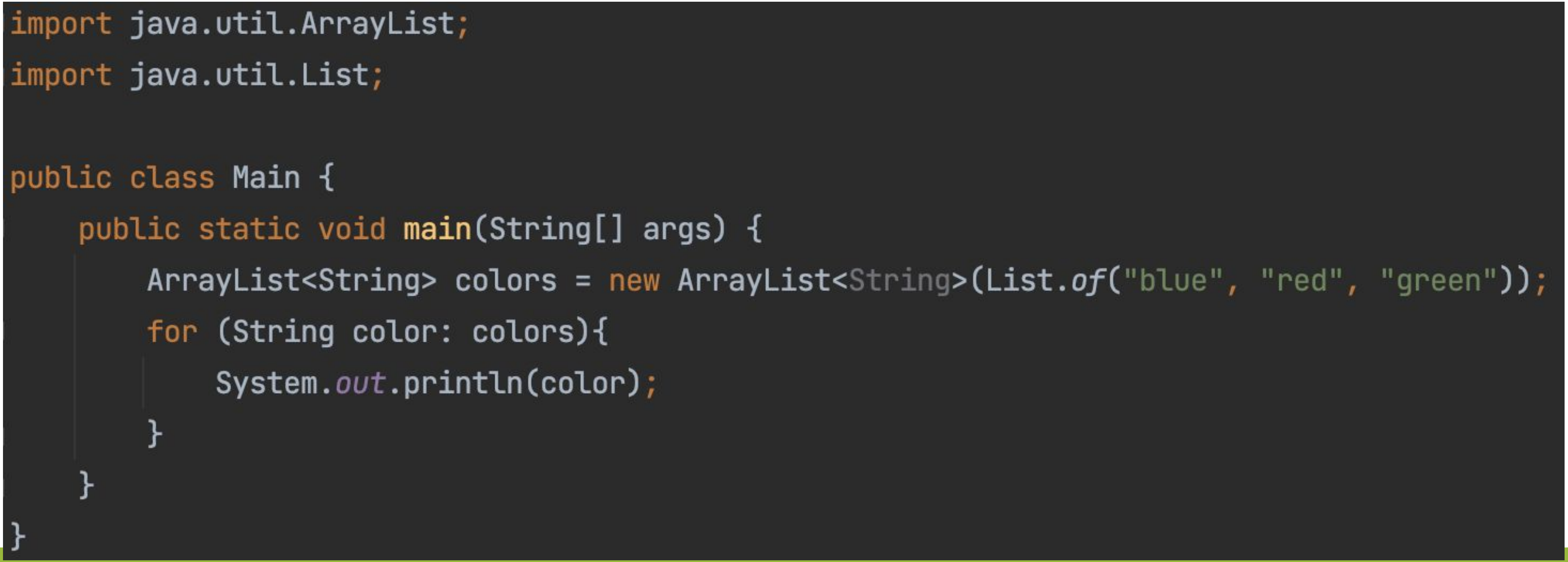
```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<String>(List.of("blue", "red", "green"));
        colors.remove( index: 2);
        System.out.println(colors);
    }
}
```

What does this code output ?



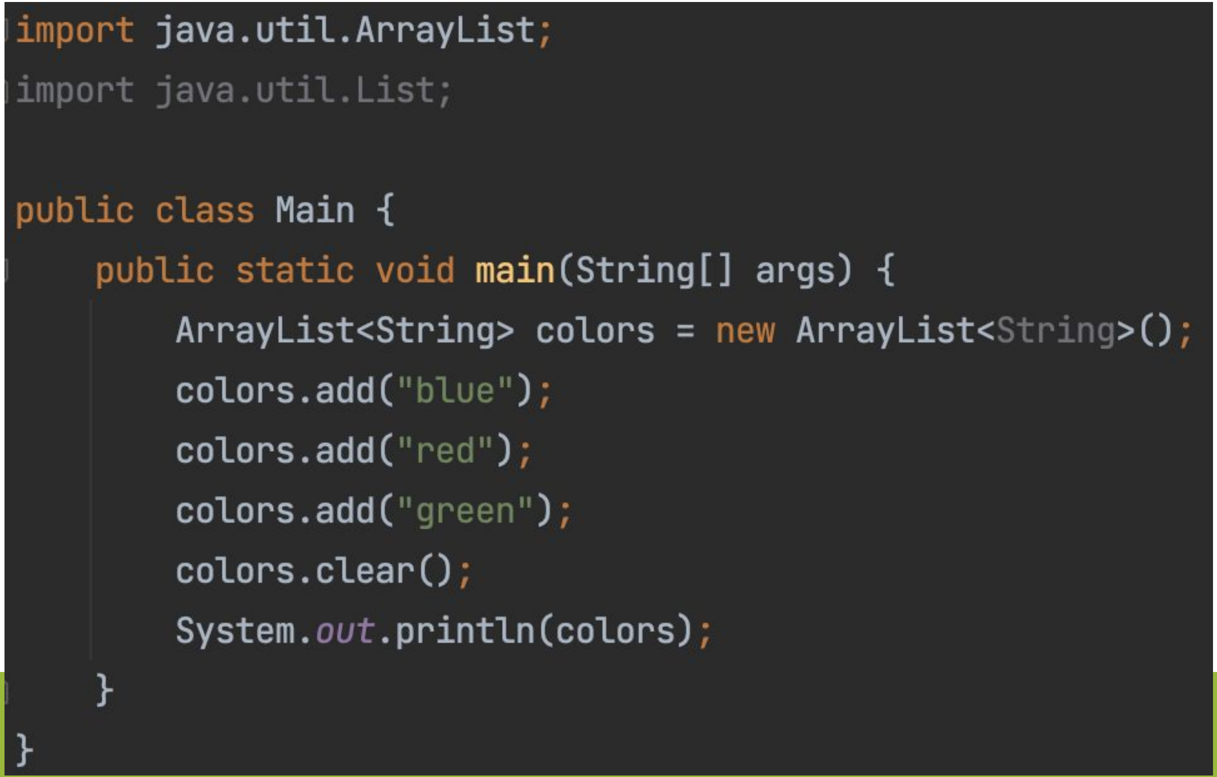
```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<String>(List.of("blue", "red", "green"));
        System.out.println(colors.indexOf("blue"));
    }
}
```

