upGrad
#LifeKoKaroLift

PGC Full Stack Development

06/10/2020

upGrad

Course: Foundation of Programming

Lecture on: Loops and Methods

Instructor : Rajesh Kumar

Course Roadmap

- Introduction to Java
- Operators and Conditionals
- **Loops and Methods**
- Strings and Arrays

upGrad

In the previous class, we discussed...

1 Operators and their types

2 Expressions and statements

3 Operator precedence and associativity

4 Conditionals in Java

5 Ternary operator

Yesterday's Homework

Find out the largest number among three given numbers.

Input format:
First line containing the first number
Second line containing the second number
Third line containing the third number

Output format:
First line containing the value of the largest number

Sample input 1:

```
12
4
5
```

Sample output 1:

```
12
```

Yesterday's Homework Solution

upGrad

```java
package com.company;
import java.util.Scanner;
public class Main {

   public static void main(String args[] ) throws Exception {
        Scanner sc = new Scanner(System.in);
        int num1 = sc.nextInt();
        int num2 = sc.nextInt();
        int num3 = sc.nextInt();
        int max = (num1 > num2) ? (num1 > num3 ? num1 : num3) : (num2 > num3 ? 
num2 : num3);
        System.out.println(max);
    }
}
```

upGrad

Today’s Agenda

1 What are loops?

2 Different types of loops: While, Do-While, For

3 Break and continue statements

4 Methods

To be filled by Ifrah

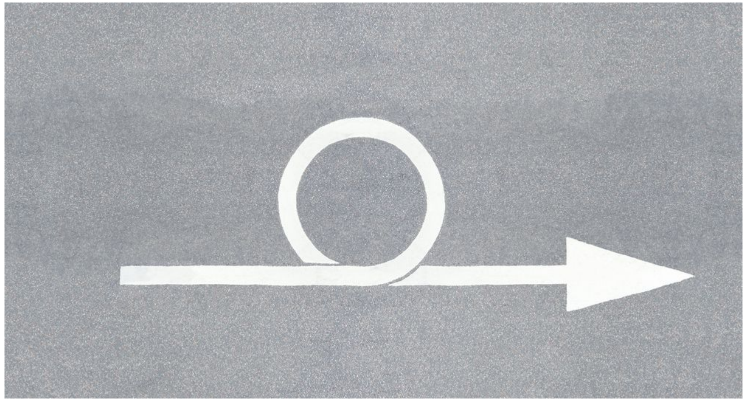
What Are Loops?

- In our daily lives, there is always repetition of events such as our morning routine or going to school/college and attending classes. You can think of so many things that happen over and over again.

- Similar to these, we have repetitions in programming languages as well. Suppose you want to print integers from 1 to 10. You can write 10 print statements. But what if you had to print integers from 1 to 100 or more? Would you still write individual statements?

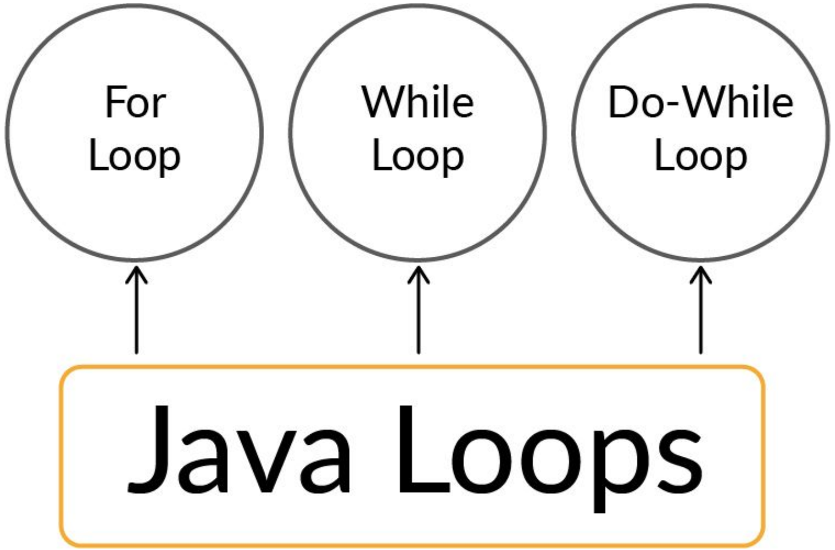
- No, programming offers loops that perform repetitive actions over and over again.

- A loop is a programing feature that facilitates performing an action repeatedly until some condition is met.



Loops in Java

upGrad



- In Java, a loop performs some lines of code in an ordered fashion until a predefined condition is met.
- The loop exits only when the condition is not satisfied anymore.
- It is important to have a condition that is false after a desired number of iterations; otherwise, the loop will never end.
- Java offers the following types of loops:
  a. For

  b. While

  c. Do-While

upGrad

While Loops

Point 1

A while loop evaluates a boolean condition and iterates through a set of statements until the condition returns false.

Point 2

The loop iterates as long as the condition given evaluates to true.

Point 3

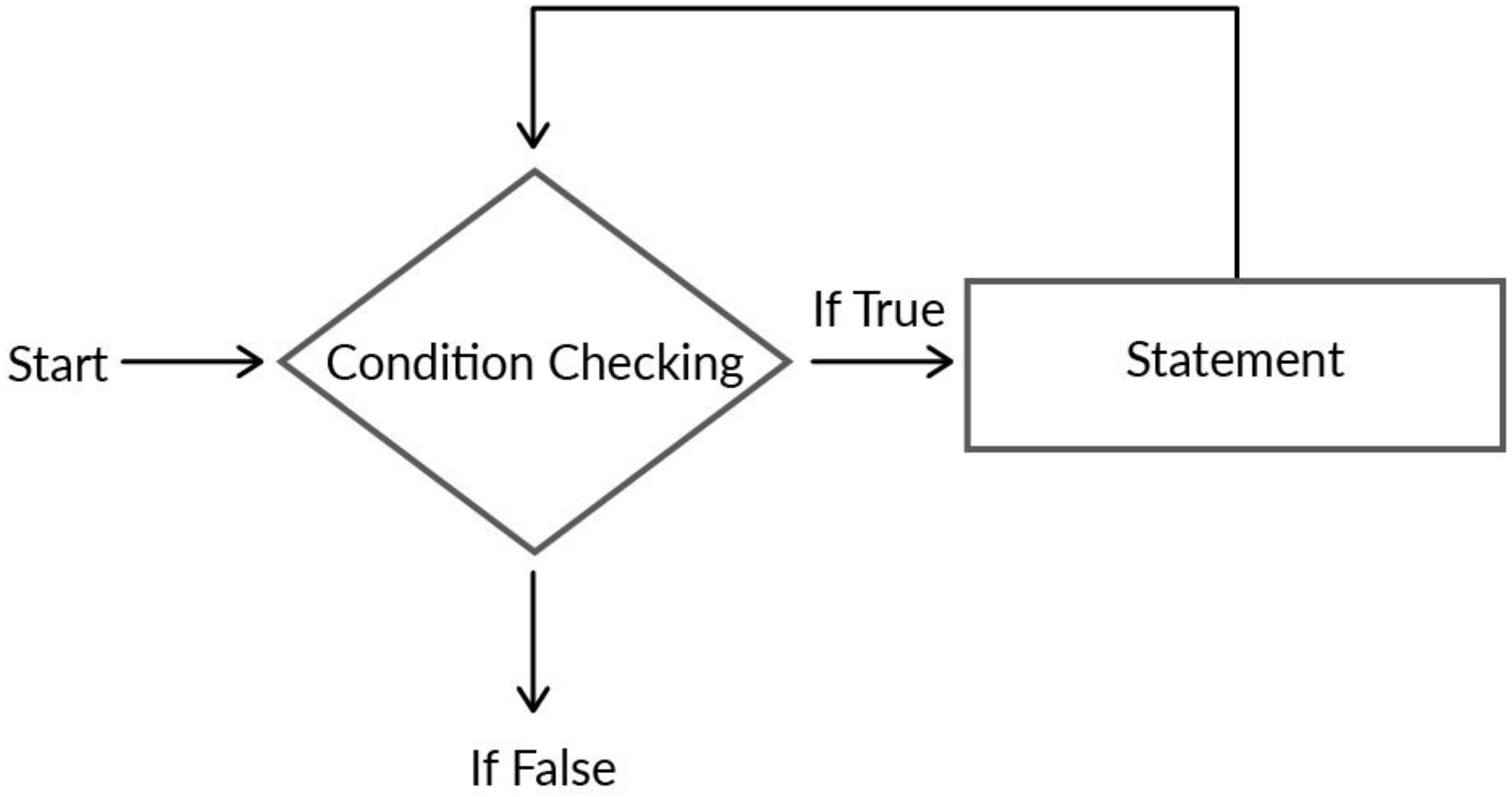
The boolean condition is checked first, and then, the control proceeds into the loop structure only if the condition evaluates to true.

