NGA

Java Interview Questions and Answers For Automation QAs

Prepared By:

Next Generation Automation Academy

Website: https://www.nextgenerationautomation.com

Question: What is Java?
Answer:
Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

Question: Mention some features of Java?
Answer:
Some of the features which play important role in the popularity of java are as follows:

a) Simple: Java is easy to learn. Eventhough Java is based on C++ , it was developed by eliminating poor programming practices of C++.
b) Object-Oriented: Java is a object oriented programming language. Everything in Java is an Object.
c) Portable: Java run time environment uses a bytecode verification process to make sure that code loaded over the network doesn't violate Java security constraints.
d) Platform independent: Java is platform independent. Java is a write once, run anywhere language. Without any modifications, we can use a program in different platforms.
e) Secured: Java is well known for its security. It delivers virus free systems.
High Performance: Java enables high performance with the use of JIT (Just-In-Time) compilers
f) Multithreaded: Java Multithreaded features allows us to write programs that can perform many tasks simulatenously. Multithreading concept of Java shares a common memory area. It doesn't occupy memory for each thread.

Question: What is the difference between Declaration and Definition in Java?
Answer:
Declaration: If you just declare a class or method/function or variable without mentioning anything about what that class or method/function or variable looks like is called as declaration in Java.

Definition: If you define how a class or method/function or variable is implemented then it is called definition in Java.

When we create an interface or abstract class, we simply declare a method/function but not define it.

Question: What is an Object in Java?
Answer:
An object is an instance of a class. Objects have state (variables) and behavior (methods).

Example: A dog is an object of Animal class. The dog has its states such as color, name, breed known as variables, and behaviors such as barking, eating, wagging her tail.

Syntax:

```
1.  public class MyClass{    //Class name (MyClass) declaration
2.      public static void main(String[] args){
3.          MyClass obj = new MyClass(); //Object Creation
4.      }
5.  }
```

Question: What is a Class in Java?
Answer:
A class can be defined as a collection of objects. It is the blueprint or template that describes the state and behavior of an object.

```
1.  public class MyClass{    //Class name (MyClass) declaration
2.  int a = 9;   // Variable declaration
3.  int b = 99;
4.      public void myMethod(){ //Method (myMethod) declaration
5.          int sum=a+b;
6.      }
7.  }
```

Question: What is Constructor in Java?
Answer:
Constructor in Java is used in the creation of an Object that is an instance of a Class. Constructor name should be same as class name. It looks like a method but its not a method. It wont return any value. We have seen that methods may return a value. If there is no constructor in a class, then compiler automatically creates a default constructor.

Question: What is Local Variable, Instance Variable & Class variable in Java?
Answer:
Local Variable:
Local variable is a variable which we declare inside a Method. A method will often store its temporary state in local variables.

Instance Variable (Non-static):
Instance variable is a variable which is declared inside a Class but outside a Method. We don't declare this variable as Static because these variables are non-static variables.

Class Variable (Static):
Class variable is a variable which is declared as Static. Additionally, the keyword final could be added to include that the value will never change.

Example:

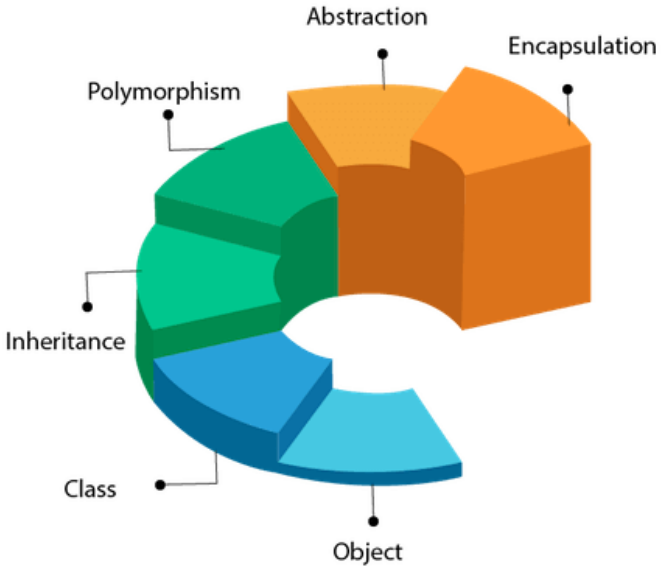
```
1.  package nextGenerationAutomationLearnJava;
2.
3.  public class VariablesInstanceClass {
4.
5.   static int staticVar = 100; // static variable
6.   int instanceVar = 200; // instance variable
7.
8.   public static void main(String [] args){
9.
10.  int localVar = 300; // local variable
11.
12.  // We can access static variables without creating an Object of a class
13.  System.out.println("Value of a Static Variable is "+staticVar);
14.
15.  // Creating an instance of a class 'VariablesLocalInstanceClass2'
16.  VariablesInstanceClass var = new VariablesInstanceClass ();
17.
18.  System.out.println("Value of a Instance Variable is "+var.instanceVar);
19.  System.out.println("Value of a Local Variable is "+localVar);
20.
21.  var.instanceVar = 201;
22.  System.out.println("Updated value of a Instance Variable is "+var.instanceVar);
23.   }
24.  }
```

