1.

Stwórz projekt oparty o Spring Boot wykorzystując starter **spring-boot-starter-web** oraz bibliotekę **lombok**. Następnie w projekcie stwórz klasę **DummyLogger**, która na starcie aplikacji zaloguje informację Hello from zadanie1.

```
public class DummyLogger {
}

```

2.

Stwórz projekt oparty o Spring Boot wykorzystując:

- spring-boot-starter-web
- lombok

Wstrzyknij klasę DummyLogger do:

- klasy CommandLineRunnerWithConstructorInjection za pomocą konstruktora
- klasy CommandLineRunnerWithFieldInjection bezpośrednio do pola
- klasy CommandLineRunnerWithSetterInjection za pomocą setera

Implementacje oprzyj o następujące klasy/szkielety klas:

```
@Component
@Slf4j
public class DummyLogger {
  public void sayHello() {
    log.info("hello from DummyLogger");
  }
}
```

```
import org.springframework.boot.CommandLineRunner;

public class CommandLineRunnerWithConstructorInjection implements CommandLineRunner {

  private DummyLogger dummyLogger;

  @Override
  public void run(final String... args) throws Exception {
    dummyLogger.sayHello();
  }
}

```

```
import org.springframework.boot.CommandLineRunner;

public class CommandLineRunnerWithFieldInjection implements CommandLineRunner {

  private DummyLogger dummyLogger;

  @Override
  public void run(final String... args) throws Exception {
    dummyLogger.sayHello();
  }
}
```

```java
public class CommandLineRunnerWithSetterInjection implements CommandLineRunner {

  private DummyLogger dummyLogger;

  @Override
  public void run(final String... args) throws Exception {
    dummyLogger.sayHello();
  }
}

```

3.

Stwórz projekt oparty o Spring Boot wykorzystując:

- spring-boot-starter-web
- lombok

```java
public interface DummyLogger {
  void sayHello();
}

```

Stwórz dwa beany implementujące interfejs DummyLogger. W implementacjach metody sayHello, niech wypisują na ekran dowolny String. Ponadto niech jedna z implementacji będzie oznaczona jako główny bean.

Stwórz beany będące implementacjami interfejsu CommandLineRunner, który odpowiednio:

- wstrzykuje główną implementację interfejsu DummyLogger i na starcie aplikacji wywołuje metodę sayHello
- wstrzykuje drugą implementację interfejsu DummyLogger i na starcie aplikacji wywołuje metodę sayHello
- wstrzykuje wszystkie implementacje interfejsu DummyLogger i na starcie aplikacji wywołuje metodę sayHello na obu implementacjach

4.

Stwórz projekt oparty o Spring Boot wykorzystując:

- spring-boot-starter-web
- lombok

Stwórz 3 beany w klasie o nazwie UtilConfiguration:

- bean o nazwie dummyLogger będący implementacją klasy DummyLogger
- bean o nazwie listUtility będący implementacją klasy ListUtil. Wykorzystaj nazwę metody do ustanowienia poprawnej nazwy beana.
- bean o nazwie stringUtility będący implementacją klasy StringUtil. Metoda tworząca ten bean powinna mieć nazwę stringUtil.

```java
import lombok.extern.slf4j.Slf4j;

@Slf4j
public class DummyLogger {

  private void sayHi() {
    log.info("Hi from zadanie4");
  }
}

```

```java
import java.util.List;

public class ListUtil {

  public int sumElements(final List<Integer> ints) {
    return ints.stream().reduce(0, Integer::sum);
  }
}

```

```java
import java.util.List;
import java.util.stream.Collectors;

public class StringUtil {

  public String formSentence(final List<String> words) {
    return words.stream().collect(Collectors.joining(" ", "", "."));
  }
}

```

5.

Stwórz projekt oparty o Spring Boot wykorzystując:

- spring-boot-starter-web
- lombok

Do poniższego szkieletu definicji klasy WelcomeMessageLogger wstrzyknij za pomocą adnotacji @Value, wartości, które zdefiniujesz w pliku application.properties za pomocą kluczy:

- pl.sagesacademy.welcome.text.value
- pl.sagescademy.welcome.text.enable

Właściwość pl.sagesacademy.welcome.text.value definiuje tekst, który powinien zostać wypisany na starcie aplikacji. Domyślną wartością powinno być none.

Właściwość pl.sagesacademy.welcome.text.enable definiuje, czy tekst zdefiniowany w pl.sagesacademy.welcome.text.value powinien zostać wyświetlony na ekran. Nie ma domyślnej wartości.

Obie właściwości wstrzyknij przy pomocy konstruktora.

```java
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;

@Component
@Slf4j
public class WelcomeMessageLogger {

  private String text;
  private Boolean shouldLog;
}

```

6.

Stwórz projekt oparty o Spring Boot wykorzystując:

- spring-boot-starter-web
- lombok

Zaimplementuj kontroler (REST) wraz z endpointem GET /api/random-number, który zwraca wartość z wykorzystaniem beanu RandomNumberProvider i metody getValue. Niech podczas wywołania kilku żądań HTTP z rzędu, endpoint zwraca inną wartość.

Zadanie wykonaj bez modyfikacji ciała klasy RandomNumberProvider.

```java
import org.springframework.stereotype.Component;
import java.util.Random;

@Component
public class RandomNumberProvider {

    private final int value = new Random().nextInt();

    public int getValue() {
        return value;
    }
}

```

7.

Stwórz projekt oparty o Spring Boot.
W oparciu o przedstawione w prezentacji treści dotyczące AOP, zademonstruj możliwości wykorzystania:
a) adnotacji Before
b) adnotacji After
c) adnotacji AfterReturning
d) adnotacji AfterThrowing
e) adnotacji Around

8.

Stwórz projekt oparty o Spring Boot.
W oparciu o przedstawione w prezentacji treści dotyczące obsługi zdarzeń, zademonstruj możliwości wykorzystania:
a) adnotacji EventListener
b) ApplicationEventPublisher

c) ApplicationEvent
d) ApplicationListener<>
e) rejestracji listener’ów poprzez ConfigurableApplicationContext
f) adnotacji Order w kontekście kolejności obsługi nasłuchiwaczy
g) wyrażenia SpEL