Point 4

It is also known as an entry-controlled loop.

While Loops

upGrad



While Loops

upGrad

Syntax:

```
while(boolean condition) {
  //statements
 update_expression;
}
```

The loop evaluates the boolean condition first. If the condition is true the statements inside the loop are executed and the expression is updated.

```
public class Main {

  public static void main(String[] args) {
     int i = 0;
     while(i!=5){
        System.out.println("I am less than 5");
        i++;
     }
  }
}
```

Hands-On: Exercise 1

upGrad

Write a program to print all the multiples of 2 till 20.

upGrad

Do-While Loops

Point 1

A Java do-while loop executes the statement first and then checks the condition.

Point 2

Other than this, it is similar to a while loop.

Point 3

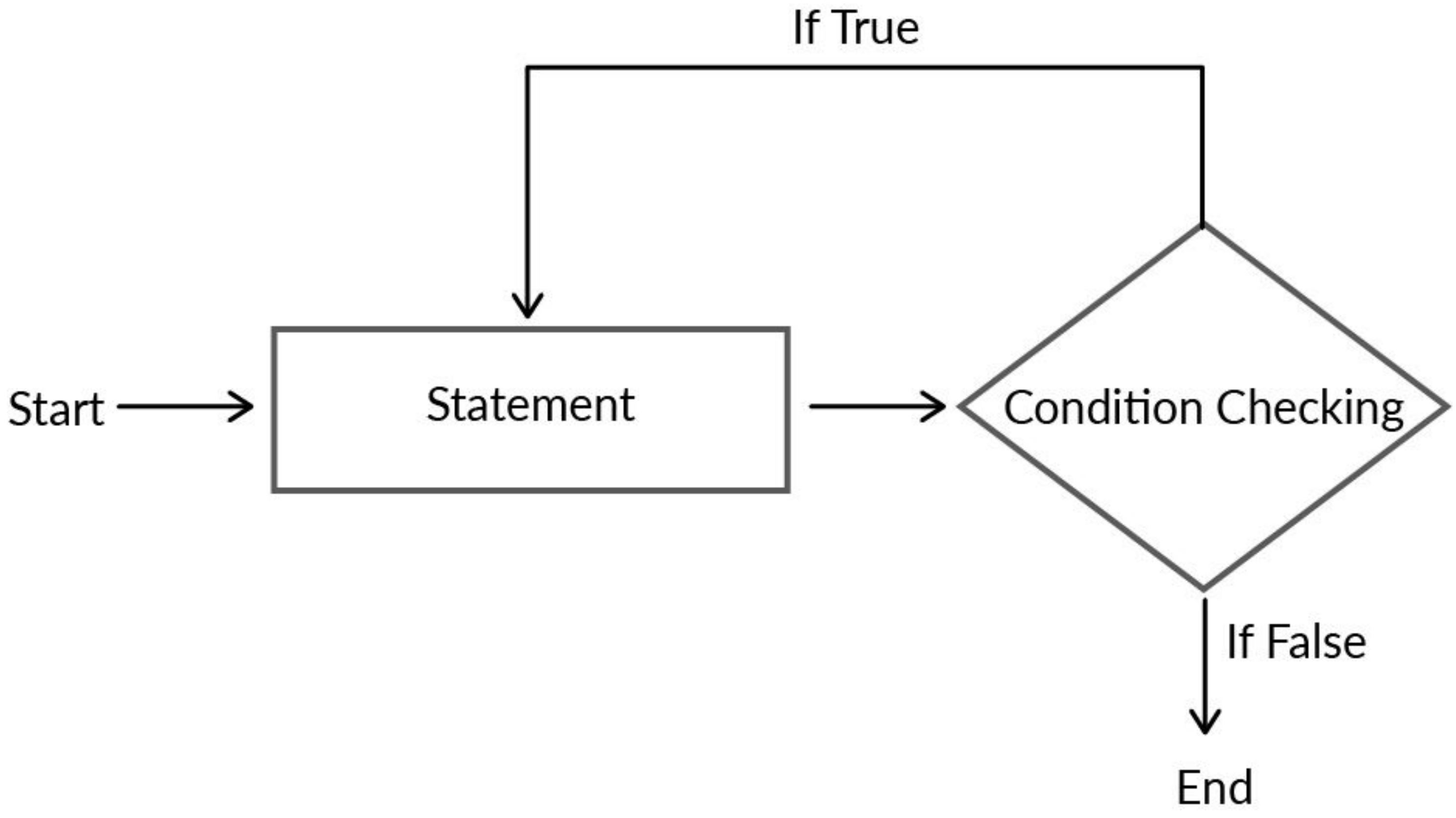
The difference lies in the fact that even if the condition is false at the starting of the loop, the statements will still get executed.

Point 4

It is called an exit-controlled loop.

Do-While Loops

upGrad



Do-While Loops

upGrad

Syntax:

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

The do while loop starts with the execution of the statement(s) and there is no checking of any condition for the first time.

After the execution of the statements once, the condition is checked for true or false value.

If evaluated to true, next iteration of loop starts until the condition becomes false.

```
public class Main {

  public static void main(String[] args) {
     int i = 0;
     do{
        System.out.println("I am less than 5");
        i++;
     }while(i!=5)
  }
}
```

Hands-On: Exercise 2

Write a program to print all the multiples of 2 till 20 using do-while loop.

For Loops

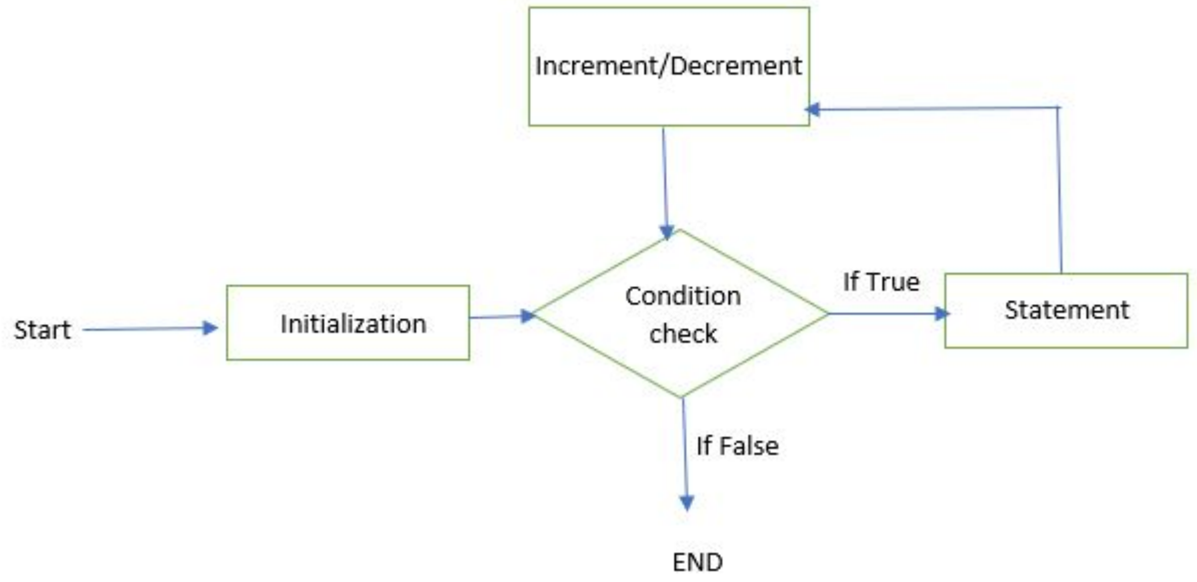
upGrad

- A for loop provides a concise way of writing a loop structure.

- For loop that had the same elements of a while loop:
  - A counter
  - A loop-continuation statement
  - An increment statement

- Unlike a while loop, a for statement consumes the initialisation, condition and increment/decrement in one line, thereby providing a shorter, easy-to-debug looping structure.

For Loops


Increment/Decrement
Start
Initialization
Condition check
If True
Statement
If False
END

For Loops

Syntax:

for (initialization condition; termination condition; increment/decrement condition) {

//Statements;

}

The for loop has the following components:

- **Initialisation**: Here, we initialise the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to the loop only.

- **Terminaion condition**: It is used for testing the exit condition for a loop. It must return a boolean value. It is also an **entry control loop,** as the condition is checked prior to the execution of the loop statements.

- **Statement execution**: Once the condition is evaluated to true, the statements in the loop body are executed.

