Instructor:

SIR MUHAMMAD NAVEED

Created by:

ARSLAN AHMED SHAAD (1163135)

MUHAMMAD BILAL (1163122)

VISIT:www.techo786.wordpress.com

CHAPTER: 3

CONTROL STATEMENTS

NOTE:

Question's Given below are **Long Question's** and also contains **Definitions.**

Java Control Flow Statements

Java Control statements control the order of execution in a java program, based on data values and conditional logic. There are three main categories of control flow statements:

- ✪ Selection statements: if, if-else and switch.
- ✪ Loop statements: while, do-while and for.
- ✪ Transfer statements: break, continue, return, try-catch-finally and assert.

We use control statements when we want to change the default sequential order of execution

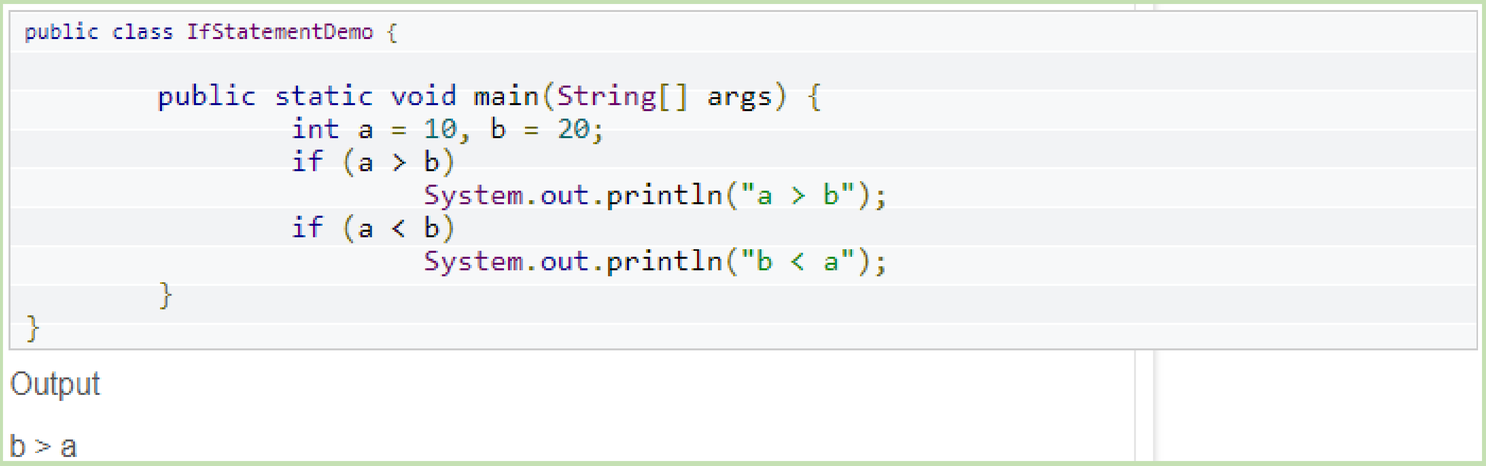
Selection statements:

IF STATEMENT

The if statement executes a block of code only if the specified expression is true. If the value is false, then the if block is skipped and execution continues with the rest of the program.
If statement has the following syntax:

```
if (<conditional expression>)
    <statement action>
```

EXAMPLE



```
public class IfStatementDemo {

        public static void main(String[] args) {
                int a = 10, b = 20;
                if (a > b)
                        System.out.println("a > b");
                if (a < b)
                        System.out.println("b < a");
        }
}
```

