```
                                Date : 19/12/2020
                                Spring Boot 9AM
                                  Mr. RAGHU
                        -----------------------------------
Runners Video:
https://www.youtube.com/watch?v=8P5x4DH2WHA
All My Videos
https://www.youtube.com/c/NareshIT/search?query=raghu
Watch List:
https://www.youtube.com/watch?
v=EA43S5R8LSc&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr
                        ----------------------------------
                              Command Line Runners

=> This is a functional interface.
=> We need to define Impl class and override method run(__)
=> Object is created for this class by Spring container.
=> When we run main class/starter class, it calls all runners in
order.
   Executed only once[setup code/test code].

-----------------------------------------------------------------
=> We can even define multiple runners in application.
   They are executed in naming order (A-Z naming rule).

Consider below runners:
 SecurityRunner, MvcRunner and JdbcRunner.

=> They are executed in Naming order.
 a) JdbcRunner
 b) MvcRunner
 c) SecurityRunner

=> We can provide our own execution order using annotation
      @Order("number").
 ** Lowest number order executed first.

--ex--
ARunner   -  @Order(30)
BRunner   -  @Order(12)
CRunner   -  @Order(55)

Execution order is : BRunner, ARunner, CRunner
---------------------------------------------------------------------
=> If we provide -ve numbers even same concept.
 Lowest number is highest priority.

(Hint: in case of -ve numbers, take big number as first priority).

--ex--
ARunner   -  @Order(-51)
BRunner   -  @Order(36)
CRunner   -  @Order(0)
DRunner   -  @Order(12)
ERunner   -  @Order(-25)
```

```
Execution order is : ARunner, ERunner, CRunner, DRunner, BRunner
-----------------------------------------------------------------
****) Incase of multiple runner, if any runner is not assigned with
order
      then executed last in order.

If we give @Order but no value or no annotation even ,then value is :
2147483647
ie un ordered Runners are executed at last (with naming rule applied)

--Ex---
A  - @Order(5)
B  - @Order               // value: 2147483647
C  - @Order(-9)
D  - no annotation even   // value: 2147483647
E  - @Order              // value: 2147483647

Execution flow:
First consider @Order with value : C, A
Next Consider  @Order without value / No annotation (Naming Order): B,
D, E

-----------------------------------------------------------------
*) If runners are given same order, then again naming rule is applied
Z - @Order(5)
M - @Order(5)
A - @Order(5)

Execution flow:- A,M,Z

*** Awlays Execution order is A-Z or -ve -> zero -> +ve
    Reverse is not possible.
======================================================================
=
** It is recomanded to provide order to all runner or for none.

Provide order Priority as:
  @Order(Ordered.LOWEST_PRECEDENCE)   --> last execution
  @Order(Ordered.HIGHEST_PRECEDENCE)  --> first execution


HIGHEST_PRECEDENCE  value  => -2147483648 (very small number in int
range)

LOWEST_PRECEDENCE   value  => 2147483647 (very big number in int
range)

----------------------------------------------------------------------
-
*) We need to execute few set of runners as in order, like

4 Runners - First
3 Runners - Next
6 Runners - Next
2 Runners - Last

=> In this case define one CustomOrder interface use same for
```

```
   our case.

----------code---------------------
#1 Create Spring Starter Project
Name : SpringBoot2MultiRunnersEx

#2. Custom Order interface

package in.nareshit.raghu.order.custom;
public interface CustomOrder {
         int FIRST = -9999;
         int NEXT = -1111;
         int MID = 5555;
         int LAST = 9999;
}

#3. Runner class
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.order.custom.CustomOrder;

@Component
@Order(CustomOrder.MID)
public class AdminRunner implements CommandLineRunner {

        @Override
        public void run(String... args) throws Exception {
                System.out.println("ADMIN RUNNER");
        }

}
---------
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.order.custom.CustomOrder;

@Component
@Order(CustomOrder.MID)
public class AppTestRunner implements CommandLineRunner {

        @Override
        public void run(String... args) throws Exception {
                System.out.println("APP TEST RUNNER");
        }

}
----------
package in.nareshit.raghu.runner;
```

```
import org.springframework.boot.CommandLineRunner;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.order.custom.CustomOrder;

@Component
@Order(CustomOrder.FIRST)
public class DataSourceRunner implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
		System.out.println("DataSource Runner");
	}

}
--------------
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.order.custom.CustomOrder;

@Component
@Order(CustomOrder.FIRST)
public class DbPoolRunner implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
		System.out.println("DB Pool Runner");
	}

}
---------
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.order.custom.CustomOrder;

@Component
//@Order(-909)
//@Order(Ordered.LOWEST_PRECEDENCE)
//@Order(Ordered.HIGHEST_PRECEDENCE)
@Order(CustomOrder.NEXT)
public class JdbcRunner implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
		System.out.println("JDBC RUNNER");
	}

}
```

```
---------
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.order.custom.CustomOrder;

@Component
//@Order(10)
@Order(CustomOrder.LAST)
public class MvcRunner implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
		System.out.println("FROM MVC RUNNER");
	}

}
-------------
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.order.custom.CustomOrder;

@Component
//@Order(20)
@Order(CustomOrder.LAST)
public class SecurityRunner implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
		System.out.println("SECURITY RUNNER");
	}

}
Output:
DataSource Runner
DB Pool Runner
JDBC RUNNER
ADMIN RUNNER
APP TEST RUNNER
FROM MVC RUNNER
SECURITY RUNNER

----------------------------------------------------------------------
----
```