- **Increment/Decrement**: It is used for updating the variable for the next iteration.

```
public class ForLoop {
    public static void main(String
args[]) {
        for (int number = 1; number <=
100; number++) {
            System.out.print(number+ "
");
            number++;
        }
    }
}
```

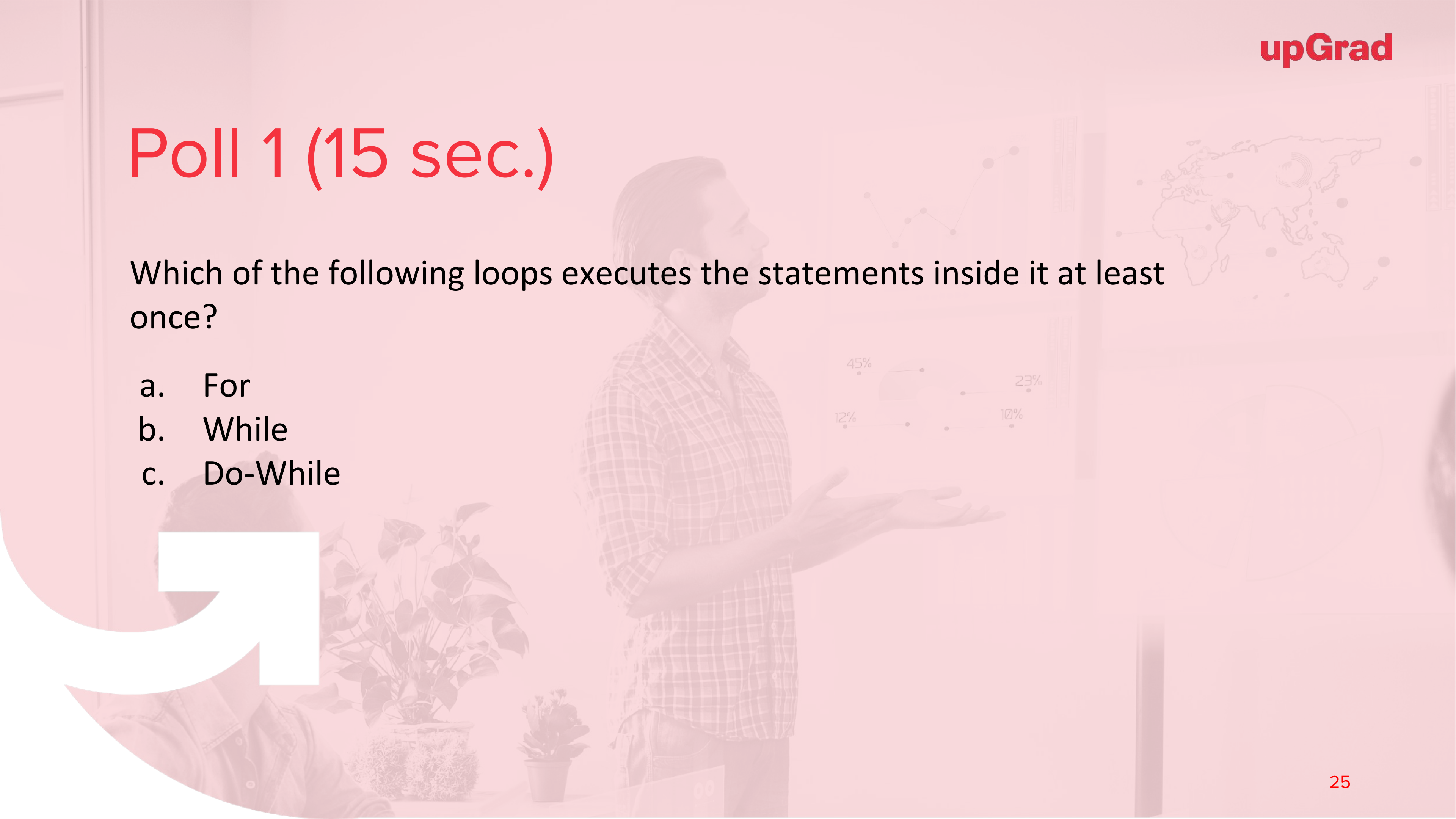
Hands-On: Exercise 3

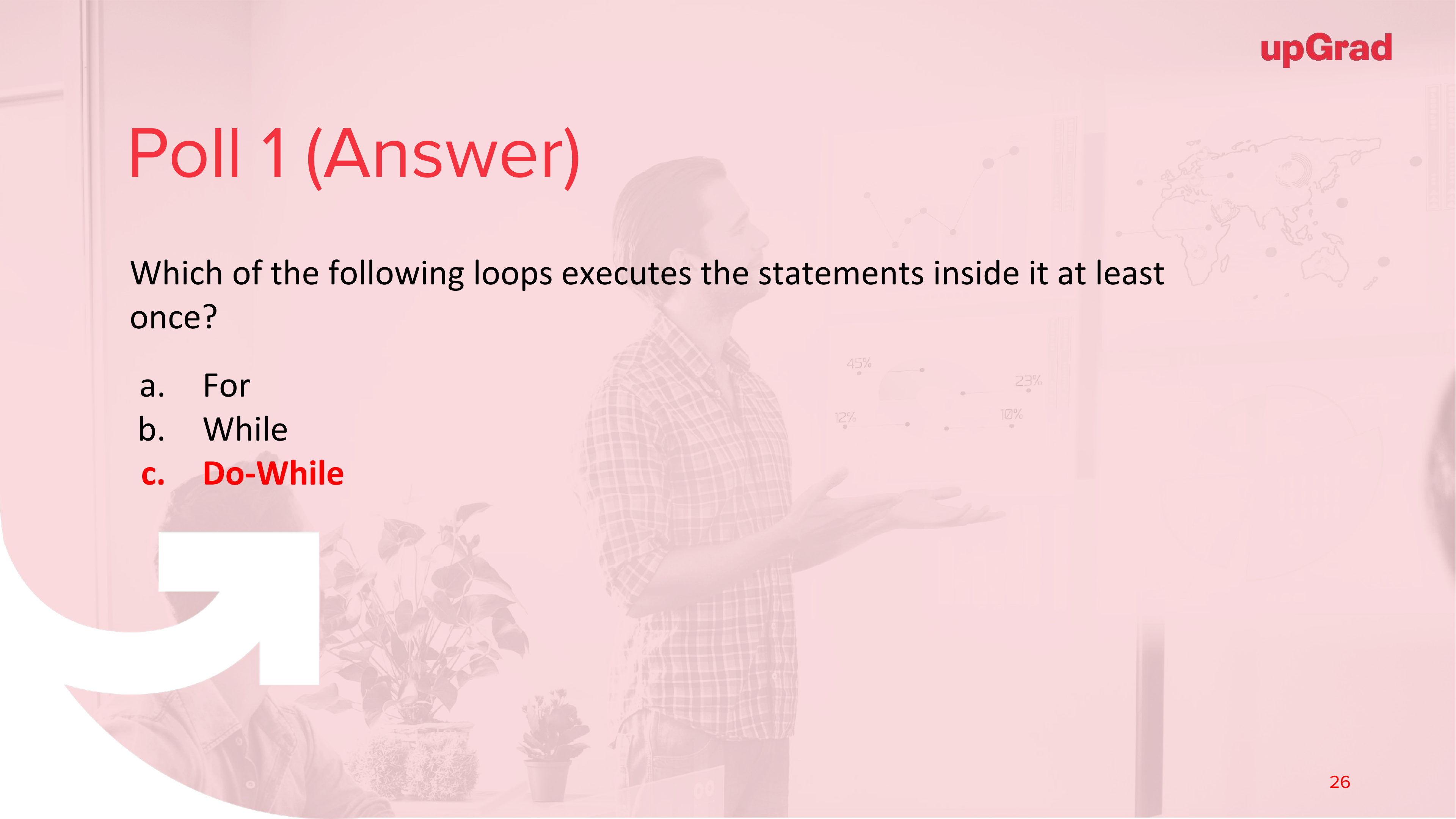
Write a program to print all integers till 50..

Poll 1 (15 sec.)

Which of the following loops executes the statements inside it at least once?

a. For
b. While
c. Do-While





Poll 1 (Answer)

Which of the following loops executes the statements inside it at least once?

a. For
b. While
c. Do-While

Poll 2 (15 sec.)