Output

```
b > a
```

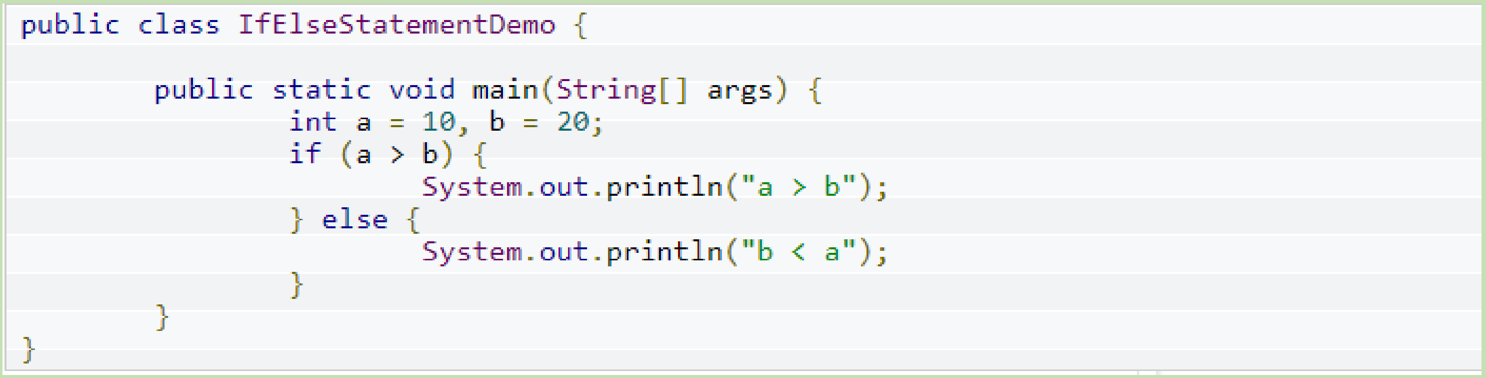
IF ELSE STATEMENT

In if-else statements, if the specified condition in the if statement is false, then the statement after the else keyword (that can be a block) will execute. Note that the conditional expression must be a Boolean expression.

If-else statement has the following syntax:

```
if (<conditional expression>)
<statement action>
else
<statement action>
```

EXAMPLE



```
public class IfElseStatementDemo {

        public static void main(String[] args) {
                int a = 10, b = 20;
                if (a > b) {
                        System.out.println("a > b");
                } else {
                        System.out.println("b < a");
                }
        }
}
```

