Name: Sopheak Ratana

Group:ST3.4

Quiz Java

1)Spring Framework គឺជា **Framework** ដែលពេញនិយមសម្រាប់ការអភិវឌ្ឍកម្មវិធី **Java**។ វាត្រូវបានបង្កើតឡើងដើម្បីជួយអភិវឌ្ឍន៍កម្មវិធីដោយមានការងាយស្រួល និងមានប្រសិទ្ធភាព។ ខាងក្រោមគឺជាសមាសភាពសំខាន់ៗរបស់ **Spring Framework** ដែលធ្វើឲ្យវាបានពេញនិយម៖

1. **គំរូនៃ Dependency Injection (DI)៖ Spring** ប្រើប្រាស់គំរូ **DI** ដើម្បីគ្រប់គ្រងការផ្គត់ផ្គង់អ្វីដែលត្រូវការនៅក្នុងកម្រិតនៃការអភិវឌ្ឍ។ វាជួយដោះសោធ្យូកង្វល់ពីការតភ្ជាប់នៃគ្រឿងបន្លាស់នានា និងធ្វើឲ្យការអភិវឌ្ឍងាយស្រួលក្នុងការផ្លាស់ប្តូរ និងសាកសម។
2. **Aspect-Oriented Programming (AOP)៖ Spring** អនុញ្ញាតឲ្យអ្នកអភិវឌ្ឍបានបែងចែកកូដដែលពាក់ព័ន្ធនឹងមុខងារចម្បង **(cross-cutting concerns)** ដូចជាការត្រួតពិនិត្យ និងកំណត់កំណត់ដូចជាលំនាំផ្សេងៗ ដែលធ្វើឲ្យការអភិវឌ្ឍដំណើរការល្អប្រសើរ។
3. **Spring MVC**៖ វាជាគំរូនៃការអភិវឌ្ឍន៍ **Web** ដែលផ្តល់ឱ្យអ្នកអភិវឌ្ឍន៍នូវឧបករណ៍ដើម្បីបង្កើតកម្មវិធី **Web** ដែលមានសមត្ថភាពនិងមានស្ថិរភាព។ វាប្រើប្រាស់មុខងារផ្សេងៗដូចជា **Controller, View,** និង **Model** ដែលអាចធ្វើឲ្យការគ្រប់គ្រងទិន្នន័យ និងការបង្ហាញសម្រួលបាន។

4. **Spring Boot**៖ គឺជាឧបករណ៍សំរាប់បង្កើតនិងបង្កើតកម្មវិធី **Spring** ងាយស្រួលជាងមុន។ វាផ្តល់ឱ្យអ្នកអភិវឌ្ឍន៍នូវការកំណត់ដែលបានបង្កើតរួច និងការកំណត់លំនាំល្អមុនសម្រាប់ការចាប់ផ្តើមនូវកម្មវិធី **Spring**។
5. **Data Access Integration**៖ **Spring** ផ្តល់ឱ្យនូវមុខងារជាច្រើនសម្រាប់ការចូលប្រើទិន្នន័យដូចជា **JDBC, Hibernate, JPA** ដែលអាចជួយគ្រប់គ្រងការចូលប្រើទិន្នន័យបានយ៉ាងមានប្រសិទ្ធភាព។
6. **Transaction Management**៖ **Spring** គាំទ្រដល់ការគ្រប់គ្រងប្រតិបត្តិការនៅក្នុងវិធីសាស្ត្រដែលងាយស្រួលនិងមានភាពខុសគ្នា ដែលធ្វើឱ្យការគ្រប់គ្រងប្រតិបត្តិការសន្តិសុខ និងការបញ្ចូលទិន្នន័យបានប្រសើរឡើង។

Spring Framework ផ្តល់នូវលក្ខណៈពិសេសទាំងនេះដែលធ្វើឱ្យវាជាជម្រើសល្អសម្រាប់ការអភិវឌ្ឍកម្មវិធី **Java** និងមានភាពយឺតយ៉ាវនៃការអភិវឌ្ឍន៍។

2) Spring Boot មានភាពខុសគ្នាពី **Spring Framework** ប្រព័ន្ធធម្មតា និងផ្តល់នូវអត្ថប្រយោជន៍ច្រើនសម្រាប់ការកំណត់គម្រោងថ្មីដោយរហ័ស។ ខាងក្រោមជារបៀបដែល **Spring Boot** ខុសពី **Spring Framework** និងអត្ថប្រយោជន៍ដែលវាផ្តល់:

1. **ការកំណត់និងការកំណត់លំនាំ**៖
    - **Spring Framework:** ប្រើប្រាស់ការកំណត់ច្រើនត្រូវការដើម្បីកំណត់សកម្មភាពផ្សេងៗទាំងអស់របស់កម្មវិធី។ អ្នកត្រូវតែបង្កើតទ្រង់ទ្រាយឬបញ្ចូលការកំណត់ជាច្រើនដែលអាចធ្វើឱ្យវាជាពិបាកសម្រាប់ការចាប់ផ្តើម។

- **Spring Boot:** ផ្តល់នូវការកំណត់លំនាំឬការកំណត់ដែលបានក្រៀមរួច **(auto-configuration)** ដើម្បីជួយកាត់បន្ថយការកំណត់ទាំងនេះ។ វាអាចតំឡើងនិងដំណើរការកម្មវិធីដោយមិនចាំបាច់កំណត់យ៉ាងច្រើន។

2.ការចាប់ផ្តើមបានឆាប់រហ័ស៖

- **Spring Framework:** អ្នកត្រូវតែបង្កើតការកំណត់ដូចជាផ្តត់ផ្តង់ឬការប្រើប្រាស់ទ្រង់ទ្រាយនៃមុខងារ និងកម្រិតដើម្បីដំណើរការកម្មវិធី។
- **Spring Boot:** ផ្តល់ឱ្យអ្នកនូវគំរូនៃការចាប់ផ្តើម **(starter projects)** ដែលអាចជួយអ្នកចាប់ផ្តើមគម្រោងថ្មីបានយ៉ាងលឿន។ វា មានឧបករណ៍ដូចជា **Spring Initializer** ដែលអាចជួយអ្នកបង្កើតគម្រោងដំបូងដោយមានការកំណត់លំនាំផ្តល់ឲ្យ។

3.ការគ្រប់គ្រងអត្ថបទឬសេវាកម្ម៖

- **Spring Framework:** អ្នកត្រូវតែជ្រើសរើស និងកំណត់សេវាកម្មផ្សេងៗដូចជាការបង្ហោះកូដផ្នែខាងក្រោយ។
- **Spring Boot:** មានការគ្រប់គ្រងដោយស្វ័យប្រវត្តិលើអត្ថបទដែលជួយក្នុងការបង្កើត និងដំណើរការកម្មវិធី ដោយមានការគ្រប់គ្រងផ្នែខាងក្រោយក៏ដូចជាសេវាកម្មដែលបានរៀបចំស្រាប់។

4.ការចុះបញ្ជីរចេញផ្សាយ:

- **Spring Framework:** អ្នកត្រូវតែបង្កើត និងបង្កើតការចេញផ្សាយ **(deployment)** យ៉ាងដាច់ខាត។
- **Spring Boot:** ផ្តល់នូវការចុះបញ្ជីសំរាប់ការចេញផ្សាយមួយដែលងាយស្រួលដូចជា **Spring Boot**

executable jar ដែលអាចរត់ដូចជាកម្មវិធី **standalone** ដោយមិនចាំបាច់តំឡើងសេវាកម្មផ្នែកខាងក្រោយទៀត។

5.ការគាំទ្រលើ Microservices:

- **Spring Framework:** ការគាំទ្រលើ **Microservices** ត្រូវតែបង្កើតដោយដៃ និងការផ្គត់ផ្គង់មុខងារដែលត្រូវការជាច្រើន។
- **Spring Boot:** មានការគាំទ្រដល់ការបង្កើត **Microservices** បានយ៉ាងងាយស្រួលជាមួយនឹងការកំណត់លំនាំ និងការគ្រប់គ្រងបន្ថែម។

Spring Boot អាចធ្វើឲ្យការកំណត់គម្រោងថ្នរហ័ស និងងាយស្រួលបំផុត ដោយសារតែវាបានផ្តល់នូវឧបករណ៍ដែលមានភាពប្រសើរនិងការផ្តល់ការកំណត់លំនាំឲ្យបានរហ័ស។