What will be the output of the below code?

```
public class MyClass {
    public static void main(String[] args) {
        boolean keepRunning = false;
        while (keepRunning = true) {
            System.out.println("Hey");
        }
    }
}
```

a. “Hey” infinite times.
b. Nothing.
c. “Hey” once
d. Error

Poll 2 (Answer)

What will be the output of the below code?

```
public class MyClass {
    public static void main(String[] args) {
        boolean keepRunning = false;
        while (keepRunning = true) {
            System.out.println("Hey");
        }
    }
}
```

a. “Hey” infinite times.

b. Nothing.

c. “Hey” once

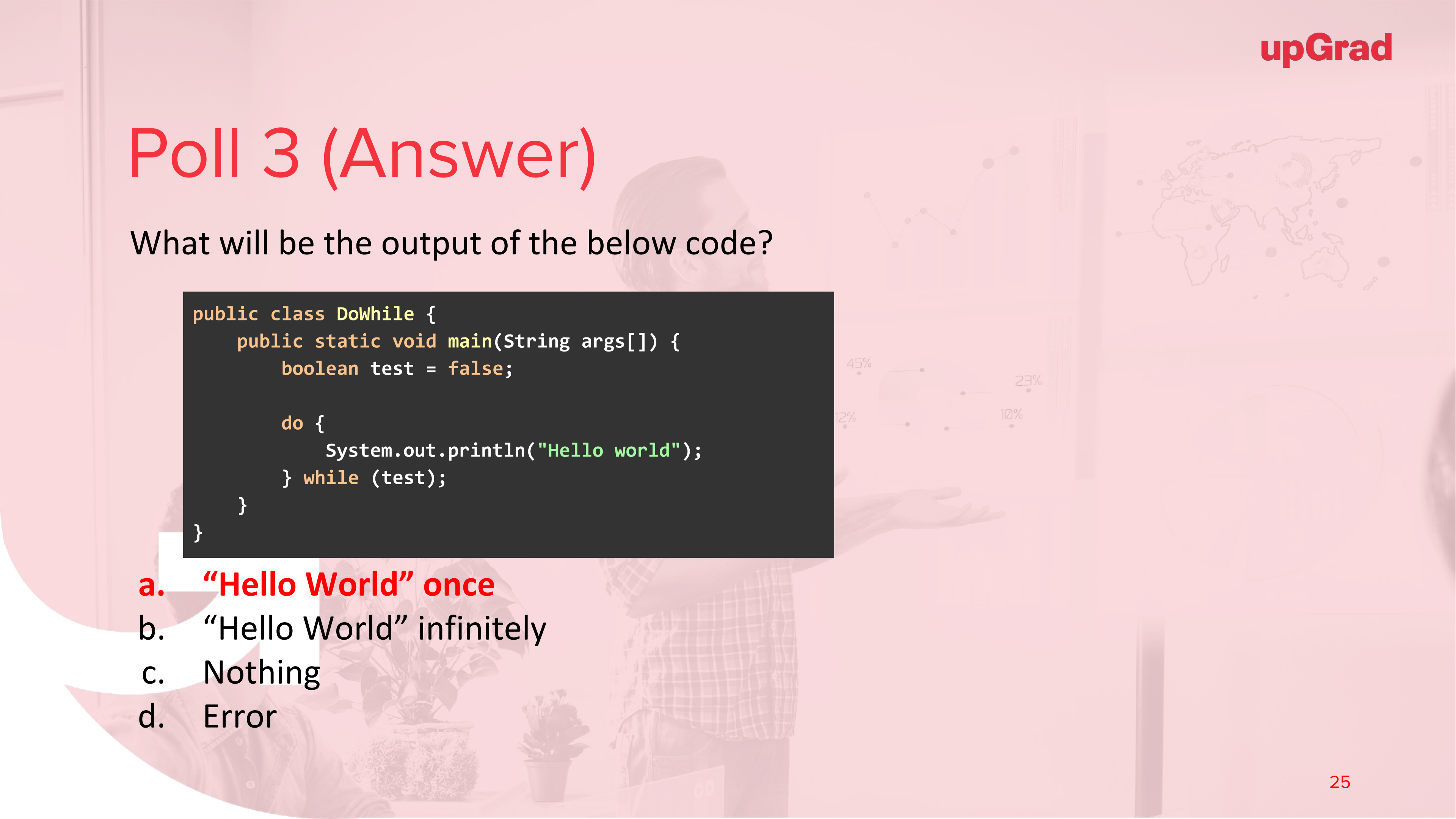
d. Error

Poll 3 (15 sec.)

What will be the output of the below code?

```
public class DoWhile {
    public static void main(String args[]) {
        boolean test = false;

        do {
            System.out.println("Hello world");
        } while (test);
    }
}
```

a. “Hello World” once
b. “Hello World” infinitely
c. Nothing
d. Error



Poll 3 (Answer)

What will be the output of the below code?

```
public class DoWhile {
    public static void main(String args[]) {
        boolean test = false;

        do {
            System.out.println("Hello world");
        } while (test);
    }
}
```

a. “Hello World” once
b. “Hello World” infinitely
c. Nothing
d. Error

Variable Scope

upGrad

- Scope of variables means where they will be accessible in the program.
- Java is a statically typed language, so every identifier has a scope.
- There are three different types of scopes in Java:

a. **Class Level Scope:** The variables declared inside a class can be accessed anywhere in the class. They need to be declared outside of methods and loops.

```
public class Test {

  int i = 0; //can be accessed anywhere
inside the class
     while(i!=5){
        //some statements
     }
  }
}
```

Variable Scope

upGrad

b. **Method Level Scope:** The variables declared inside a method have only method level scope and cannot be accessed outside the methods.

```
public static void main(String[] args) {
  for(int i=0; i<5; i++){
     //some statements
  }
  i = 0; //cannot be accessed here
}
```

c. **Block Scope:** The variables declared inside curly brackets {} can be accessed only inside the brackets. These can be defined inside a method or a loop.

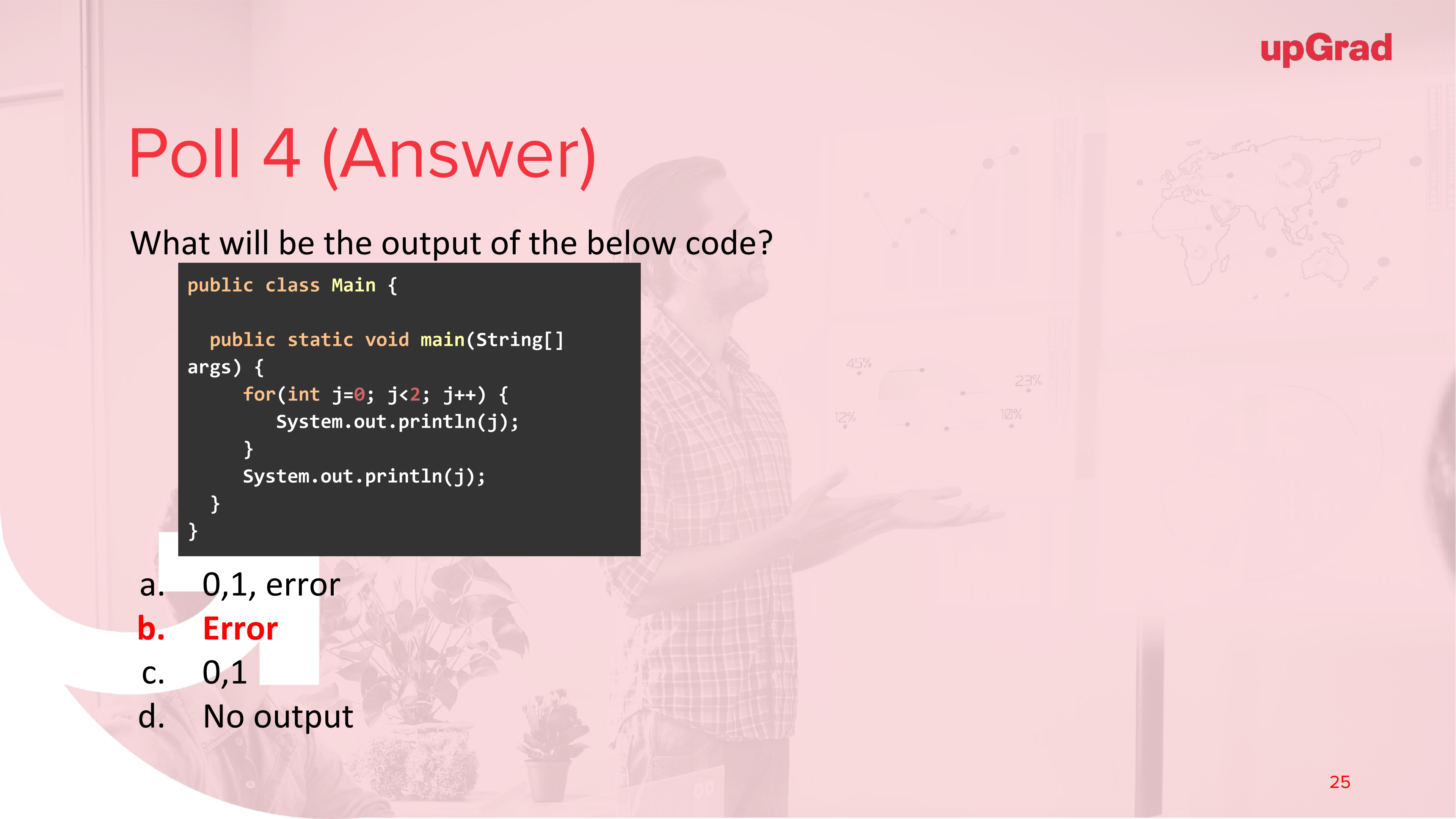
```
public static void main(String[] args) {
  for(int i=0; i<5; i++){
     {
      Int j=0;
     }
     j = 1; //cannot be accessed here
  }
  j = 1; //cannot be accessed here
}
```

Poll 4 (15 sec.)