What does this code output ?



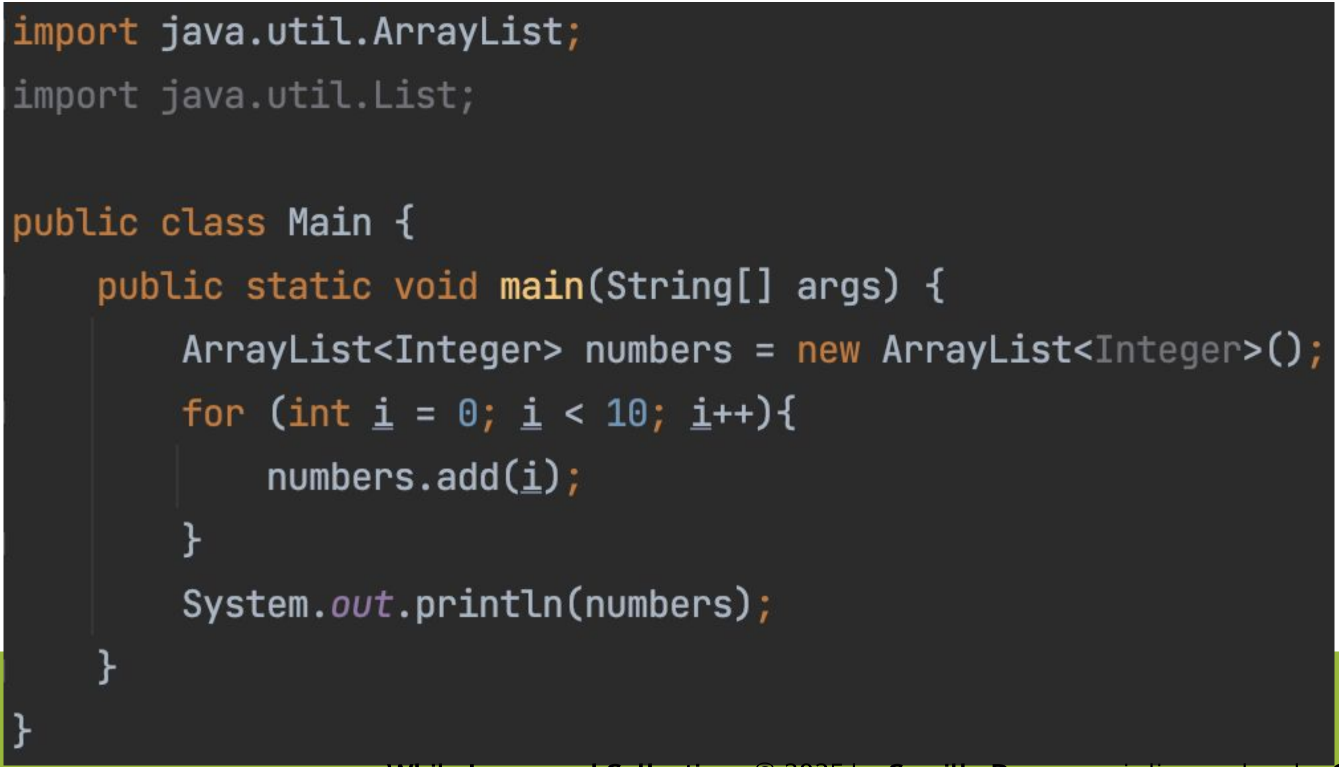
```java
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<String>(List.of("blue", "red", "green"));
        for (String color: colors){
            System.out.println(color);
        }
    }
}
```

What does this code output ?