3) @SpringBoot Application គឺជាការបញ្ជាក់សំខាន់មួយនៅក្នុង **Spring Boot** ដែលមានគោលបំណងសំរាប់ការកំណត់ និងការចាប់ផ្តើមកម្មវិធី **Spring Boot**។ វាជួយធ្វើឲ្យការកំណត់ និងការចាប់ផ្តើមកម្មវិធីរបស់អ្នកសាមញ្ញជាងមុន។ ខាងក្រោមគឺជាគោលបំណងនៃ **@SpringBootApplication** និងរបៀបដែលវាធ្វើឲ្យការកំណត់កាន់តែស្រួល៖

គោលបំណងនៃ @SpringBootApplication

1.រួបរួមការបញ្ជាក់ច្រើន៖ @SpringBootApplication បង្ហាញពីការបញ្ជាក់មួយដែលរួបរួមនូវការបញ្ជាក់បីដូចខាងក្រោម:

- **@Configuration:** បញ្ជាក់ថាគ្លាសនេះជាប្រភពនៃការកំណត់ **bean** ហើយត្រូវបានប្រើសម្រាប់ការកំណត់។
- **@EnableAutoConfiguration:** បើកប្រើមុខងារកំណត់ស្វ័យប្រវត្តិរបស់ **Spring Boot** ដែលព្យាយាមកំណត់កម្មវិធីដោយស្វ័យប្រវត្តិឡើងវិញដោយផ្អែកលើអត្ថបទ និងអាជីវកម្មដែលមាននៅលើ **classpath** ។
- **@ComponentScan:** បញ្ជាក់ឲ្យ **Spring** ស្កេនសម្រាប់អង្គភាព និងសេវាកម្មនៅក្នុងកញ្ចប់ដែលកម្មវិធីមានទីតាំងនិងសេវាកម្មនៅក្នុងកញ្ចប់បំព្រងនោះ។

2. **ចំណុចចាប់ផ្តើមកម្មវិធី៖** វាបង្ហាញពីគ្លាសដែលនឹងត្រូវប្រើសម្រាប់ដំណើរការកម្មវិធី **Spring Boot**។ គ្លាសនេះមានវិធី **main** ដែលជាចំណុចចាប់ផ្តើមសម្រាប់កម្មវិធី។ វាហៅ **SpringApplication.run** ដើម្បីចាប់ផ្តើមកម្មវិធី។

របៀបដែលវាធ្វើឲ្យការកំណត់ស្រួល

1. **ការកំណត់ស្វ័យប្រវត្តិ៖** ការបញ្ជាក់ **@EnableAutoConfiguration** នៅក្នុង **@SpringBootApplication** បើកប្រើមុខងារកំណត់ស្វ័យប្រវត្តិរបស់ **Spring Boot**។ នេះមានន័យថា **Spring Boot** នឹងព្យាយាមកំណត់កម្មវិធីដោយស្វ័យប្រវត្តិផ្អែកលើអត្ថបទ និងអាជីវកម្មដែលមាននៅលើ **classpath**។ ឧទាហរណ៍ បើអ្នកបញ្ចូលអត្ថបទសម្រាប់មូលដ្ឋានទិន្នន័យ, **Spring Boot** នឹងកំណត់

DataSource និង **bean** ដែលទាក់ទងជាមួយទាំងនោះ ដោយស្វ័យប្រវត្តិ។

2. **ការស្កេនគ្រប់គ្រងបញ្ចូល៖** ការបញ្ជាក់ **@ComponentScan** នៅក្នុង **@SpringBootApplication** ធ្វើឲ្យ **Spring Boot** ស្កេនគ្រប់គ្រងសម្រាប់អង្គភាព និងសេវាកម្មនៅក្នុងកញ្ចប់ដែលកម្មវិធីមានទីតាំង និងសេវាកម្មនៅក្នុងកញ្ចប់បំព្រងនោះ។ នេះធ្វើឲ្យការស្កេននិងការចុះបញ្ជី **bean** ជាការស្រួលជាងមុន។

3. **ការកំណត់មផ្សំមណ្ឌលៈ** ដោយរួបរួមការបញ្ជាក់ទាំងនេះចូលក្នុងការបញ្ជាក់តែមួយ, **@SpringBootApplication** បង្កើតការកំណត់ក្នុងតែមួយបែប។ នេះធ្វើឲ្យមានការកំណត់សាមញ្ញ និងបន្ថយកូដដែលត្រូវការសម្រាប់ការចាប់ផ្តើមគម្រោងថ្មី។