What will be the output of the below code?

```
public class Main {

  public static void main(String[]
args) {
     for(int j=0; j<2; j++) {
        System.out.println(j);
     }
     System.out.println(j);
  }
}
```

a. 0,1, error
b. Error
c. 0,1
d. No output



Poll 4 (Answer)

What will be the output of the below code?

```
public class Main {

  public static void main(String[]
args) {
     for(int j=0; j<2; j++) {
        System.out.println(j);
     }
     System.out.println(j);
  }
}
```

a. 0,1, error

b. Error

c. 0,1

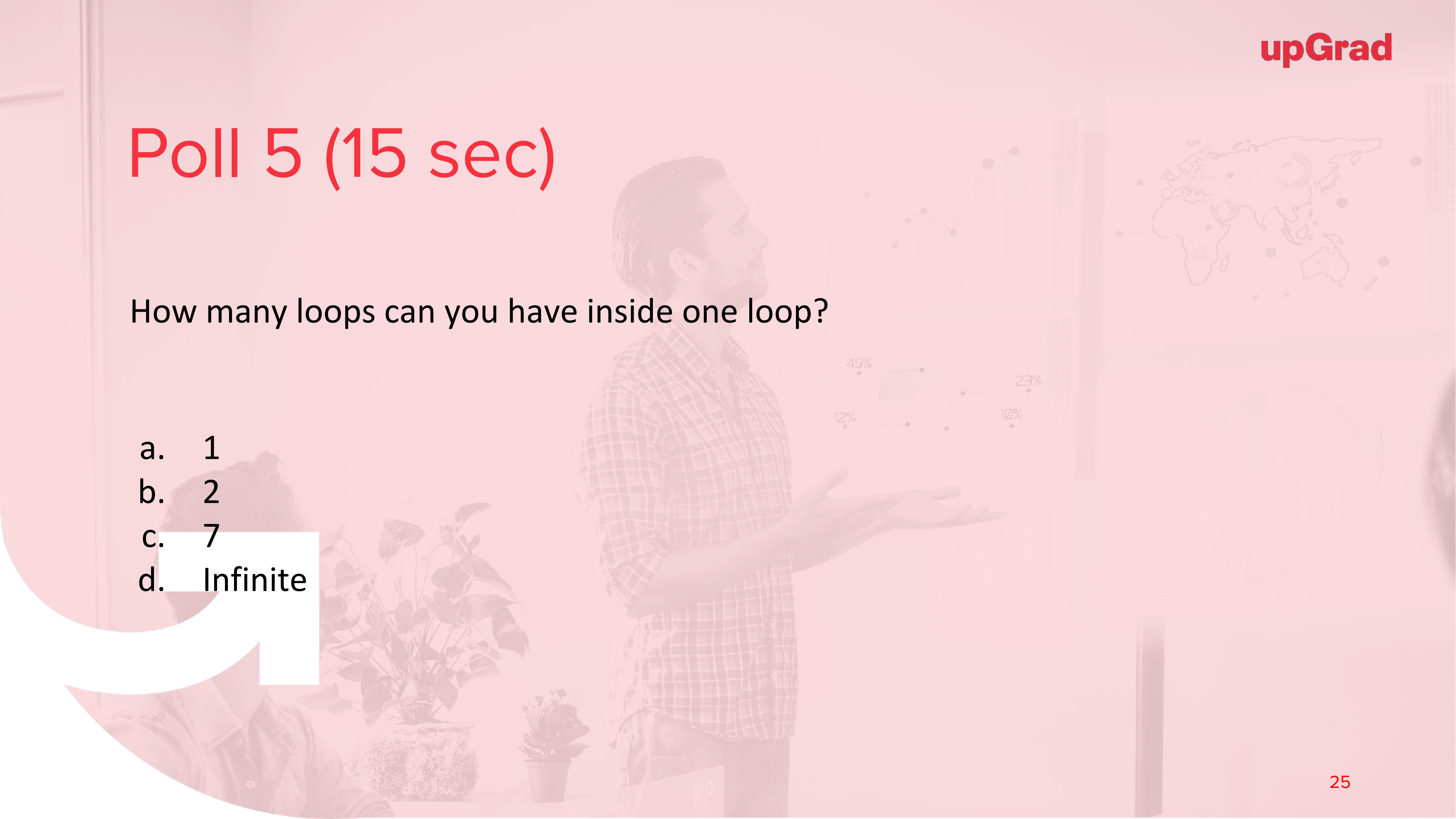
d. No output

Nested Loops

upGrad

- Similar to nested if-else statements, loops can also be nested.
- A nested loop is one which has a loop within a loop.

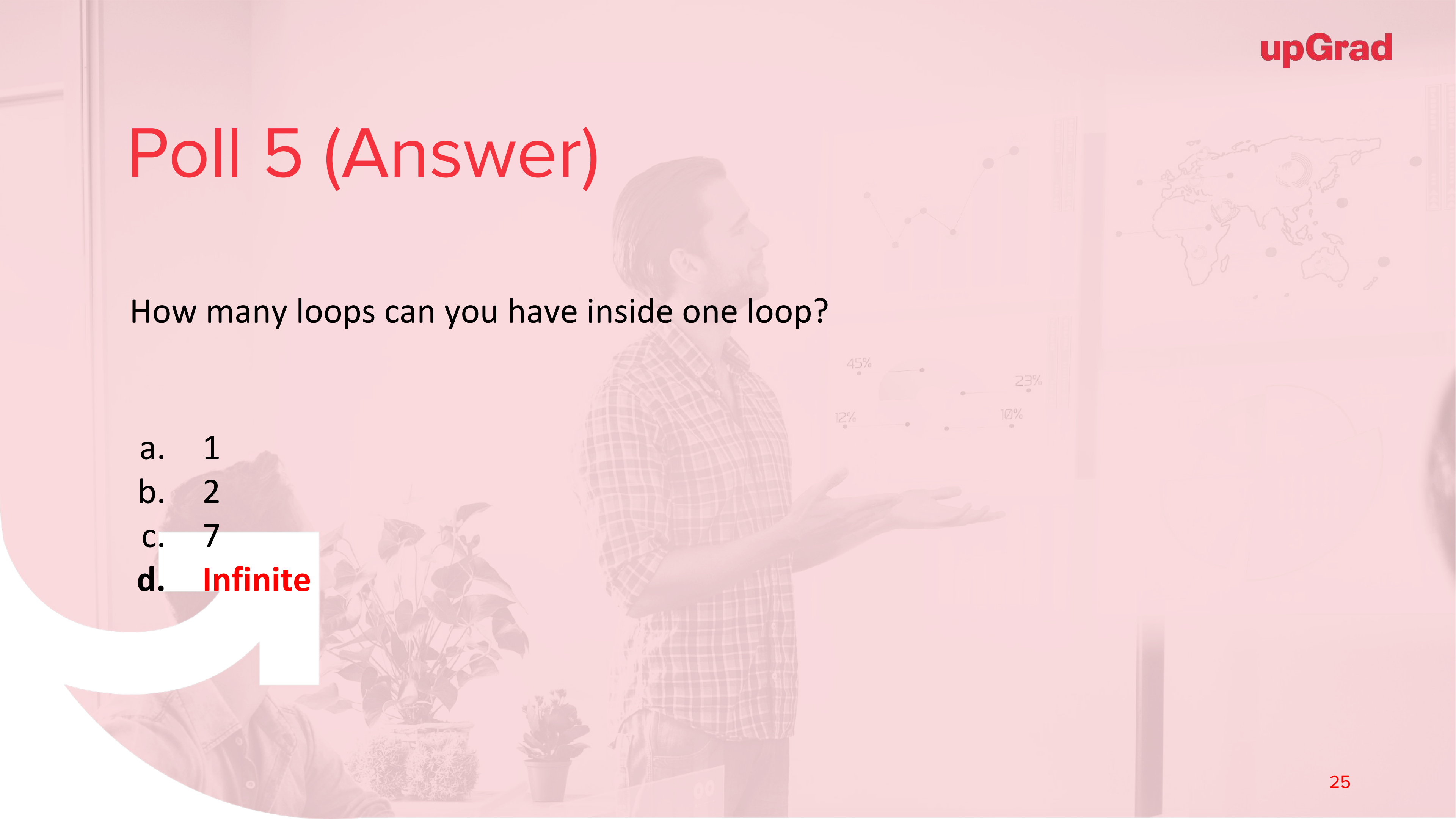
```
public class Main {

  public static void main(String[] args) {
     //Outer loops
     for (int num = 2; num <= 1000; num++) {
        //inner loop
        for (int i = 2; i <= num; i++) {

        }
     }
  }
}
```