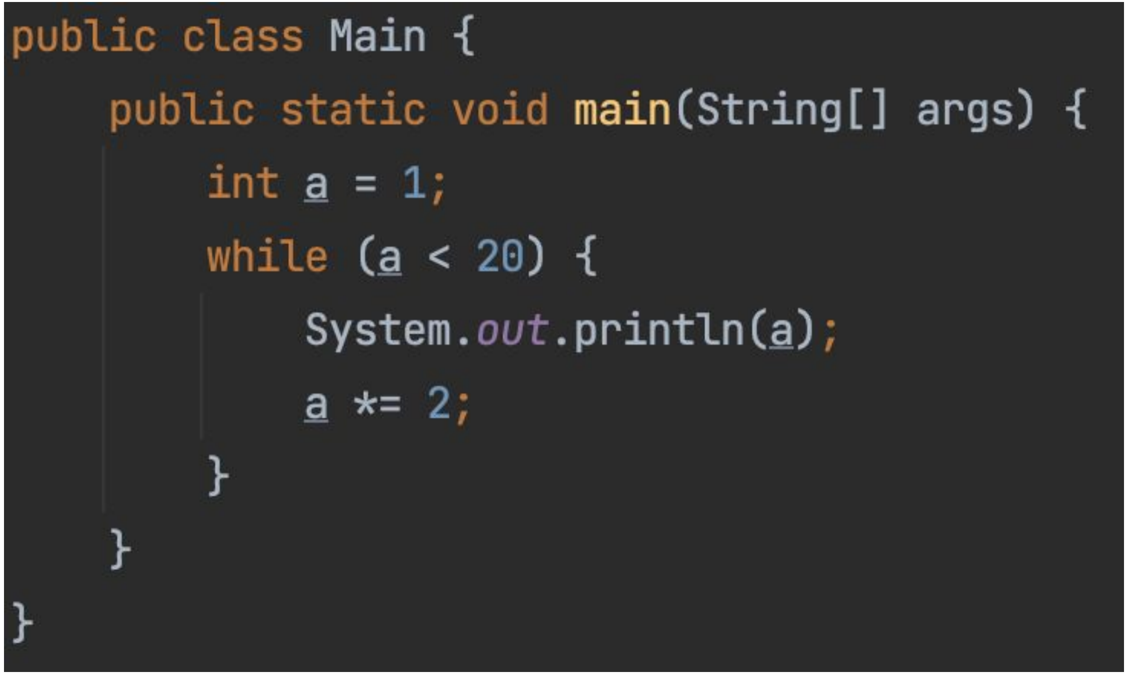
```java
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<String>();
        colors.add("blue");
        colors.add("red");
        colors.add("green");
        colors.clear();
        System.out.println(colors);
    }
}
```

What does this code output ?



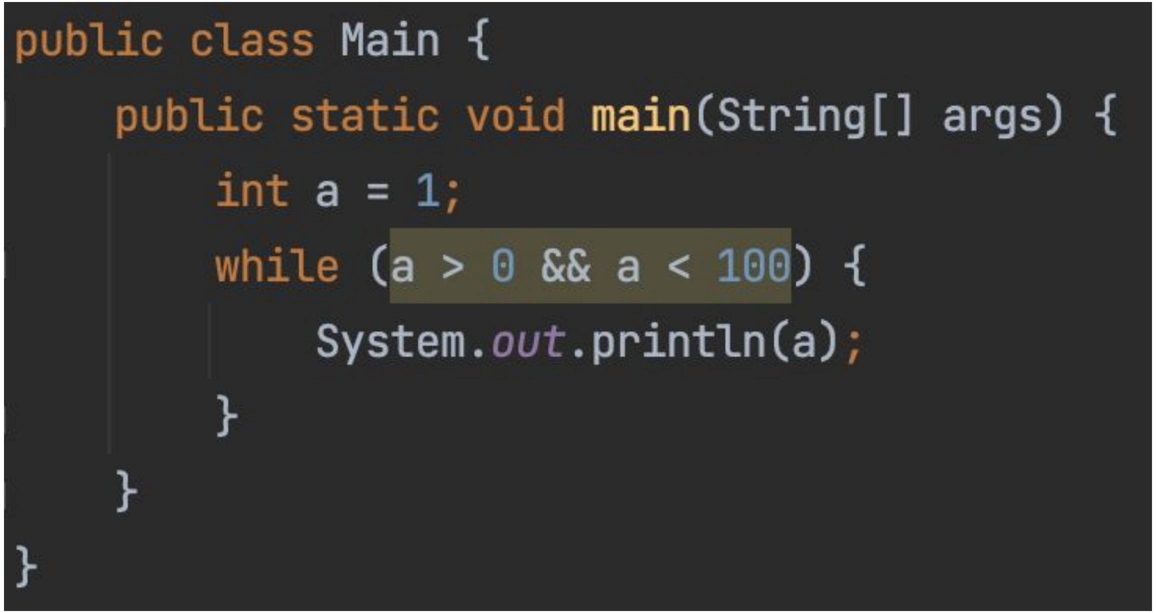
```java
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<Integer>();
        for (int i = 0; i < 10; i++){
            numbers.add(i);
        }
        System.out.println(numbers);
    }
}
```

What does this code output ?