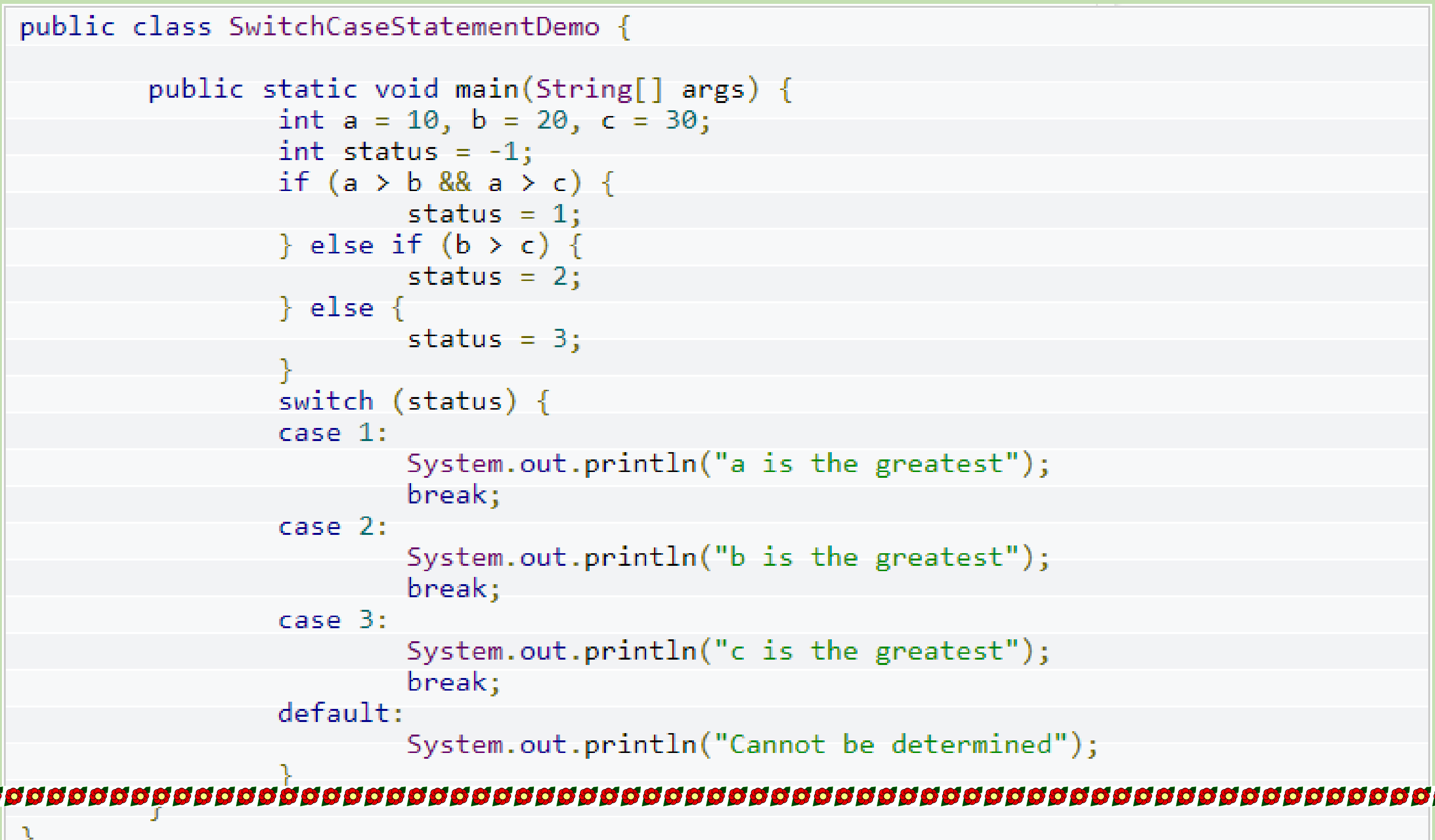
SWITCH CASE STATEMENT

The switch statement is control flow statement that start with an expression and transfer control to one of the case statements on the basis of the value of expression. The keyword "switch" is followed by an expression that should evaluates to byte, short, char or int primitive data types, only.
Break statement is used at the end of every case.
Default label is permitted at the end to improve readability.

Switch statement has following syntax

```
switch(control_expression){
 case expression 1:
 <statement>;
 case expression 2:
 <statement>;
  ...
  ...
 case expression n:
 <statement>;
 default:
 <statement>;
 }//end switch
```

EXAMPLE



```
public class SwitchCaseStatementDemo {

        public static void main(String[] args) {
                int a = 10, b = 20, c = 30;
                int status = -1;
                if (a > b && a > c) {
                        status = 1;
                } else if (b > c) {
                        status = 2;
                } else {
                        status = 3;
                }
                switch (status) {
                case 1:
                        System.out.println("a is the greatest");
                        break;
                case 2:
                        System.out.println("b is the greatest");
                        break;
                case 3:
                        System.out.println("c is the greatest");
                        break;
                default:
                        System.out.println("Cannot be determined");
                }
        }
}
```

Loop statements:

Loop:

Loop allow us to repeat the statement or set of statement on the basis of some condition.

Loop is used to repeatedly execute the line of code or block of code. It is also called repetition/ iteration