Poll 5 (15 sec)

How many loops can you have inside one loop?

a. 1
b. 2
c. 7
d. Infinite



Poll 5 (Answer)

How many loops can you have inside one loop?

a. 1
b. 2
c. 7
d. Infinite

Hands-On: Exercise 4

upGrad

Write a program to print the below pattern.

```
1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

1 2 3 4 5 6

1 2 3 4 5 6 7

1 2 3 4 5 6 7 8
```

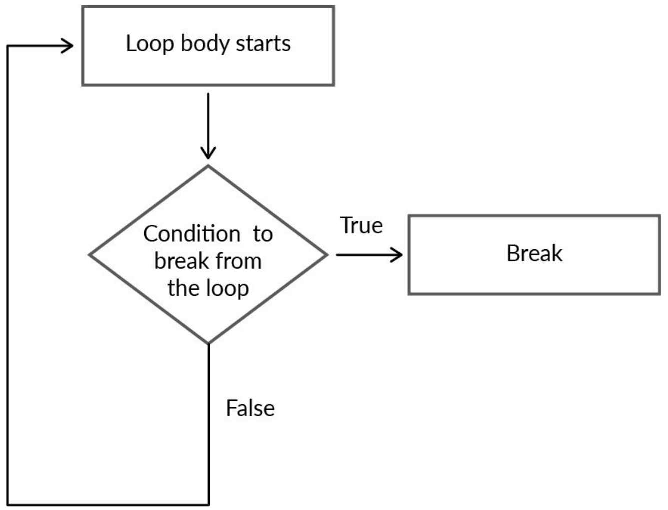
Break and Continue Statements

upGrad

- Break and continue statements are jump statements that are used to skip some statements inside a loop or terminate a loop immediately.
- These statements can be used inside any loop, such as a for, while or do-while loop.

Break Statement

- A break statement is a loop control statement that is used to terminate a loop.
- As soon as the break statement is encountered within a loop, the loop iterations stops there. The control returns from the loop immediately to the first statement after the loop.
- As we discussed in the previous session, a break statement is also used with a switch statement.



Continue Statement

upGrad

- A continue statement in Java is used to skip the current iteration of a loop.

- We can use a continue statement inside any type of loop, such as a for, while or do-while loop.

- Basically, continue statements are used in situations where we want to continue the loop but do not want the remaining statement after the continue statement.

Break vs Continue Statement

upGrad

Break Statement

- A break statement is used to terminate the loop immediately.
- Break keywords are used to indicate break statements in Java programming.
- We can use a break statement with a switch statement.
- A break statement terminates the whole loop early.
- It stops the execution of the loop.

Continue Statement

- A continue statement is used to skip the current iteration of the loop.
- Continue keywords are used to indicate continue statements in Java programming.
- We cannot use a continue statement with a switch statement.
- A continue statement starts the next iteration early.
- It does not stop the execution of the loop.

Hands-On: Exercise 5

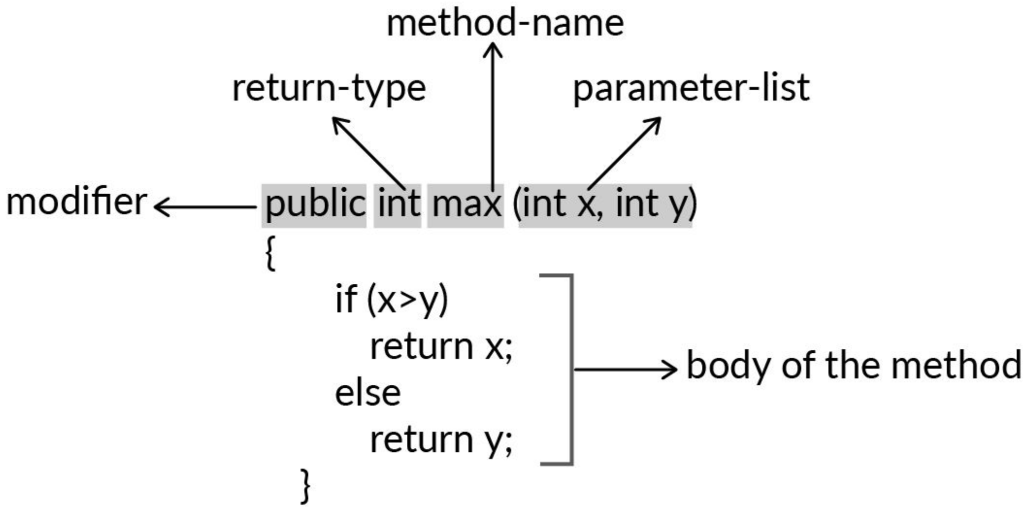
Write a program to find out odd numbers in first 100 integers.

Methods

upGrad

- Methods are used to perform specific tasks.
- They are a collection of statements that return the result to the caller.
- A method can also perform a specific task without returning anything.
- Methods allow the reuse of code to avoid retyping it.
- In Java, a method must be part of some class.

Method Declaration:



```
public int max (int x, int y)
{
      if (x>y)
         return x;
      else
         return y;
   }
```

Components of a method declaration

upGrad

- **Modifier: It defines the access type of the method, i.e., from where it can be accessed in your application. Java has the following types of access specifiers:**
    - Public: These are accessible in all classes in your application.
    - Protected: These are accessible within the class in which they are defined and in its **subclass(es).**
    - Private: These are accessible only within the class in which they are defined.
    - Default (declared/defined without using any modifier): These are accessible within the same class and package within which their classes are defined.
- **The return type**: This is the data type of the value returned by the method, or void, which does not return a value.
- **Method name**: This denotes the method name.
- **Parameter list**: This is a comma-separated list of the input parameters defined, preceded by their data type within parentheses. **Note: In case there are no parameters, you must use empty parentheses ().**
- **Exception list**: You can specify the exceptions that you expect the method to throw.
- **Method body**: It is enclosed by braces. The code that you need to be execute to perform your intended operations using the method.

Method Signature

upGrad

Method Signature:

A method signature consists of name and a parameter list.

For example:

max(int x, int y)