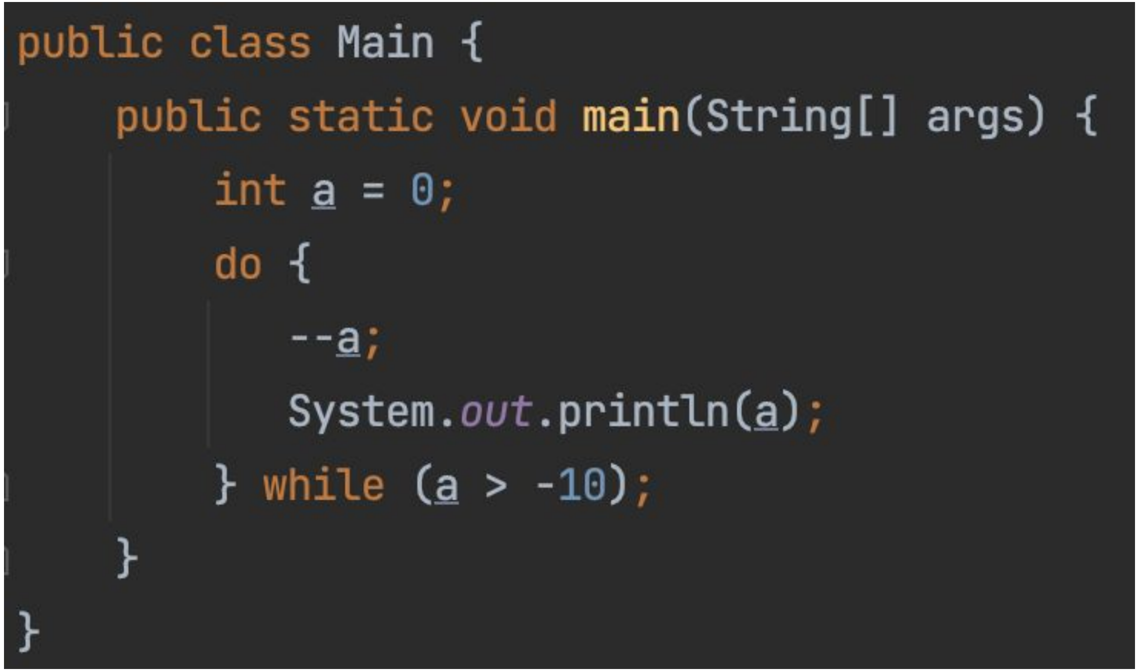
```java
public class Main {
    public static void main(String[] args) {
        int a = 1;
        while (a < 20) {
            System.out.println(a);
            a *= 2;
        }
    }
}
```

What does this code output ?



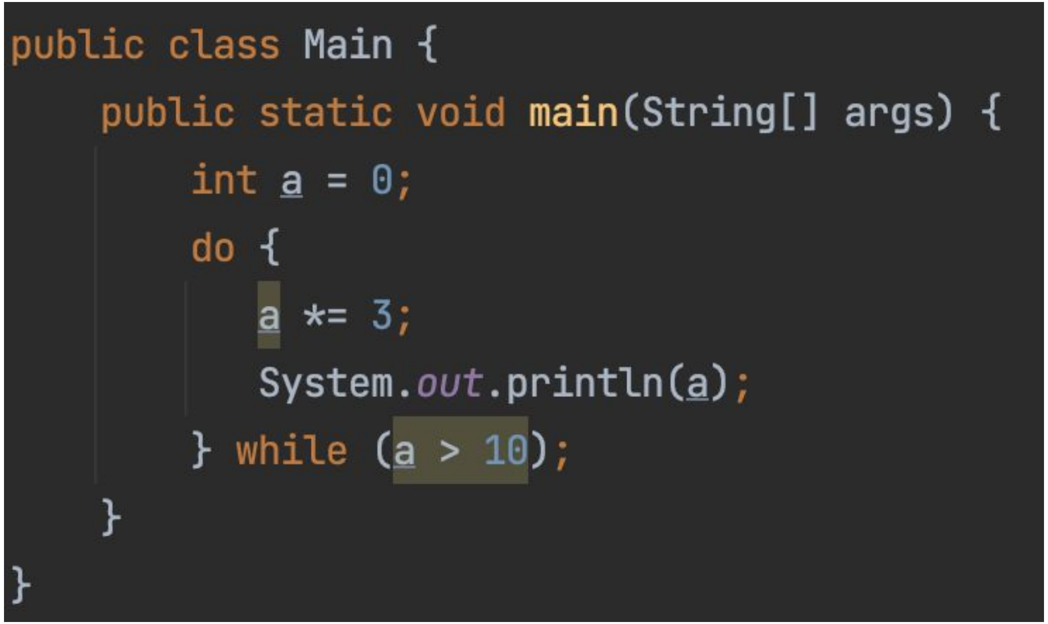
```java
public class Main {
    public static void main(String[] args) {
        int a = 1;
        while (a > 0 && a < 100) {
            System.out.println(a);
        }
    }
}
```

What does this code output ?