Types of Loop

- For loop
- While loop
- Do while loop
- For each loop

FOR LOOP

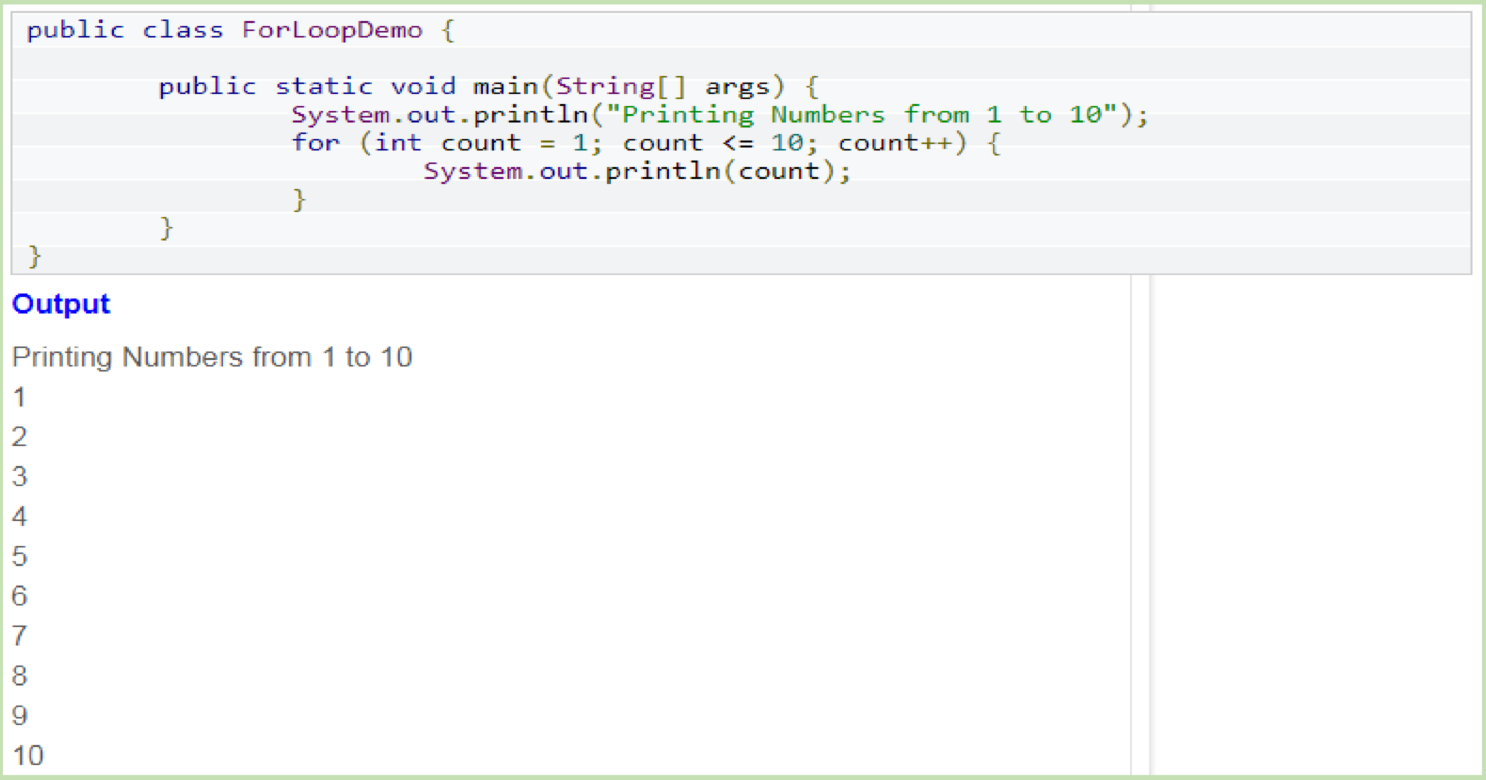
For loop is used to repeatedly execute statement or block of statement for a specified number of times. For loop is suitable when number of iteration are predefined.
For loop has following syntax

```
for(initialization;condition;incr/decr){

//code to be executed

}
```

EXAMPLE



```
public class ForLoopDemo {

        public static void main(String[] args) {
                System.out.println("Printing Numbers from 1 to 10");
                for (int count = 1; count <= 10; count++) {
                        System.out.println(count);
                }
        }
}
```