4. **ការចាប់ផ្តើមបានស្រួលៈ** ពេលអ្នកប្រើ **@SpringBootApplication,** កម្មវិធីត្រូវបានកំណត់ជាមួយនឹងតម្រូវការមានភាពសមស្រប និងអ្នកអាចដាក់ឲ្យមានការកំណត់នាពេលក្រោយតាមកំណត់ដែលត្រូវការ។ វាអនុញ្ញាតឲ្យមានការចាប់ផ្តើម និងការកំណត់សំរាប់ការងារយ៉ាងលឿន។

5. **វិធីសាស្ត្រចាប់ផ្តើមសាមញ្ញៈ** គ្លាសដែលមានការបញ្ជាក់ **@SpringBootApplication** មានវិធី **main** ដែលជាចំណុចចាប់ផ្តើមសម្រាប់កម្មវិធី **Spring Boot**។ ការហៅ **SpringApplication.run** ជាការចាប់ផ្តើមនៃ **Spring application context** និងការចាប់ផ្តើមកម្មវិធីក្នុងមួយខ្សែអក្សរ, ដែលធ្វើឲ្យមានភាពងាយស្រួលក្នុងការចាប់ផ្តើម។

នៅក្នុងឧទាហរណ៍នេះ:

- ការបញ្ជាក់ **@SpringBootApplication** ត្រូវបានប្រើដើម្បីបញ្ជាក់ថា នេះជាគ្លាសចម្បងរបស់កម្មវិធី **Spring Boot**។
- វិធី **main** ហៅ **SpringApplication.run** ដើម្បីចាប់ផ្តើមកម្មវិធី។

សរុបមក, **@SpringBootApplication** បានធ្វើឲ្យការកំណត់កម្មវិធីកាន់តែស្រួលដោយការរួបរួមការបញ្ជាក់សំខាន់ៗ និងការផ្តល់នូវការកំណត់ស្វ័យប្រវត្តិ។

4)បច្ចេកទេស **Dependency Injection (DI)** គឺជាគម្រងមួយសំខាន់ក្នុង **Spring Framework** ដែលជួយបង្កើតកម្មវិធីដែលងាយស្រួលក្នុងការរក្សាទុក និងតេស្ត។ ខាងក្រោមគឺជាការពិពណ៌នាអំពីកំណត់ **DI** និងរបៀបដែលវាជួយបង្កើតកម្មវិធីដែលមានស្ថិរភាព និងអាចតេស្តបាន:

គោលការណ៍នៃ Dependency Injection

Dependency Injection គឺជាបច្ចេកទេសដែលប្រើសម្រាប់សម្រេចការប្រកបជាមួយ **Inversion of Control (IoC)** រវាងថ្នាក់និងការពឹងផ្អែករបស់ពួកវា។ វាស្តីពីការផ្តល់នូវអ **obyject** និងសេវាកម្មដែលថ្នាក់ត្រូវការដោយមិនអនុញ្ញាតឲ្យថ្នាក់នោះបង្កើតវា។ វាជាការបង្ហាញពីការតភ្ជាប់ទាបរវាងអង្គភាព។

នៅក្នុង **Spring Framework, DI** ត្រូវបានប្រើសម្រាប់គ្រប់គ្រងការពឹងផ្អែករវាងគម្រោងនានា **(beans)** ក្នុងកម្មវិធី។

5) Spring Boot ផ្តល់នូវវិធីសាស្ត្រចម្បងចំនួនបីសម្រាប់ដំណើរការកម្មវិធី និងធ្វើឲ្យការបង្ហោះកម្មវិធី **Java** ជាសេវាកម្មផ្ទាល់ខ្លួនងាយស្រួល។ ខាងក្រោមជាវិធីសាស្ត្រចម្បង និងរបៀបដែល **Spring Boot** ធ្វើឲ្យការបង្ហោះស្រួលៈ

វិធីសាស្ត្រចម្បងសម្រាប់ដំណើរការកម្មវិធី Spring Boot

1.ការប្រើប្រាស់ Command Line (CLI):

- អ្នកអាចដំណើរការកម្មវិធី **Spring Boot** តាមរយៈ **Command Line** ដោយប្រើប្រាស់ **java -jar** ពីសំណាក់ **JAR** ដែលបានបង្កើតដោយ **Spring Boot**។ ឧទាហរណ៍:

  bash
  Copy code
  java -jar my-spring-boot-app.jar

- វានឹងចាប់ផ្តើមកម្មវិធី **Spring Boot** នៅលើតំបន់បណ្តាញទេពិតដោយស្វ័យប្រវត្តិ។

2.ការប្រើប្រាស់ IDE:

- អ្នកអាចដំណើរការកម្មវិធី **Spring Boot** តាមរយៈ **IDE** ដូចជា **IntelliJ IDEA** ឬ **Eclipse** ដោយប្រើប្រាស់ប៊ូតុង **“Run”** ឬ **“Debug”** ក្នុង **IDE** ដែលធ្វើឲ្យមានការចាប់ផ្តើមកម្មវិធីដោយស្វ័យប្រវត្តិ។

3.ការប្រើប្រាស់ Spring Boot Maven/Gradle Plugin:

- ប្រសិនបើអ្នកប្រើ **Maven** ឬ **Gradle** សម្រាប់ការគ្រប់គ្រងការសាងសង់, អ្នកអាចដំណើរការកម្មវិធី

ដោយប្រើ **Maven Plugin** ឬ **Gradle Plugin**។ ឧទាហរណ៍៖

- **Maven: mvn spring-boot:run**
- **Gradle: ./gradlew bootRun**

របៀបដែល Spring Boot ធ្វើឲ្យការបង្ហោះងាយស្រួល

1.JAR File Executable:

- **Spring Boot** អាចបង្កើត **JAR** ដែលអាចដំណើរការបានដោយឯករាជ្យ **(standalone executable JAR file)** ដែលមានការតំឡើងទាំងអស់និងការបង្កើតជា **META-INF** និង **BOOT-INF** នៃកញ្ចប់ **JAR**។ នេះមានន័យថា អ្នកអាចដំណើរការកម្មវិធី **Spring Boot** ដោយមិនចាំបាច់តំឡើងសេវាកម្មទៀត។ គ្រាន់តែដំណើរការ **java -jar my-spring-boot-app.jar** នោះគឺគ្រប់គ្រាន់។

2.Embedded Server:

- **Spring Boot** មានសេវាកម្មដែលស្ថិតនៅក្នុង **(embedded server)** ដូចជា **Tomcat, Jetty,** ឬ **Undertow** ដែលត្រូវបានចូលរួមជាមួយកម្មវិធី។ នេះអនុញ្ញាតឲ្យអ្នកមិនចាំបាច់តំឡើងសេវាកម្ម **Web Server** ផ្នែកខាងក្រៅឡើយ ហើយអាចដំណើរការកម្មវិធីជា **Web Server** ផ្ទាល់ខ្លួន។

3.Auto-Configuration:

- **Spring Boot** មានមុខងារ **auto-configuration** ដែលអាចកំណត់ការកំណត់និងអាចជួយបង្កើតការកំណត់ទាំងអស់ដែលត្រូវការ

សម្រាប់បង្កើតសេវាកម្ម និងដំណើរការកម្មវិធីបានយ៉ាងងាយស្រួល។ វាធ្វើឲ្យការបង្ហោះកាន់តែស្រួលដោយកំណត់លំនាំសម្រាប់ការចាប់ផ្តើម។

4.Spring Boot Actuator:

- **Spring Boot Actuator** ផ្តល់នូវសេវាកម្មជាច្រើនសម្រាប់ការគ្រប់គ្រង និងមើលទិន្នន័យនៃកម្មវិធី **(metrics, health checks)** ដែលអាចជួយអ្នកក្នុងការបង្ហោះ និងគ្រប់គ្រងសេវាកម្មបានល្អប្រសើរ។

5.Configuration Properties:

- **Spring Boot** អាចកំណត់លំនាំប្រសើរផ្ទាល់ខ្លួនដោយប្រើការកំណត់ពីឯកសារ **application.properties** ឬ **application.yml** ដែលអនុញ្ញាតឲ្យអ្នកកំណត់ការកំណត់ដែលត្រូវការបានយ៉ាងងាយស្រួល។

ដោយសារតែការចុះបញ្ជីឯកសារបញ្ចូល **(embedded)** និងការបង្កើត **JAR** សំរាប់ដំណើរការ**, Spring Boot** អាចធ្វើឲ្យការបង្ហោះកម្មវិធី **Java** ជាសេវាកម្មផ្ទាល់ខ្លួនកាន់តែស្រួល និងអាចដំណើរការបានតាមប្រព័ន្ធនានា។