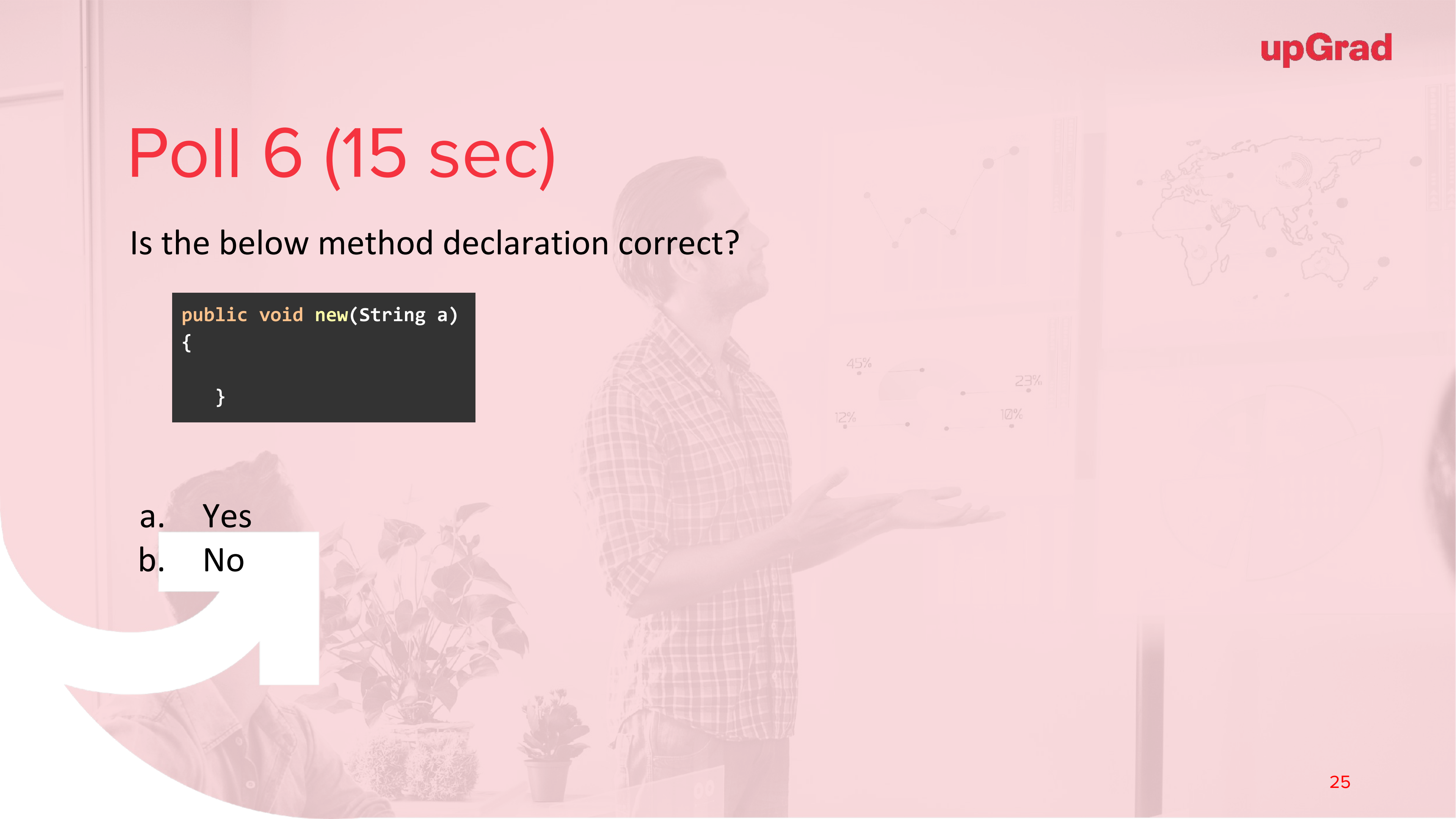
Note: The return type and exceptions are not considered to be part of it.

Naming a method: A method name is typically a single word that should be a **verb** in lowercase or multiple words that begin with a **verb** in lowercase followed by an **adjective, noun, etc.** After the first word, the first letter of each word should be capitalised.

getSum(int x, int y)

changeSpeed()

computeMax(int x, int y, int z)



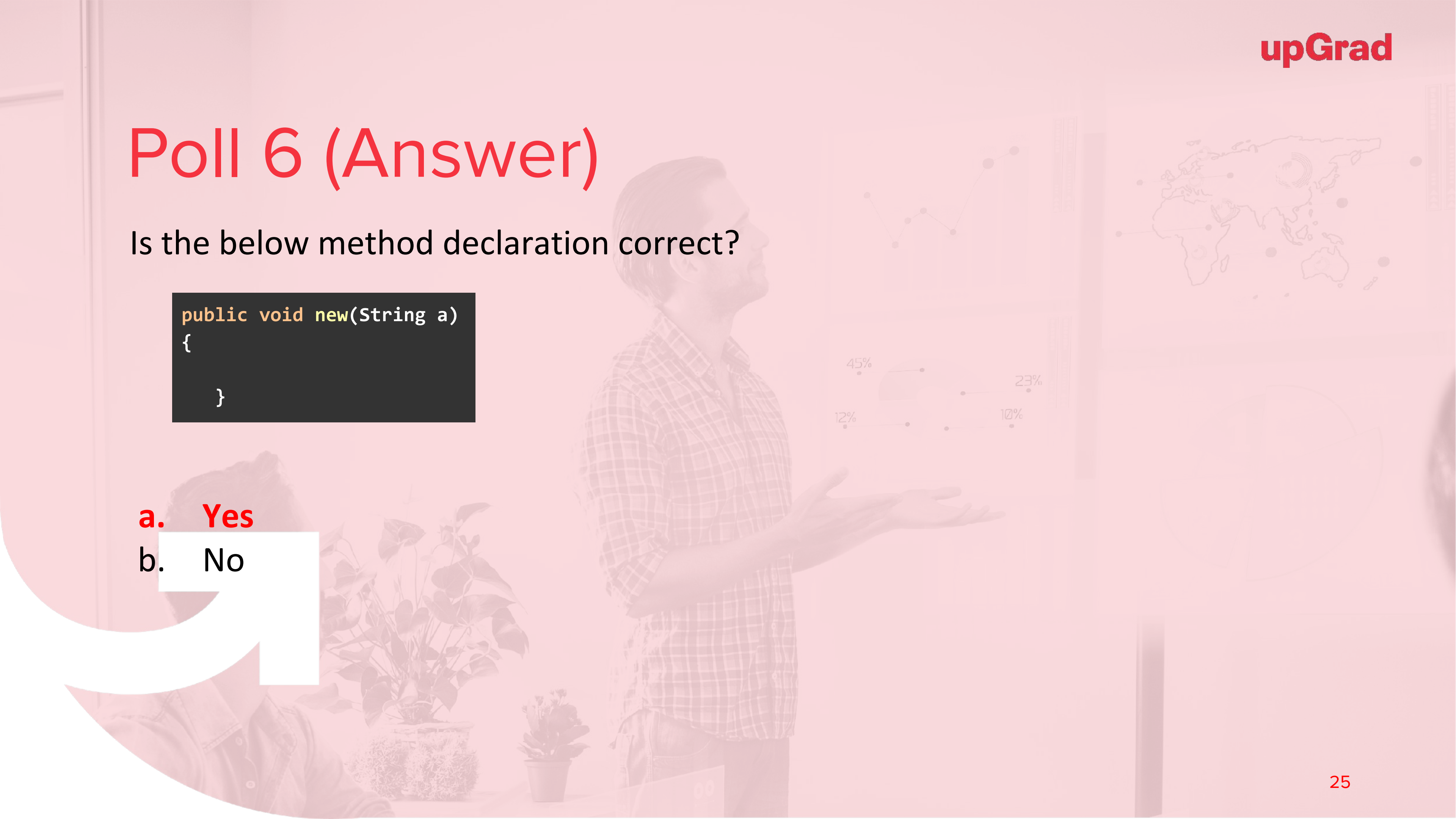
upGrad

Poll 6 (15 sec)

Is the below method declaration correct?

```
public void new(String a)
{

   }
```

a. Yes
b. No



Poll 6 (Answer)

Is the below method declaration correct?

```
public void new(String a)
{

   }
```

[✓ a. Yes]

b. No

Method Parameters and Variables

Method parameters make it possible to pass values to a method, on which the method can operate. They are declared inside the parentheses after the method name.

The **max(int x, int y)** method in the example before takes two parameters **x** and **y**. The parameters are both of type **int** as written in front of each parameter name.

A method parameter is similar to a variable. You can read its value and also change it.

Variables inside methods:
As discussed earlier, you can declare variables inside a method. These variables are known as local variables. A local variable is used just like any other variable, but it is only accessible inside the scope of the method.

```java
public void multiply(int a, int b) {
   int c = 0;
   c = a*b;
   System.out.println(c);
}
```

Calling Methods

A method needs to be called for using its functionality. We can call methods as follows:

a. **With parameters:**

```
public void multiply(int a, int b) {
   int c = 0;
   c = a*b;
   System.out.println(c);
}
```

b. **Without parameter:**

```
public class Main {

  public static void main(String[] args) {
     int d = multiply(3, 10);
     System.out.println(d);
        }
  public static int multiply(int a, int b) {
     int c = 0;
     c = a*b;
     return c;
  }
}
```

Loops and Conditionals in methods

- A method can have loops and conditional within it.
- These can be used to perform various complex operations.

a. **Conditional within a method:**

```
public void multiply(int a, int b) {
   if(a < b)
     System.out.println(“A is greater);
}
```

b. **Loop within a method:**

```
public void multiply(int a, int b) {
   for(i=0; i<10; i++)
     //some statements
}
}
```

Hands-On: Exercise 6

Write a program that checks and prints all prime numbers from 1 to 1000.

