Date : 17/12/2020
Spring Boot 9AM
Mr. RAGHU

YAML (__.yml)

*) YAML - YAMalian Language.
*) This is a file (like Text Format) which stores data
in key=val format (without any duplicate words/levels).

*) But, Still Java supports key-val pairs input using '__.properties'.

*) YAML is a new Format of writing Properties.
*) YAML File internally converted into Properties(C) format only
using API : Snake YAML.

----Note----------
*) Level in Key = Symbol dot(.) in keys creates a new/next level
ex: my.app.code=A
Her my - 1st level in key, app-2nd level in key, code-3rd level in key

*) In realtime we use keys which may have more duplicate levels.
Ex:

```
---application.properties--------------
  spring.datasource.driver-class-name=OracleDriver
  spring.datasource.url=jdbc:oracle
  spring.datasource.username=system
  spring.datasource.password=abc
----------------------------------------
```

```
--application.yml---
spring:
  datasource:
     driver-class-name: OracleDriver
     url: jdbc:oracle
     username: system
     password: abc
---------------------------------
```

==========RULES TO WRITE YAML FILE
==========
1. Replace dot(.) and equals(=) with colon(:)

__.properties

```
my.app.id=10
my.app.code=A
```

__.yml

```
my:app:id:10
my:app:code:A
```

2. After colon , move next level/word to next line
[Do not write duplicates]

__.yml

```
my:
app:
id:10
code:A
```

3. Before every new level(not 1st level) provide spaces (at least one)
   [Space count must match for same level]

```
__.yml
  my:
    app:
       id:10
       code:A
```

4. Finally for value, give exactly one space (between last level and data)

```
__.yml   (final yaml output)
  my:
    app:
       id: 10    //one space only between last level and value.
       code: A
```

1. Replace dot(.) and equals(=) with colon(:)
2. new next level/word goto to next line
   [Do not write duplicates]
3. at least one space / Space count must match for same level
4. after last level and value, give one space
** tab space also valid.

========Examples================================

Ex#1.

```
--application.properties---
my.app.code=A
my.app.model=B
my.grade.service=new
my.grade.cost=600

---application.yml----
my:
  app:
     code: A
     model: B
  grade:
     service: new
     cost: 600
---------------------
```

Ex#2

```
--application.properties---
spring.jpa.show-sql=true
spring.jpa.ddl-auto=create
spring.hikari.size=20
spring.hikari.name=hrk

---application.yml---
```

```
spring:
  jpa:
    show-sql: true
    ddl-auto: create
  hikari:
    size: 20
    name: hrk
--------------------

Ex#3
---application.properties---
spring.jpa.hibernate.auto-create=true
spring.jpa.show-sql=true
spring.jpa.hibernate.format.export=new
spring.jpa.model=entity
spring.jpa.grade.code=accept

---application.yml----
spring:
   jpa:
     hibernate:
       auto-create: true
       format:
         export: new

     show-sql: true
     model: entity
     grade:
       code: accept
---------------------------------

Ex#4
---application.proprties---
my.grade.mode=A
spring.format.text=one
my.accept.mode=new
spring.jpa.show=true
my.grade.state=SA
spring.format.active=true
spring.jpa.final=mode

---application.yml----
my:
  grade:
    mode: A
    state: SA
  accept:
    mode: new
spring:
  format:
    text: one
    active: true
  jpa:
    show: true
    final: mode
```

```
*) if we do not follow proper rules to write __.yml file
 then SnakeYAML API throws data parsing exception like

org.yaml.snakeyaml.scanner.ScannerException:
  mapping values are not allowed here

*) duplicate keys are not allowed in YAML
org.yaml.snakeyaml.constructor.DuplicateKeyException: while
constructing a mapping

----------code----------------------------------
#1 Create Spring Boot Starter Project
Name : SpringBoot2YamlExp

#2. create YAML file under src/main/resource folder
> Right click on 'src/main/resource' folder
> new > file > enter name 'application.yml'

#3. Provide data
--application.yml--
my:
 app:
   id: 9999
   name: YAML DATA
   cost: 9898.3
-------------------

#4. Runner class
package in.nareshit.raghu;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class DataReadRunner implements CommandLineRunner {

        @Value("${my.app.id}")
        private Integer pid;

        @Value("${my.app.name}")
        private String pname;

        @Value("${my.app.cost}")
        private Double pcost;

        public void run(String... args) throws Exception {
                System.out.println(this);
        }

        public String toString() {
                return "DataReadRunner [pid=" + pid + ", pname=" +
pname + ", pcost=" + pcost + "]";
        }

}
-------------------------------------------------------------
```