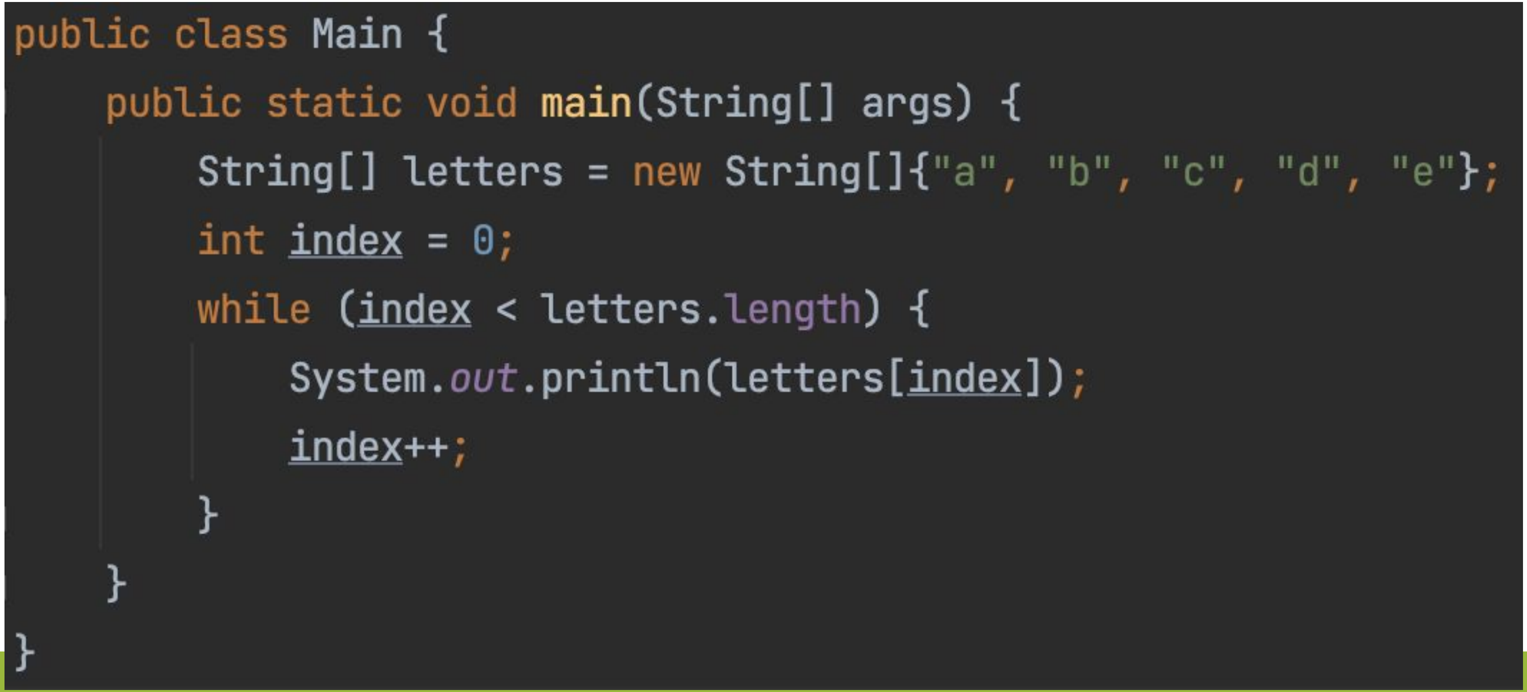
```java
public class Main {
    public static void main(String[] args) {
        int a = 0;
        do {
            --a;
            System.out.println(a);
        } while (a > -10);
    }
}
```

What does this code output ?



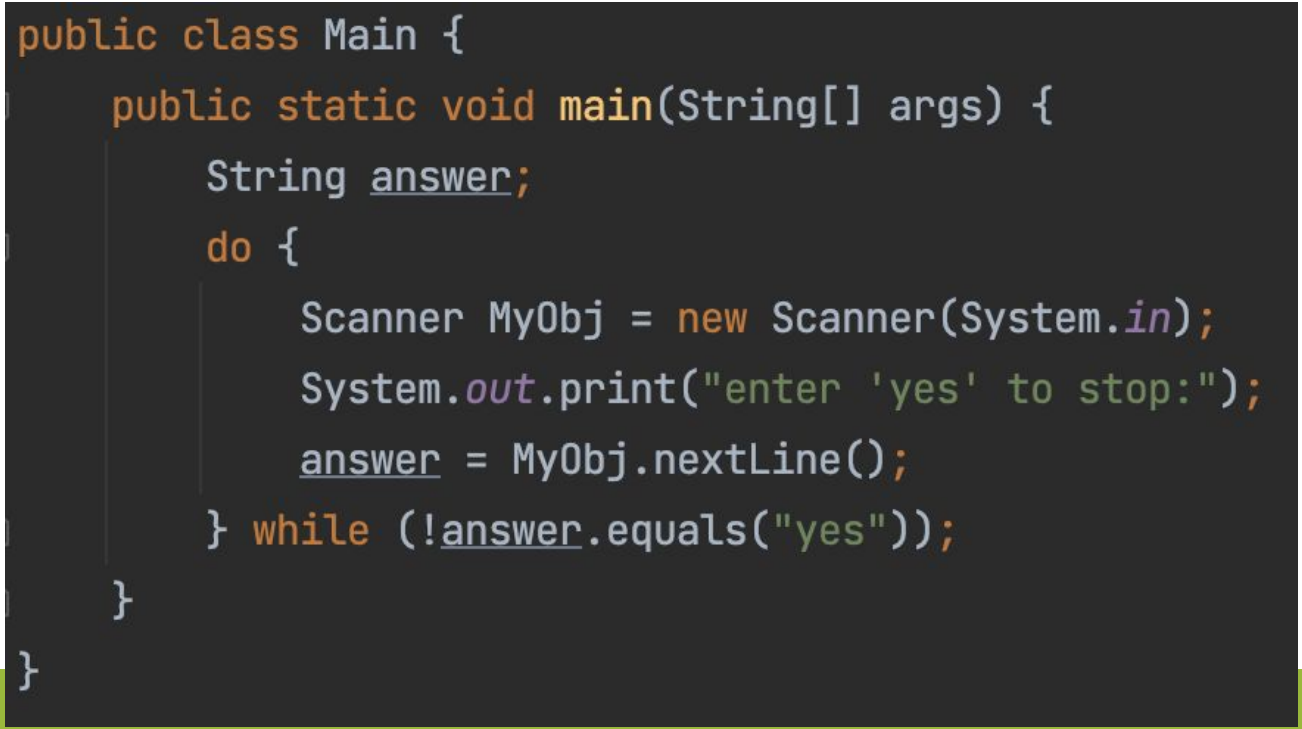
```java
public class Main {
    public static void main(String[] args) {
        int a = 0;
        do {
            a *= 3;
            System.out.println(a);
        } while (a > 10);
    }
}
```