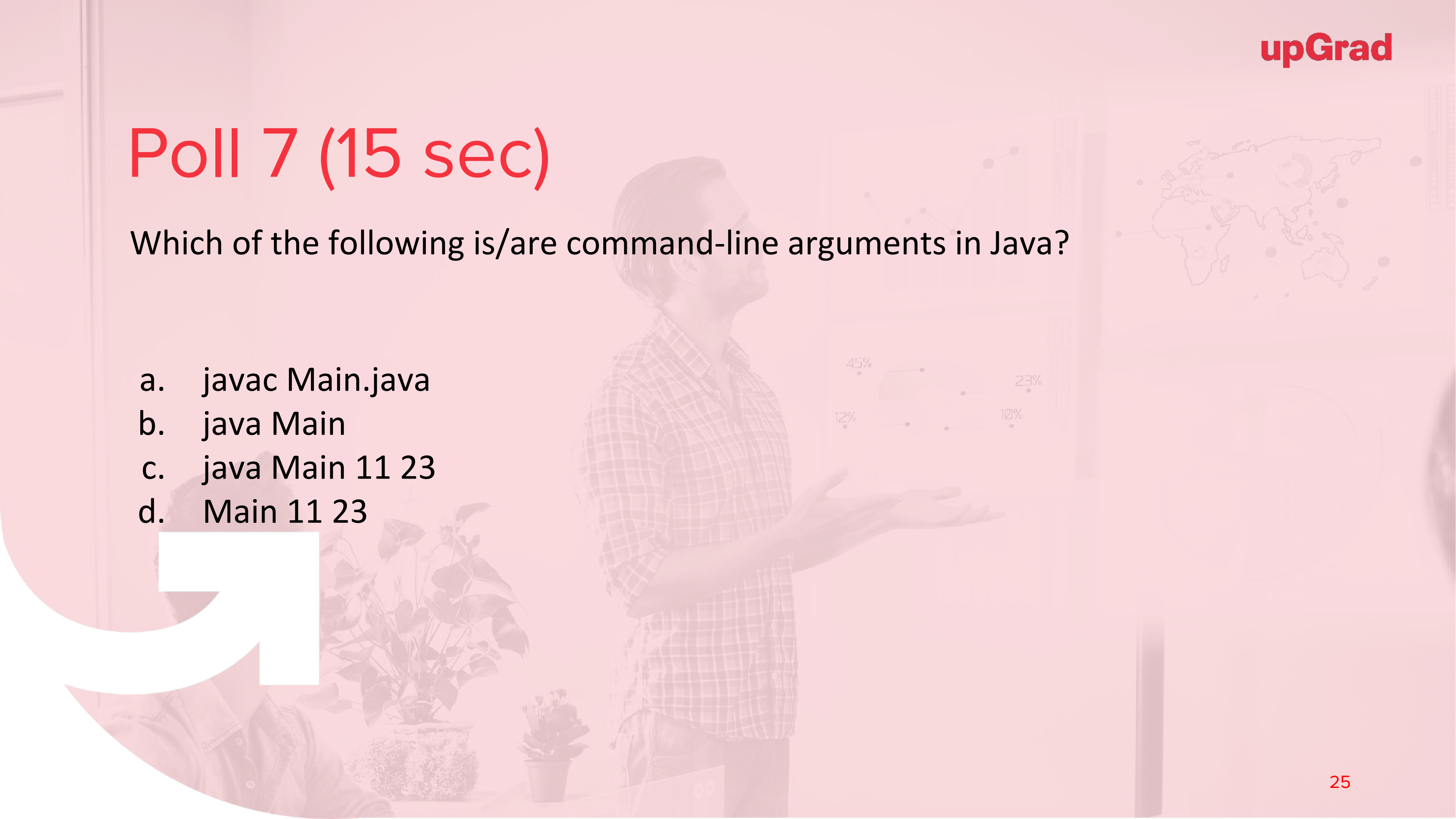
Command line arguments

upGrad

A command-line argument is an information that is directly executed in the command line. They follow the program's name on the command line when executed.
Command line arguments are stored as strings in the String array passed to main( ).

We can run a Java program by writing the below command:
java className Hello World
Here, the name of the class is className, followed by, after the className, i.e, ‘Hello World’, are command line arguments.

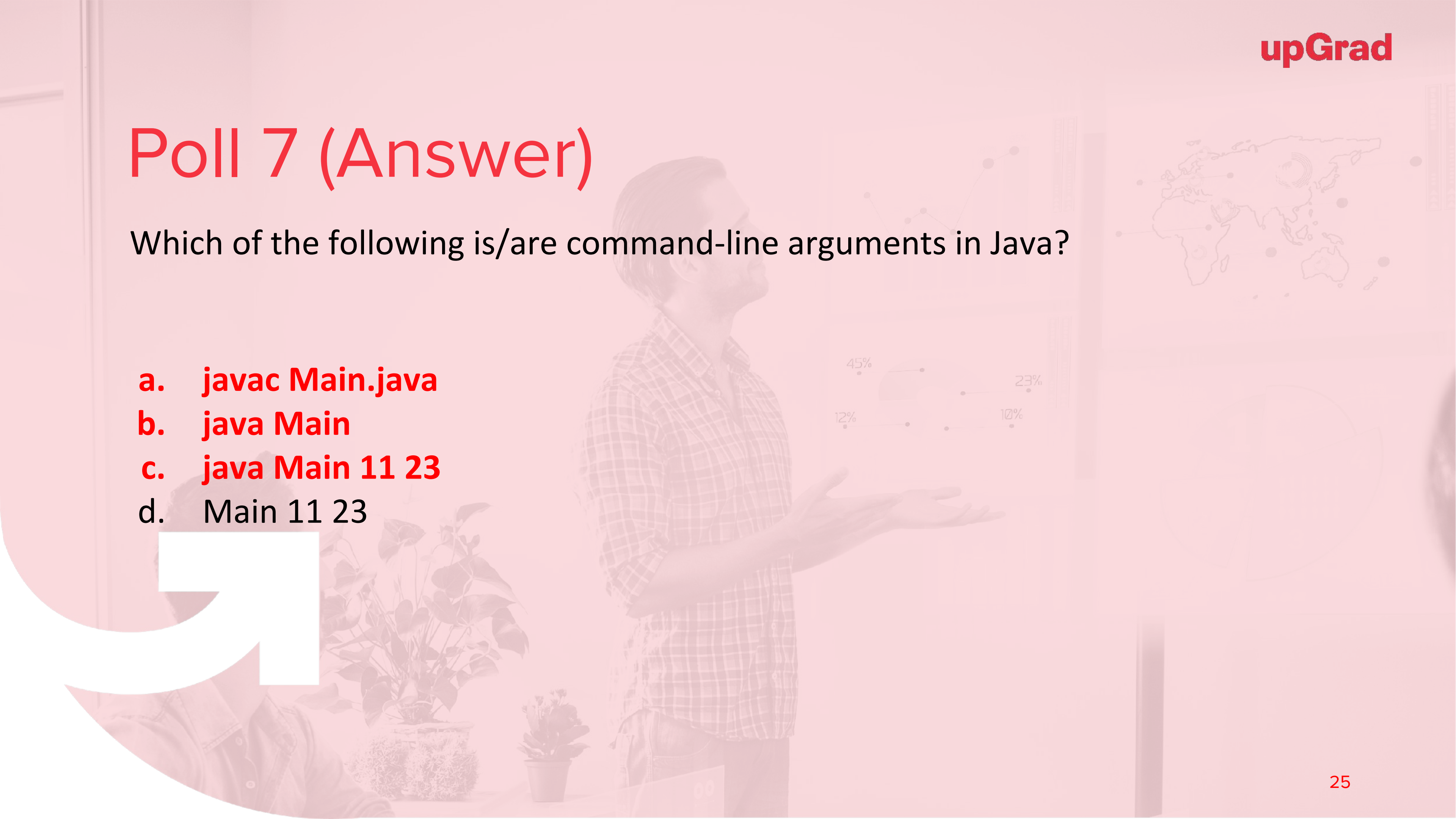
The command line arguments are supplied to JVM, which wraps these and supplies them to args[]. You can be confirm that they are actually wrapped up in an args array by checking the length of the args using args.length.



Poll 7 (15 sec)

Which of the following is/are command-line arguments in Java?

a. javac Main.java
b. java Main
c. java Main 11 23
d. Main 11 23



Poll 7 (Answer)

Which of the following is/are command-line arguments in Java?

a. **javac Main.java**
b. **java Main**
c. **java Main 11 23**
d. Main 11 23

Today's Homework

Write a program to print the following pattern.

```
1*2*3*4*17*18*19*20
 5*6*7*14*15*16
  8*9*12*13
   10*11
```

Input format:

No input required

Output format:

First line containing 1*2*3*4*17*18*19*20

Second line containing 5*6*7*14*15*16

Third line containing 8*9*12*13

Fourth line containing   10*11

upGrad

Key Takeaways

1 Use of different types of loops in Java with examples.

2 To come out of a loop, use break statement.

3 To continue to the next iteration, use continue statement.

4 Basics of methods, including defining the method, defining variables inside methods, calling methods and returning values from methods

Tasks to Complete After the Session

Homework Questions
MCQs
Coding Questions

upGrad

Next Class

1 Strings: Different types of operations on Strings

2 Arrays: Basics and different types of operations on Arrays

upGrad
#LifeKoKaroLift

Thank you!