Output

```
Printing Numbers from 1 to 10
1
2
3
4
5
6
7
8
9
10
```

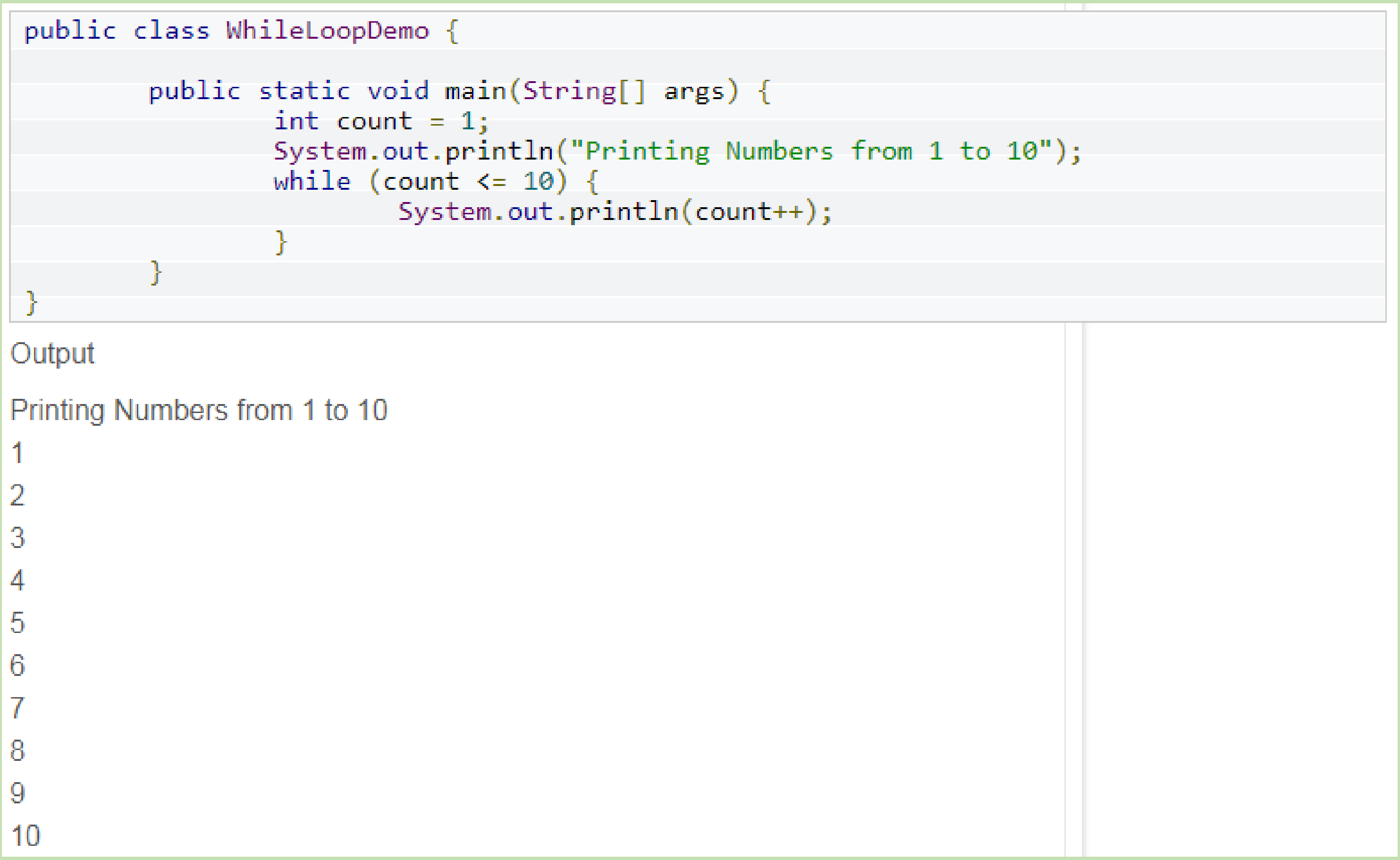
WHILE LOOP

While loop is a statement that repeat the execution of statements or set of statements after while until the condition remains true

While loop has following syntax

```
while(condition){
//code to be executed
}
```

EXAMPLE



```
public class WhileLoopDemo {

	public static void main(String[] args) {
		int count = 1;
		System.out.println("Printing Numbers from 1 to 10");
		while (count <= 10) {
			System.out.println(count++);
		}
	}
}
```