Question: What are the OOPs concepts?
Answer:
OOPS Stands for Object Oriented Programming System. It includes Abstraction, Encapsulation, Inheritance, Polymorphism, Interface etc.,

NGA

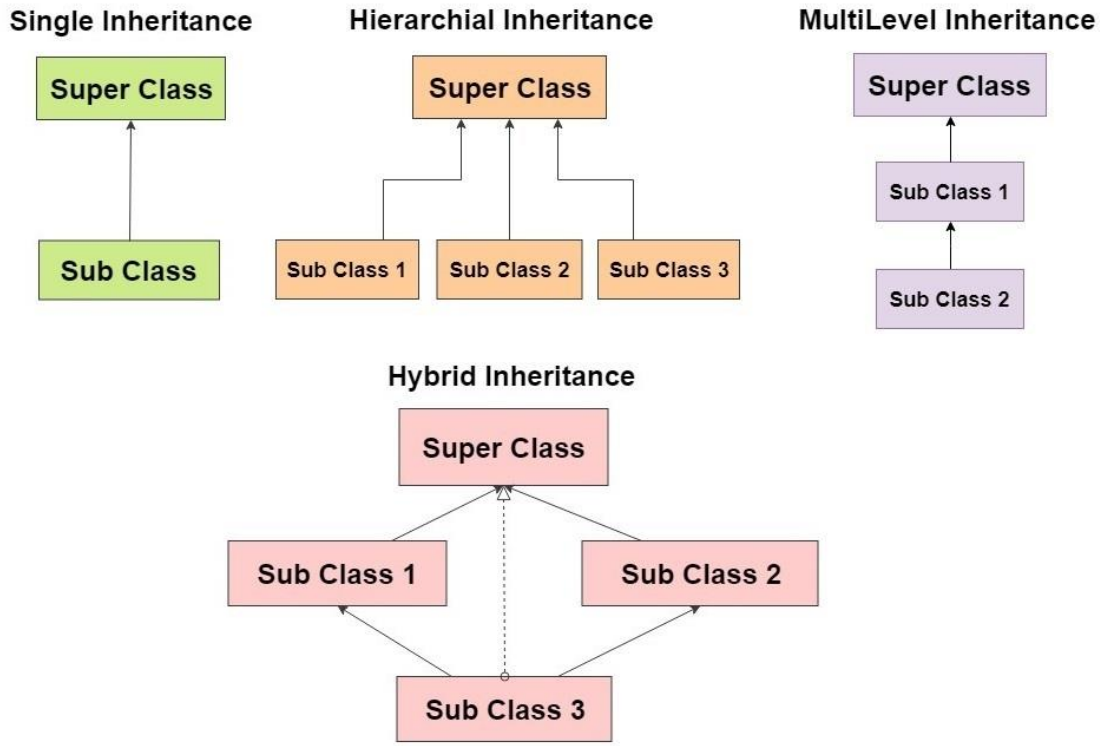
NGA

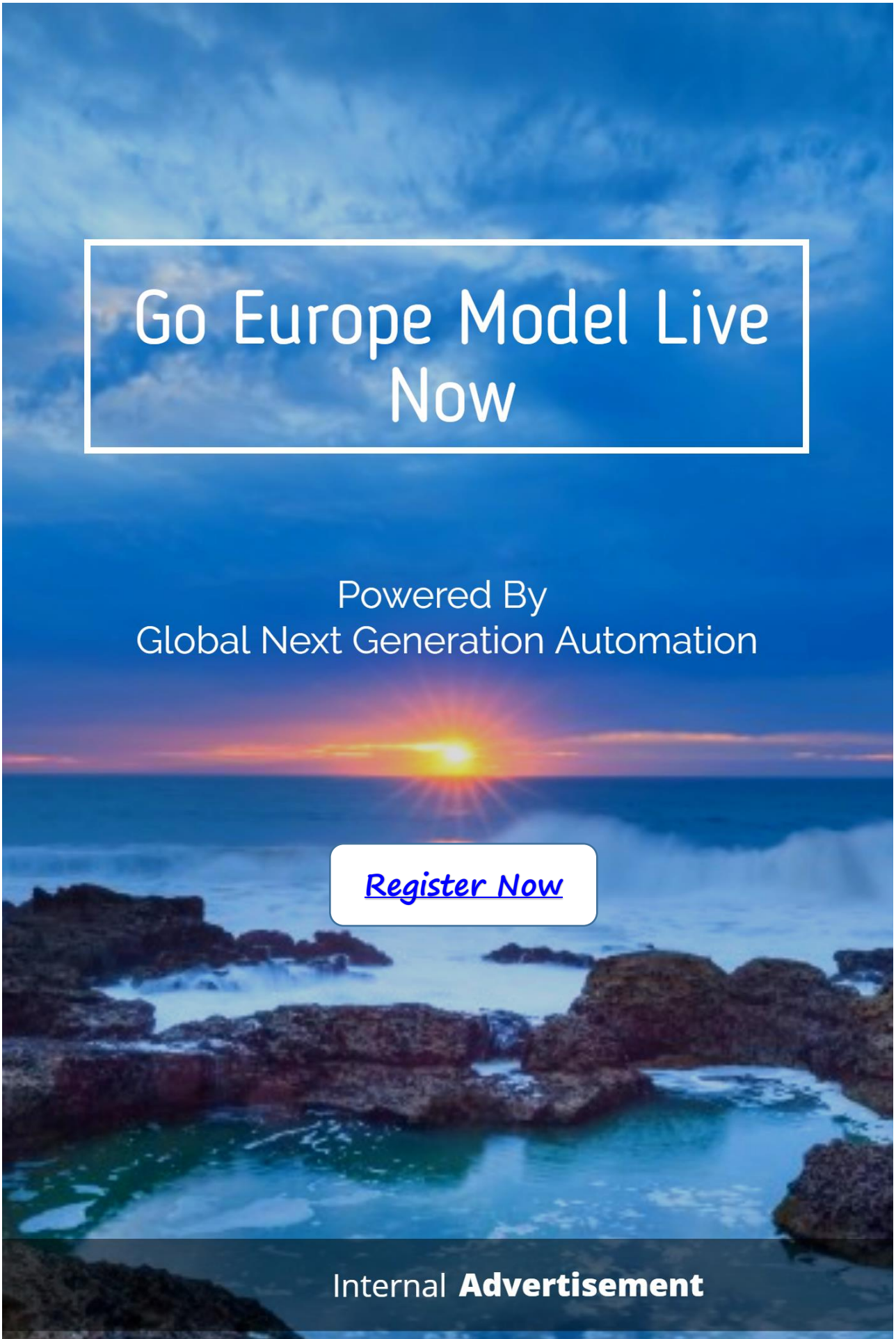
OOPs (Object-Oriented Programming System)