What does this code output ?



```java
public class Main {
    public static void main(String[] args) {
        String[] letters = new String[]{"a", "b", "c", "d", "e"};
        int index = 0;
        while (index < letters.length) {
            System.out.println(letters[index]);
            index++;
        }
    }
}
```

What does this code output ?



```java
public class Main {
    public static void main(String[] args) {
        String answer;
        do {
            Scanner MyObj = new Scanner(System.in);
            System.out.print("enter 'yes' to stop:");
            answer = MyObj.nextLine();
        } while (!answer.equals("yes"));
    }
}
```

Dynamic data structures

ArrayLists are one of several data structures that are considered **Dynamic data structures.**

A dynamic data structure has:

- No predefined size
- Stored in memory locations that are chained together (but not necessary contiguous)
- Has no direct access to elements

Advantages of Dynamic Data Structures

- Flexibility -> they can grow or shrink during runtime
- Efficient use of memory -> no waste as we are always using the memory needed for the current number of elements
- Insertion and deletions are optimal -> in terms of time and space complexity (we'll study this later on)

Disadvantages of Dynamic Data Structures

- Potential memory overflow -> since elements are not contiguous in memory
- Harder to program

Pause & Recall

Created by Victoruler
from Noun Project

Close your eyes and try to recall as many things as possible that were covered during this lesson.

Alternatively, you can keep your eyes open and write down as many things you remember on a piece of paper.

This will help strengthen your memory of key concepts 💪

Exercise 1 *(GenAI 🟠 Orange)*

Create an ArrayList object of your choice and iterate over each element with the iterator() function.

Exercise 2 *(GenAI 🟠 Orange)*

Write a program that takes a String as input from the user and return an ArrayList of all the characters in the String.

For example if I input `“Camille”` I should get `['C', 'A', 'M', 'I', 'L', 'L','E']`

Exercise 3 *(GenAI 🟠 Orange)*

Write a do-while loop that asks the user to enter two numbers. The numbers should be added and the sum displayed. The loop should ask the user whether it wishes to perform the operation again. If so, the loop should repeat; otherwise it should terminate.

Exercise 4 *(GenAI 🟠 Orange)*

Write a program to enter numbers as long as the user wants and at the end the program should display the largest and smallest numbers entered.

Exercise 5 *(GenAI 🟠 Orange)*

Write a program that takes a String as input from the user and outputs each character n times, with n being the position of the character in the String.

Example input: `“CAMILLE”`

Example output:

```
C
AA
MMM
IIII
LLLLL
LLLLLL
EEEEEEE
```

Exercise 6 *(GenAI 🟠 Orange)*

Write a program that takes a String as input from the user and outputs:

- on even indexes, the character of the String
- and on odd indexes, the index number.

Example input: `“CAMILLE”`

Example output:

```
C
1
M
3
L
5
E
```

Exercise 7 *(GenAI 🟠 Orange)*

Write a program that takes a String as input from the user, and that outputs an ArrayList of all sliding pairs of the String.

Example input: `“CAMILLE”`

Example output: `['CA', 'AM', 'MI', 'IL', 'LL', 'LE']`

Exercise 8 *(GenAI 🟠 Orange)*

Write a program that takes a String as input from the user, and outputs an ArrayList of all the characters appearing in the string.

Input: `”Camillelleeaa”`

Output: `['C', 'a', 'm', 'i', 'l', 'e']`

Exercise 9 *(GenAI 🟠 Orange)*

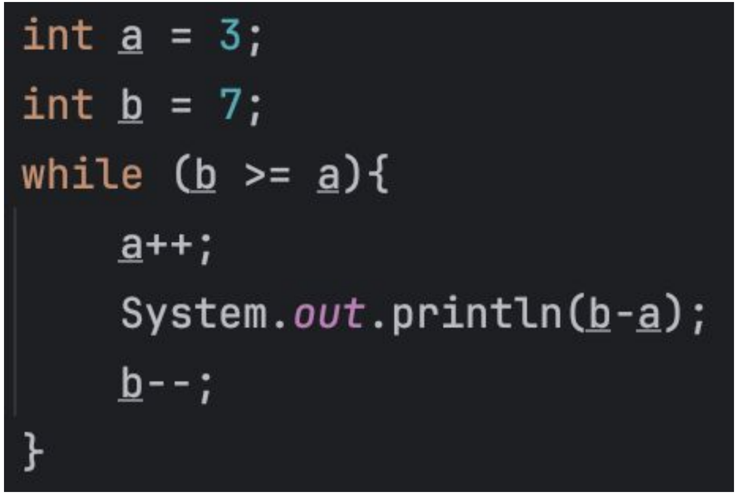
Write a program that takes as input from the user: a String and a number n. The script should return the String with the nth character removed from the string.