Output

```
Printing Numbers from 1 to 10
1
2
3
4
5
6
7
8
9
10
```

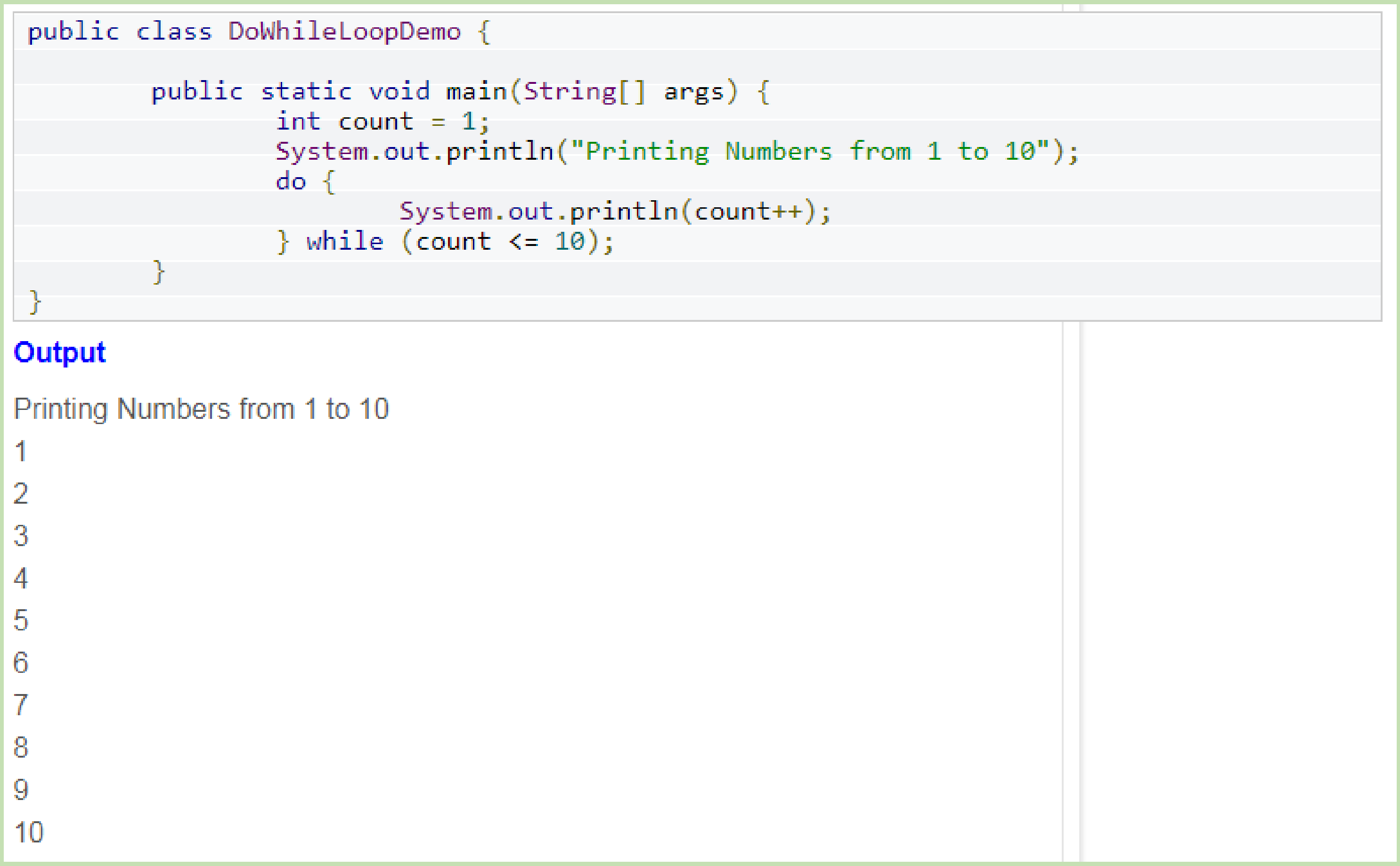
DO WHILE LOOP

Working of do while loop is most likely as while loop except that condition is evaluated at the bottom of loop in do while

Do while loop has following syntax

```
do{
//code to be executed
}while(condition);
```

EXAMPLE



```
public class DoWhileLoopDemo {

        public static void main(String[] args) {
                int count = 1;
                System.out.println("Printing Numbers from 1 to 10");
                do {
                        System.out.println(count++);
                } while (count <= 10);
        }
}
```