Question: What is Inheritance in Java?
Answer:
Inheritance is a process where one class inherits the properties of another class

NGA




Go Europe Model Live Now
Powered By
Global Next Generation Automation
Register Now
Internal Advertisement

Question: What is Polymorphism?
Answer:
Polymorphism allows us to perform a task in multiple ways. Let's break the word Polymorphism and see it, 'Poly' means 'Many' and 'Morphos' means 'Shapes'

In picture below, Same Gentleman can take different roles like Father, Employee, Shopper as per need etc.

Question: What are the types of Polymorphism?
Answer:
There are two types of Polymorphism in Java
1. Compile time polymorphism (Static binding) – Method overloading
2. Runtime polymorphism (Dynamic binding) – Method overriding
We can perform polymorphism by 'Method Overloading' and 'Method Overriding'

Question: What is Method Overloading?
Answer:
A class having multiple methods with same name but different parameters is called Method Overloading. There are three ways to overload a method.
a) Parameters with different data types
b) Parameters with different sequence of a data types
c) Different number of parameters

Question: What is Method Overriding?
Answer:
Declaring a method in child class which is already present in the parent class is called Method Overriding.

In simple words, overriding means to override the functionality of an existing method.

In this case, if we call the method with child class object, then the child class method is called. To call the parent class method we have to use super keyword.

Question: What is Abstraction in Java?
Answer:
Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.

Abstraction In Java
Example: Mobile Phone.

A layman who is using mobile phone doesn't know how it works internally but he can make phone calls.

Question. What is Abstract Class in Java?
Answer:
We can easily identify whether a class is an abstract class or not. A class which contains abstract keyword in its declaration then it is an Abstract Class.