Exercise 10 *(GenAI 🟠 Orange)*

Write a program to enter the numbers till the user wants and at the end it should display the count of positive numbers, negative numbers and zeros entered.

Exercice 11 *(GenAI 🟠 Orange)*

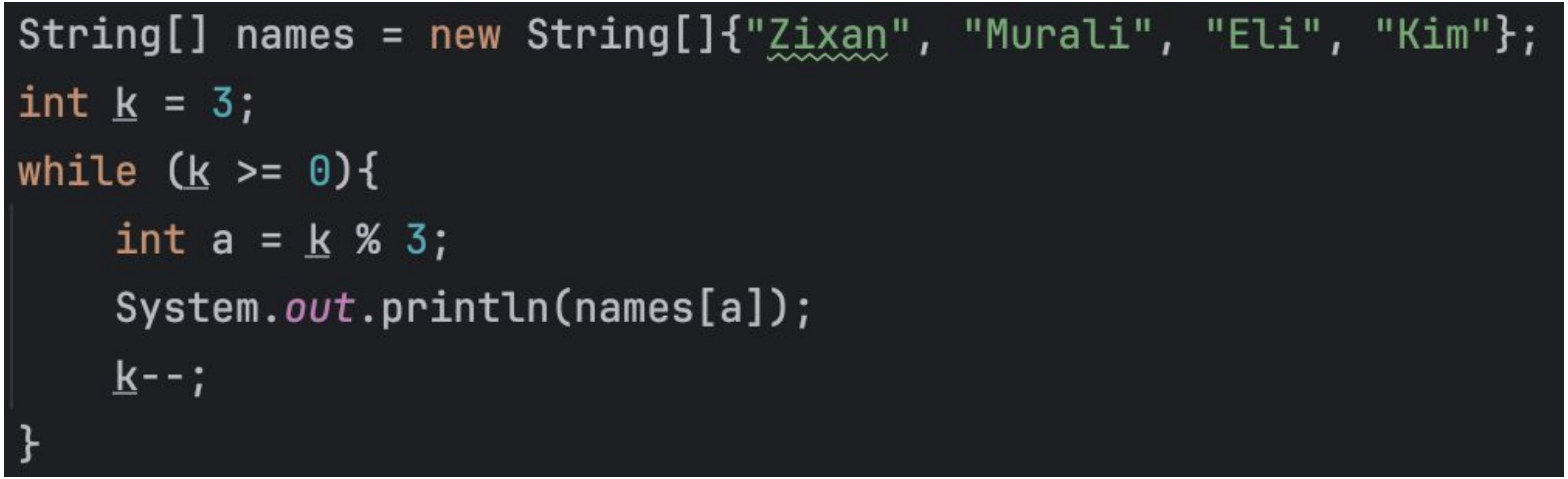
Write the tracetable of the following code



```java
int a = 3;
int b = 7;
while (b >= a){
    a++;
    System.out.println(b-a);
    b--;
}
```