Output

```
Printing Numbers from 1 to 10
1
2
3
4
5
6
7
8
9
10
```

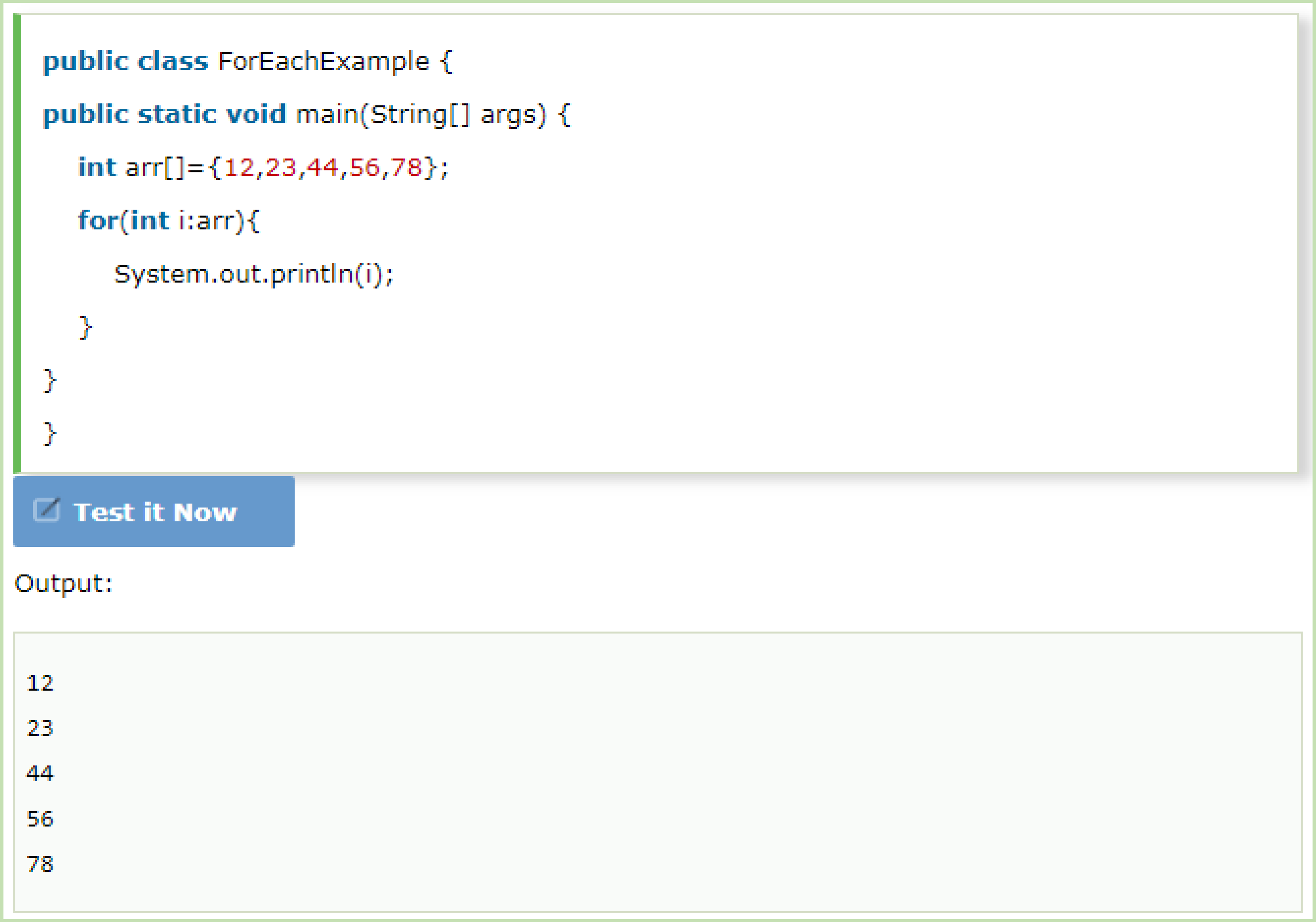
FOR EACH LOOP

When we loop through collection and arrays we use for each loop. It is easier to use than simple for loop because we don't need to increment value and use subscript notation.

For each loop has following syntax

```
for(Type var:array){
//code to be executed
}
```

EXAMPLE



```
public class ForEachExample {
public static void main(String[] args) {
    int arr[]={12,23,44,56,78};
    for(int i:arr){
        System.out.println(i);
    }
}
}
```

Output:

```
12
23
44
56
78
```

Q: What for each loop cannot do?

Answer:

1: for each loop cannot iterate background through the element of an array

2: cannot use the single loop counter for two different arrays.

3: we cannot iterate through array using get () method.

Transfer statements:

Transfer statements are used to unconditionally transfer the program control to another part of the program.
Java provides the following jump statements:

- break statement
- continue statement
- return statement

BREAK STATEMENT

The Java break is used to break loop or switch statement. It breaks the current flow of the program at specified condition.
Syntax

```
Jump-statement;

break;
```

CONTINUE STATEMENT

The continue statement is used when you want to continue running the loop with the next iteration and want to skip the rest of the statements of the body for the current iteration.

Syntax

```
jump-statement;

continue;
```

RETURN STATEMENT

The return statement is used to immediately quit the current method and return to the calling method.