Syntax:
abstract class <class-name>{}

Points to remember:
a) Abstract classes may or may not include abstract methods

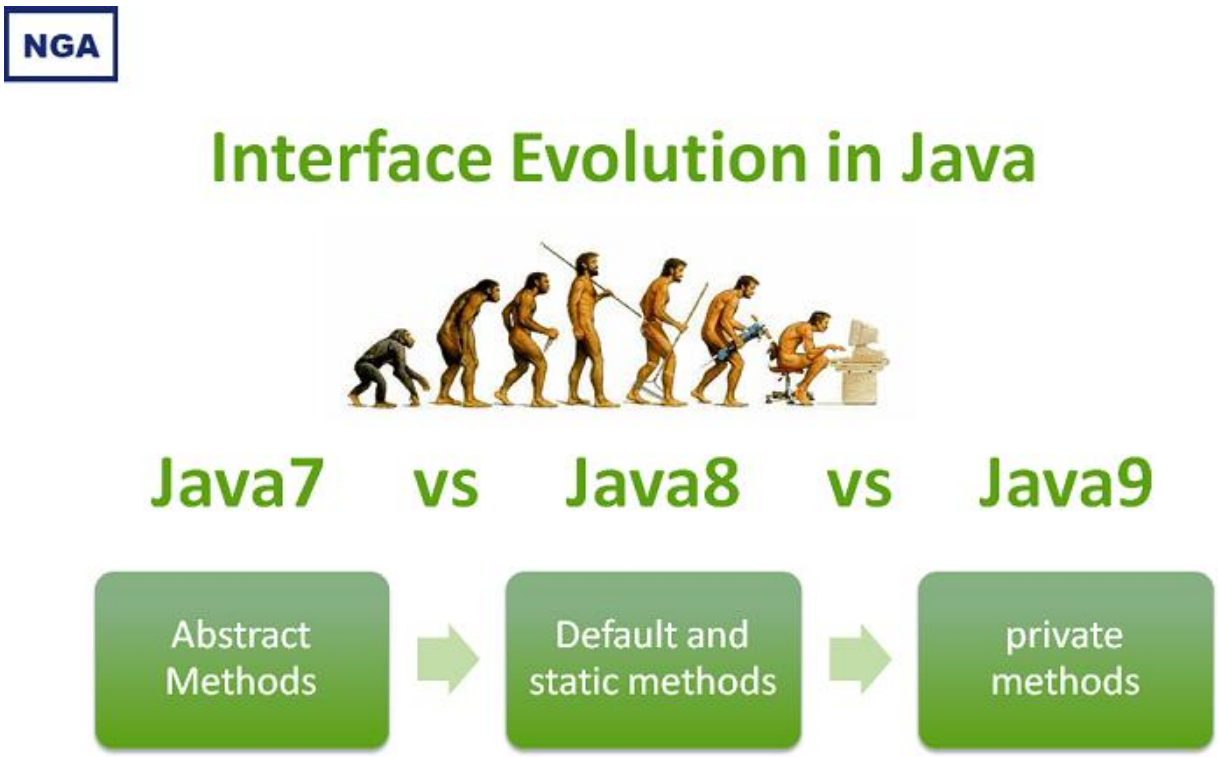
b) If a class is declared abstract then it cannot be instantiated.
c) If a class has abstract method then we have to declare the class as abstract class
d) When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract.

Question: What is Abstract Method?
Answer:
An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

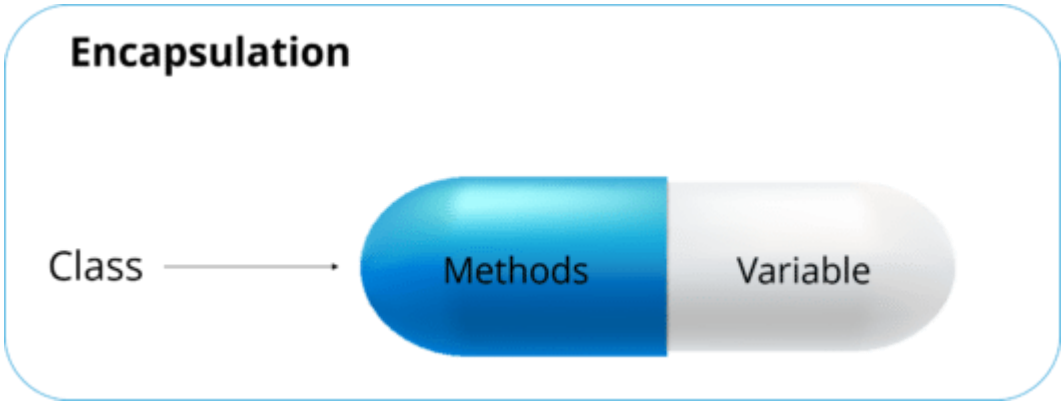
abstract void myMethod();

In order to use an abstract method, you need to override that method in sub class.

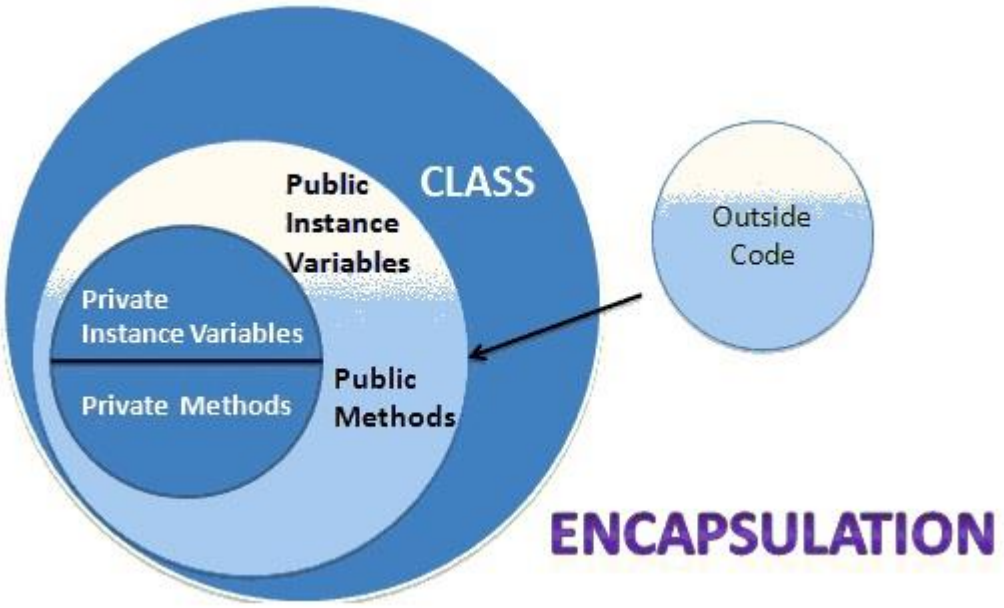
Question: What is Interface in Java?
Answer:
An interface in Java looks similar to a class but both the interface and class are two different concepts. An interface can have methods and variables just like the class but the methods declared in interface are by default abstract. We can achieve 100% abstraction and multiple inheritance in Java with Interface.