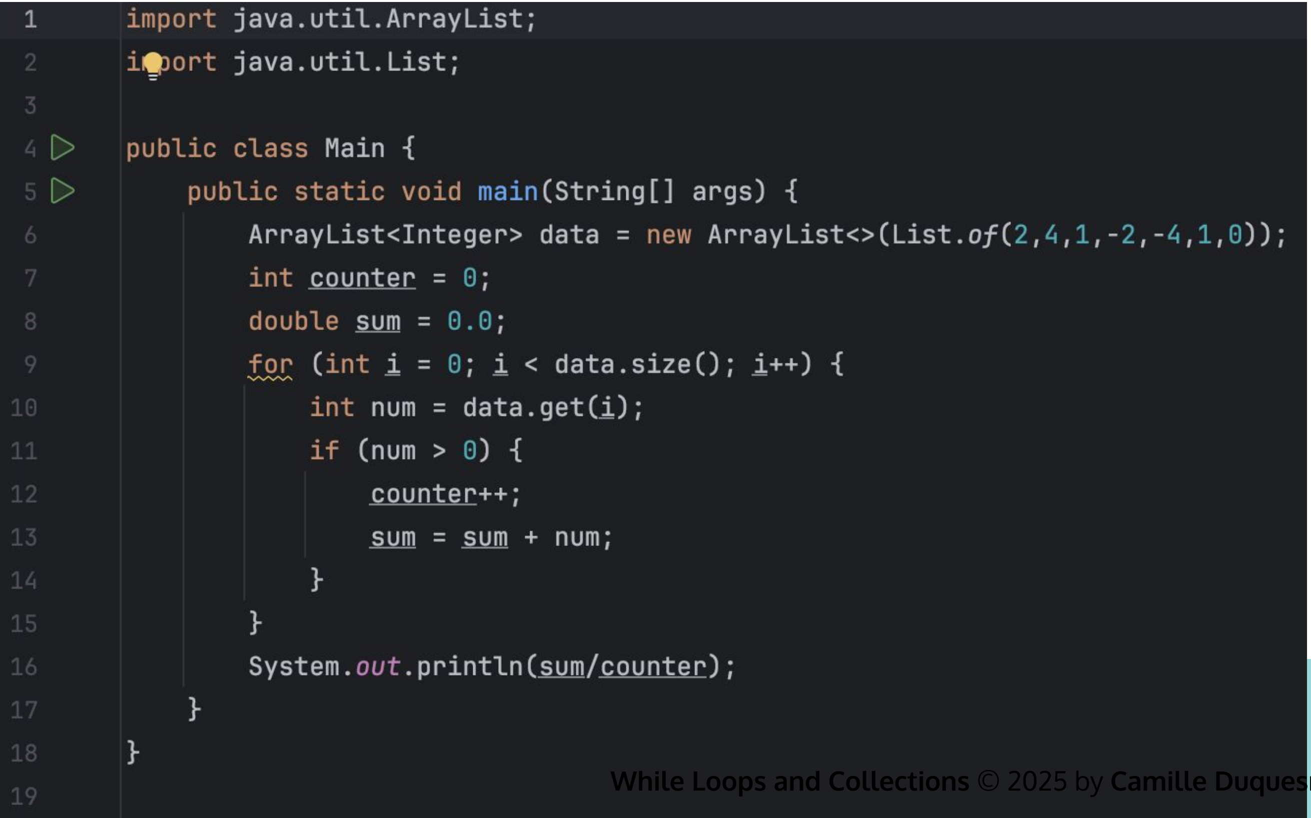
Exercice 12 *(GenAI 🟠 Orange)*

Write the tracetable of the following code



```java
String[] names = new String[]{"Zixan", "Murali", "Eli", "Kim"};
int k = 3;
while (k >= 0){
     int a = k % 3;
     System.out.println(names[a]);
     k--;
}
```

Exercice 13 *(GenAI 🟠 Orange)*



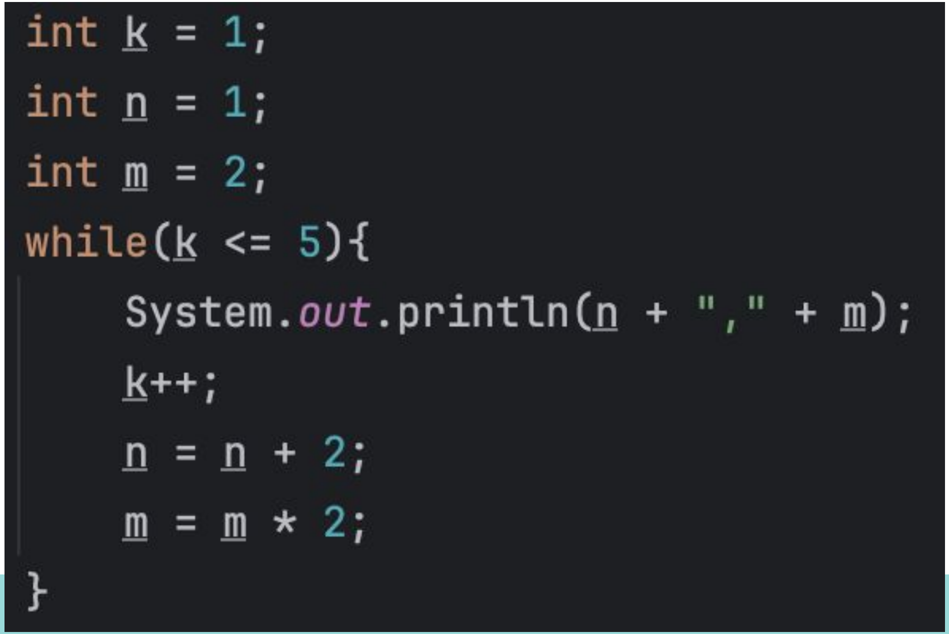
```
1   import java.util.ArrayList;
2   import java.util.List;
3
4   public class Main {
5       public static void main(String[] args) {
6           ArrayList<Integer> data = new ArrayList<>(List.of(2,4,1,-2,-4,1,0));
7           int counter = 0;
8           double sum = 0.0;
9           for (int i = 0; i < data.size(); i++) {
10              int num = data.get(i);
11              if (num > 0) {
12                  counter++;
13                  sum = sum + num;
14              }
15          }
16          System.out.println(sum/counter);
17      }
18  }
19
```

Write the tracetable of the following code.

What happens if I move line 16, between line 14 and 15 ?

Exercice 14 *(GenAI 🟠 Orange)*

Write the tracetable of the following code



```
int k = 1;
int n = 1;
int m = 2;
while(k <= 5){
    System.out.println(n + "," + m);
    k++;
    n = n + 2;
    m = m * 2;
}
```

Homework

Finish all the exercises

Create flashcards for the following terms:

Termination condition
Validation
Verification
Pre-condition loops
Post condition loops
Endless loop
When should you use for loops and when should you use while loops ?
ArrayLists methods: add(), get(), set(), remove(), size(), clear()