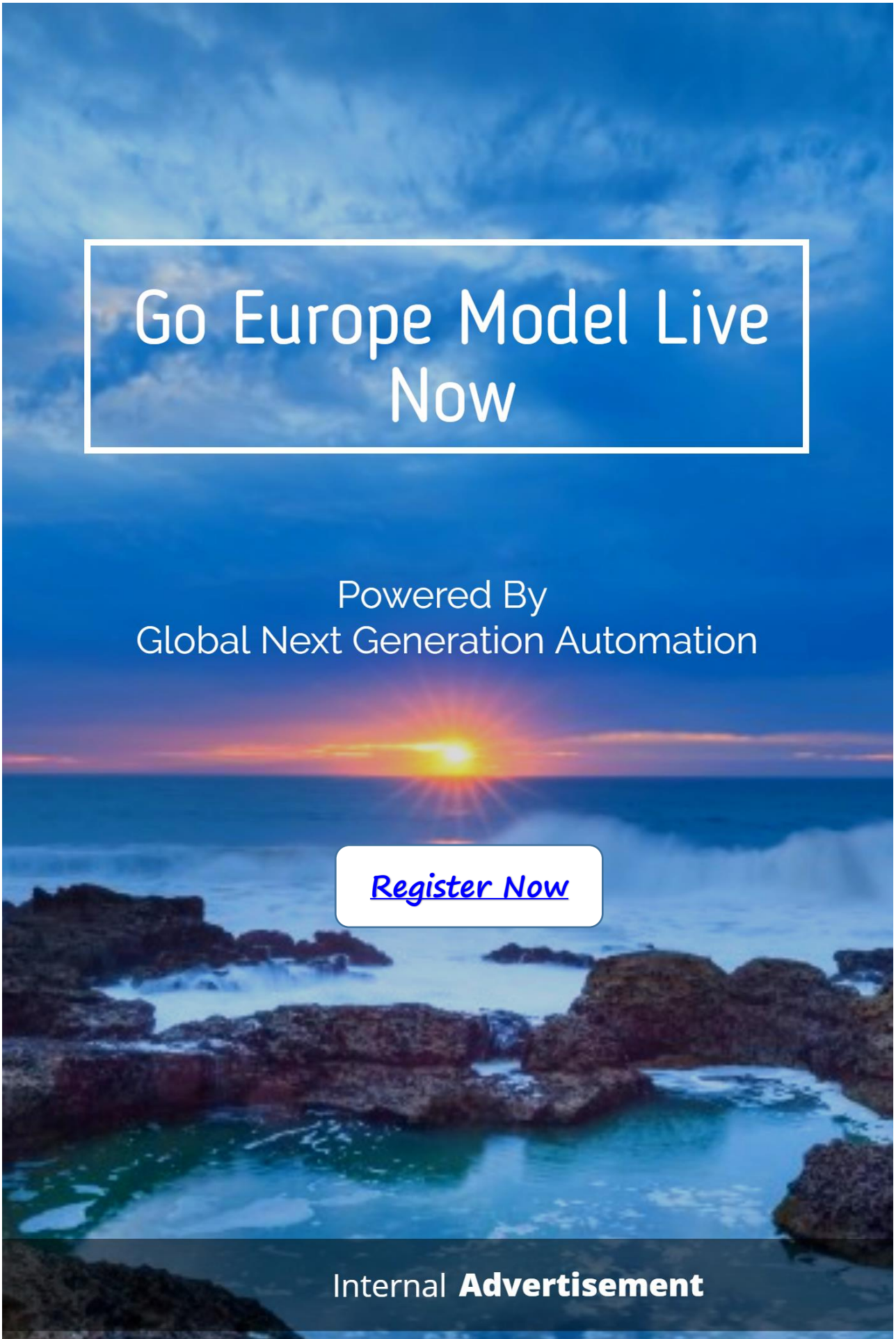
Question: What is Encapsulation in Java?
Answer:
Encapsulation is a mechanism of binding code and data together in a single unit. Let's take an example of Capsule. Different powdered or liquid medicines are encapsulated inside a capsule. Likewise in encapsulation, all the methods and variables are wrapped together in a single class.



Second View



NGA


Go Europe Model Live Now
Powered By
Global Next Generation Automation
Register Now
Internal Advertisement

Question: Write a program to print the pattern given below
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```
1.  package nextGenerationAutomationLearnJava;
2.
3.  public class NumberPattern {
4.      public static void main(String[] args) {
5.          for (int x = 1; x <= 5; x++) {
6.              for (int y = 1; y <= x; y++) {
7.              System.out.print(y+" ");
8.              }
9.          System.out.println();
10.         }
11.     }
12. }
```

Question: Write a program to print Fibonacci Series up to count 10.
Answer:

```
1.  package nextGenerationAutomationLearnJava;
2.
3.  public class FibonacciSeries {
4.      public static void main(String args[]) {
5.          int a = 0, b = 1, c, i, count = 10;
6.          // To print 0 and 1
7.          System.out.print(a + " " + b);
8.          // loop starts from 2. We have already printed 0 and 1 in the previous step
9.          for (i = 2; i < count; i++) {
10.             c = a + b;
11.             System.out.print(" " + c);
12.             a = b;
13.             b = c;
14.         }
15.     }
16. }
```

Question: How to reverse a String in Java?
Answer:

```
1.  package nextGenerationAutomationLearnJava;
2.
3.  public class ReverseString {
4.   public static void main(String[] args) {
5.        // Using StringBuffer class
6.      StringBuffer a = new StringBuffer("Next Generation Automation");
7.      // use reverse() method to reverse string
```

```
8.      System.out.println(a.reverse());
9.      }
10.
11. }
```

Second Approach:

```
1.  package nextGenerationAutomationLearnJava;
2.
3.  public class ReverseString {
4.
5.      public static void main(String[] args) {
6.          String input="Next Generation Automation";
7.          StringBuilder input1 = new StringBuilder();
8.          input1.append(input);
9.          input1=input1.reverse();
10.         for (int i=0;i<input1.length();i++)
11.         System.out.print(input1.charAt(i));
12.     }
13.
14. }
```

Question: How To Find The Largest Value From The Given Array.
Answer:

```
1.  package nextGenerationAutomationLearnJava;
2.
3.  public class LargestValue {
4.      public static void main(String[] args){
5.          int[] arr={28,3,15,9,17,4,23,2};
6.          int val=arr[0];
7.          for(int i=0; i<arr.length; i++){
8.              if(arr[i] > val){
9.                  val=arr[i];
10.             }
11.         }
12.         System.out.println("Largest value in the Given Array is "+ val);
13.     }
14. }
```

Question: How to display all the prime numbers between 1 and 100
Answer:

The number which is only divisible by 1 and itself is known as a prime number. For example 2, 3, 5, 7, 11… are prime numbers.

```
1.  package nextGenerationAutomationLearnJava;
2.
3.  public class PrimeNumbersOneToHundred {
4.     public static void main (String[] args){
5.         int i =0;
6.         int num =0;
```

```
7.          String  primeNumbers = "";
8.
9.          for (i = 1; i <= 100; i++){
10.            int counter=0;
11.            for(num =i; num>=1; num--){
12.               if(i%num==0){
13.                   counter = counter + 1;
14.               }
15.            }
16.            if (counter ==2){
17.               primeNumbers = primeNumbers + i + " ";
18.            }
19.         }
20.         System.out.println("Prime numbers from 1 to 100 are :");
21.         System.out.println(primeNumbers);
22.    }
23. }
```

Question: How to display all the prime numbers between 1 and n (n is the number, get the input from user)
Answer:

```
1.  package nextGenerationAutomationLearnJava;
2.
3.  import java.util.Scanner;
4.
5.  public class PrimeNumbersOneToN {
6.     public static void main (String[] args){
7.        Scanner scanner = new Scanner(System.in);
8.        int i =0;
9.        int num =0;
10.       String  primeNumbers = "";
11.       System.out.println("Enter the value of n :");
12.       int n = scanner.nextInt();
13.       scanner.close();
14.       for (i = 1; i <= n; i++)
15.       {
16.          int counter=0;
17.          for(num =i; num>=1; num--)
18.          {
19.         if(i%num==0)
20.         {
21.         counter = counter + 1;
22.         }
23.      }
24.      if (counter ==2)
25.      {
26.         primeNumbers = primeNumbers + i + " ";
27.      }
28.       }
29.       System.out.println("Prime numbers from 1 to n are :");
30.       System.out.println(primeNumbers);
31.    }
32. }
```

Question: How to find the given number is a prime number or not by getting input from the user ?
Answer:

```
1.  package nextGenerationAutomationLearnJava;
2.
3.  import java.util.Scanner;
4.
5.  public class PrimeNumberVerification {
6.      public static void main(String args[])
7.      {
8.          int i, j, flag = 0;
9.          System.out.print("Enter any number which you want to verify whether it is a prime
10.         number or not :");
11.         Scanner s = new Scanner(System.in);
12.         j = s.nextInt();
13.         for( i = 2; i < j; i++){
14.             if(j % i == 0){
15.                 flag = 0;
16.                 break;
17.             }
18.             else
19.             {
20.                 flag = 1;
21.             }
22.          }
23.          if(flag == 1){
24.              System.out.println(j+" is a prime number.");
25.          }
26.          else{
27.              System.out.println(+j+" is not a prime number.");
28.          }
29.     }
30. }
```

Question: Write a program to print Fibonacci Series?
Answer:

Method 1:

```
1.  package nextGenerationAutomation;
2.
3.  public class FibonacciSeries {
4.      public static void main(String args[]) {
5.          int a = 0, b = 1, c, i, count = 10;
6.
7.          // To print 0 and 1
8.          System.out.print(a + " " + b);
9.          // loop starts from 2. We have already printed 0 and 1 in the previous step
10.         for (i = 2; i < count; i++) {
11.             c = a + b;
12.             System.out.print(" " + c);
13.             a = b;
14.             b = c;
```

```
        }
    }
}
```

Method 2:

```
package nextGenerationAutomation;

import java.util.Scanner;

public class FibonacciSeriesOne {
    public static void main(String[] args){
        System.out.println("Enter Iteration to print Fibonacci Series");
        FibonacciCheck.checkFibonacci(new Scanner(System.in).nextInt());
    }
}

class FibonacciCheck {
    public static void checkFibonacci(int number){
        int first=0,second=1;
        int third=0;
        int i=1;
        System.out.print("Fibonacci Series upto: "+number+" is ");
        System.out.print(first+","+second+",");
        while(i<=number){
            third=first+second;
            System.out.print(third+",");
            first=second;
            second=third;
            ++i;
        }
    }
}
```

Question: Difference between Array and ArrayList?
Answer:
Array:

- Array is static
- Size of the array should be given at the time of array declaration. We cannot change the size of array after creating it
- Array can contain both primitive data types as well as objects
- Arrays are multidimensional

ArrayList:

- ArrayList is dynamic
- Size of the array may not be required. It changest the size dynamically. Capacity of ArrayList increases automatically whenever we add elements to an ArrayList
- ArrayList cannot contain primitive data types. It contains only objects
- ArrayList is always single dimension

Question: Difference between ArrayList and HashSet in Java?

Answer:
ArrayList:

- ArrayList implements List interface
- ArrayList allows duplicates
- ArrayList is an ordered collection and maintains insertion order of elements
- ArrayList is backed by an Array
- ArrayList is an index based

In ArrayList, we can retrive object by calling get() method or remove object by calling remove() method

HashSet:

- HashSet implements Set interface
- HashSet doesn't allow duplicates
- HashSet is an unordered collection and doesn't maintain insertion order
- HashSet is backed by an HashMap instance
- HashSet is object based

In HashSet, we can't achieve get() method

Question: What are the different access modifiers available in Java?
Answer:
Access modifiers are subdivided into four types such as Default, Public, Private, Protected

a) default: The scope of default access modifier is limited to the package only. If we do not mention any access modifier, then it acts like a default access modifier.

b) private: The scope of private access modifier is only within the classes.

Note: Class or Interface cannot be declared as private

c) protected: The scope of protected access modifier is within a package and also outside the package through inheritance only.

Note: Class cannot be declared as protected

d) public: The scope of public access modifier is everywhere. It has no restrictions. Data members, methods and classes that declared public can be accessed from anywhere.

Question: Difference between static binding and dynamic binding?
Answer:
1. Static binding is also known as early binding whereas dynamic binding is also known as late binding.
2. Determining the type of an object at compile time is Static binding whereas determining the type of an object at run time is dynamic binding
3. Java uses static binding for overloaded methods and dynamic binding for overridden methods.

Question: Difference between Abstract Class and Interface?
Answer:
ABSTRACT CLASS

- To declare Abstract class we have to use abstract keyword
- In an Abstract class keyword abstract is mandatory to declare a method as an abstract
- An abstract class contains both abstract methods and concrete methods(method with body)
- Compiler treats all the methods as abstract by default
- An abstract class provides partial abstraction
- An abstract class can have public and protected abstract methods
- An abstract class can have static, final or static final variables with any access modifiers
- An abstract class can extend one class or one abstract class
- Abstract class doesn't support multiple inheritance

INTERFACE:

- To declare Interface we have to use interface keyword
- In an Interface keyword abstract is optional to declare a method as an abstract.
- An interface can have only abstract methods
- An interface provides fully abstraction
- An interface can have only public abstract methods
- An interface can have only public static final variables
- An interface can extend any number of interfaces
- Interface supports multiple inheritance

Question: What is Multiple Inheritance?
Answer:
If a class implements multiple interfaces, or an interface extends multiple interfaces then it is known as multiple inheritance.

Question: How to create singleton class in java?
Answer:
1. Declare a private constructor to prevent others from instantiating the class.

2. Create the instance of the class either during class loading in a static field/block, or on-demand in a static method that first checks whether the instance exists or not and creates a new one only if it doesn't exist.

Example:
Eagerly Initialized Singleton : Class loading in a static field

```java
package nextGenerationAutomationLearnJava;

public class EagerSingleton {

    /** private constructor to prevent others from instantiating this class */
    private EagerSingleton() {}

    /** Create an instance of the class at the time of class loading */
    private static final EagerSingleton instance = new EagerSingleton();

    /** Provide a global point of access to the instance */
    public static EagerSingleton getInstance() {
        return instance;
    }
}
```

Eagerly Initialized Static Block Singleton: Class loading in a static block

```java
package nextGenerationAutomationLearnJava;

public class EagerStaticBlockSingleton {
private static final EagerStaticBlockSingleton instance;

    /** Don't let anyone else instantiate this class */
    private EagerStaticBlockSingleton() {}

    /** Create the one-and-only instance in a static block */
    static {
        try {
            instance = new EagerStaticBlockSingleton();
        } catch (Exception ex) {
            throw ex;
        }
    }

    /** Provide a public method to get the instance that we created */
    public static EagerStaticBlockSingleton getInstance() {
        return instance;
    }
}
```

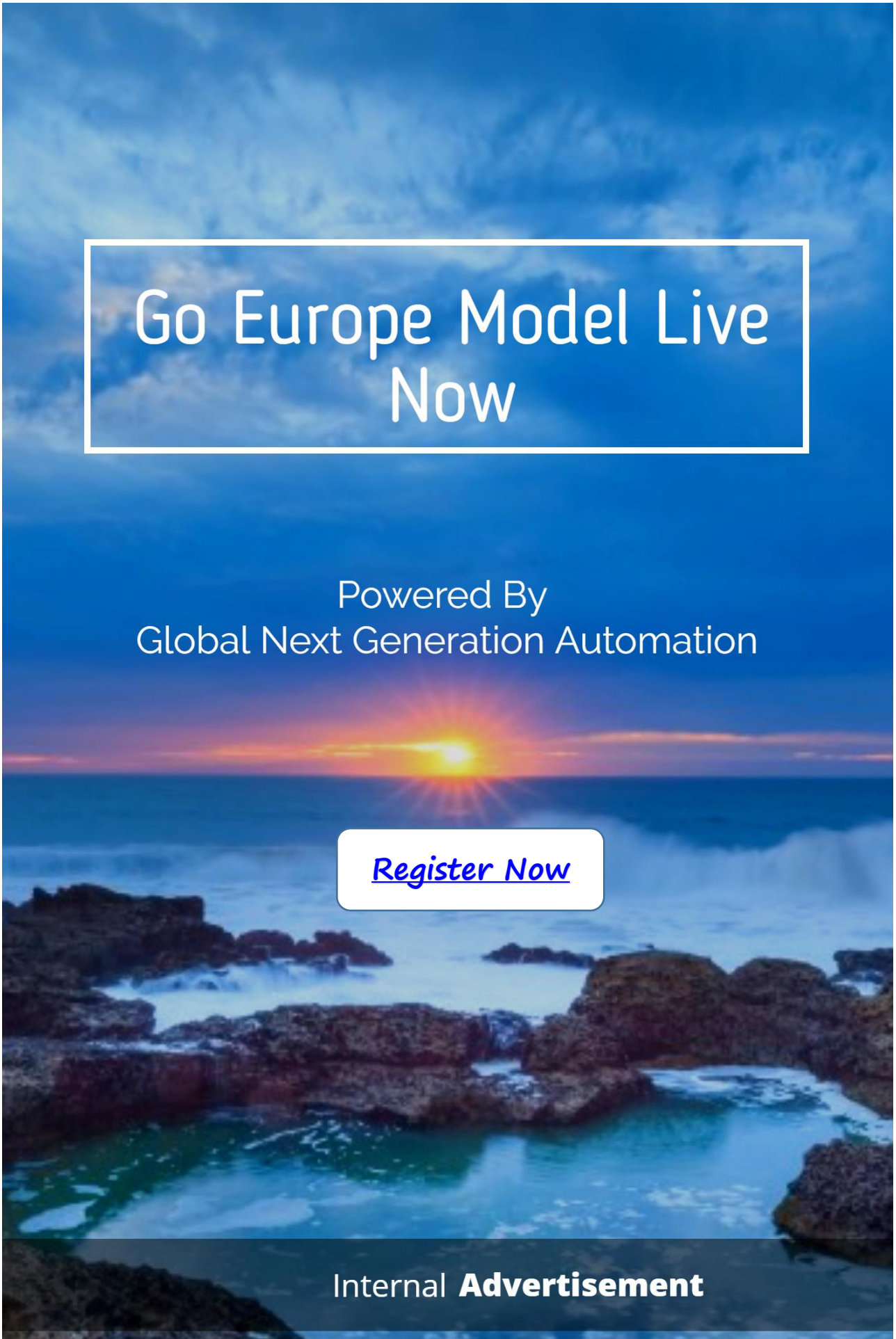
Lazily Initialized Singleton: On-demand in a static method

```java
package nextGenerationAutomationLearnJava;

public class LazySingleton {
    private static LazySingleton instance;
    /** Don't let anyone else instantiate this class */
    private LazySingleton() {}
    /** Lazily create the instance when it is accessed for the first time */
    public static synchronized LazySingleton getInstance() {
        if(instance == null) {
            instance = new LazySingleton();
        }
        return instance;
    }
```

```
14. }
```

#NGAutomation
Building better QA for tomorrow


Go Europe Model Live Now
Powered By
Global Next Generation Automation
Register Now
Internal Advertisement

Thanks!!

Next Generation Automation Academy