1. Pierwszy projekt - ściągnij puste repozytorium z linka, dodaj zależności do pom.xml oraz utwórz główną klasę. Pamiętaj o odpowiedniej adnotacji.

2. Prosty kontroler - utwórz nowy kontroler, a następnie stwórz następujące endpointy:
    a. **/my-name** - Wyświetli Twoje imię
    b. **/day-of-week** - wyświetli aktualny dzień tygodnia
    c. **/number-of-days** - zwróci liczbę dni, które upłynęły od Twoich urodzin

3. Parametry - utwórz nowy kontroler i stwórz następujące endpointy:
    a. **/transformed-sentence** - Przyjmie parametr **sentence** zawierający zdanie, podzieli je na wyrazy, odrzuci wszystkie zawierające mniej niż 4 litery a pozostałe wypisze wielkimi literami, np. “Maciej ma mniej niż trzydzieści lat” -> [“MACIEJ”, “MNIEJ”, “TRZYDZIEŚCI”]
    b. **/current-date** - przyjmie parametr **format** (domyślna wartość yyyy-mm-dd), a następnie zwróci odpowiednio sformatowaną dzisiejszą datę

4. Ścieżki - przygotuj klasę, która będzie przechowywać następujące informacje o wyrazie:
    a. Ilość liter
    b. Czy zaczyna się od wielkiej litery
    c. Czy zawiera wskazaną literę*

Stwórz endpoint **/word/{givenWord}/info**, który odczyta podane słowo ze ścieżki i zwróci podane informacje. Przyjmij dodatkowy, opcjonalny parametr “letter”, który określi literę do znalezienia w punkcie c. Jeżeli “letter” nie zostanie podany parametr powinien przyjąć wartość true.

5. Dependency injection - stwórz nowy kontroler oraz komponenty odpowiedzialne za wykonywanie poniższych operacji, a następnie stwórz następujące endpointy:
    a. **/calculator/{number}/abs** - przyjmie w ścieżce liczbę, a następnie zwróci jej wartość bezwzględną
    b. **/calculator/quotient** - przyjmie w parametrze dwie liczby i wyświetli wynik dzielenia. Jeżeli dzielną będzie zero, zwróć maksymalną wartość inta

6. Dependency injection refactor - wykonaj następujące kroki
    a. Przenieś logikę z zadania czwartego do osobnych serwisów. Każda operacja sprawdzająca poszczególne parametry powinna otrzymać swój własny serwis.
    b. Utwórz serwis który wykorzysta trzy serwisy powstałe w punkcie a. I na podstawie otrzymanych wyników utworzy oczekiwany obiekt
    c. Wstrzyknij i użyj serwisu w kontrolerze

7. Singleton - utwórz kontroller i serwis odpowiedzialny za zliczanie odwiedzin. Wystaw na zewnątrz dwa endpointy:
    - /page-visited - każde wywołanie tego endpointa powinno zwiększyć licznik odwiedzin
    - /visited-message - z opcjonalnym parametrem “name” z domyślną wartością “Nieznajomy”. Endpoint powinien zwrócić następującą wiadomość “Hello {{name}}. This page was visited {{count}} times.”.

8. Wykonaj następujące kroki:
    a. Utwórz serwis i nazwij go SimpleFileReader.
    b. W momencie tworzenia się obiektu sprawdź czy istnieje plik simple-file.txt oraz czy nie jest on pusty. Jeżeli któryś z warunków nie jest spełniony rzuć IllegalStateException z odpowiednią wiadomością
    c. Uruchom aplikację - uruchomienie nie powinno się powieść
    d. Utwórz plik w root projektu i nazwij go simple-file.txt.
    e. Uruchom aplikację - uruchomienie nie powinno się powieść
    f. Wypełnij plik dowolnym zdaniem.
    g. Ponownie uruchom aplikację - tym razem uruchomienie powinno zakończyć się sukcesem
    h. Utwórz kontroler oraz endpoint **/files/simple** - zwróć zawartość pliku

9. Utwórz konfigurację o nazwie SimpleFileReaderConfiguration. Stwórz dwa beany SimpleFileReader, jeden o nazwie “simpleBean”, drugi o nazwie “easyBean”. simpleBean powinien czytać plik simple-file.txt, a easyBean o nazwie easy-file.txt. Oba beany wstrzyknij do kontrolera utworzonego w zadaniu nr 8. Dorzuć kolejny endpoint **/files/easy**, który wyświetli zawartość pliku easy-file.txt

10. Do application.properties dodaj następujące parametry:
    a. app.surname -> twoje nazwisko
    b. app.date -> dzisiejsza data
    c. app.date-format -> format daty - domyślna wartość yyyy-MM-dd

Stwórz kontroler i wystaw następujący endpoint /welcome. Wykorzystaj utworzone parametry, aby zwrócić wiadomość: “Hello {{surname}}. Today is {{formattedDate}}”. Zatrzymaj aplikację, zmień konfigurację i uruchom ponownie. Tym razem uruchom ją na porcie 9999.

11. Przepisz aplikację z punktu 10 tak aby wykorzystać ConfigurationProperties. Upewnij się, że nazwisko oraz data nie jest pusta. Pamiętaj o domyślnej wartości dla formatu.

12. Napisz aplikację, która pozwoli na sprawdzenie czy za podany lot samolotowy pasażer może otrzymać odszkodowanie. W tym celu:
    a. Wystaw endpoint **/flights/{flightIdentifier}/eligible**
    b. Sparsuj flight identifier wg. Formatu: {departureAirport}-{arrivalAirport}-{number}
    c. Z pliku flights-timetable.csv odczytaj na podstawie numeru lotu opóznienie
    d. Dla wartości:
        i. < 180 zwroc status ineligible
        ii. > 180 zwroc status eligible-delayed
        iii. brakująca wartość - zwróć status eligible-cancelled
    e. Wystaw endpoint **/flights/{flightIdentifier}/potential-compensation**
    f. Sparsuj flight identifier wg. formatu opisanego w punkcie b.
    g. Z pliku distances.csv odczytaj odległość na podstawie lotniska docelowego i wylotowego
    h. Oblicz odszkodowanie wg. Następującego wzoru: $\sqrt{odległość * \pi}$. Wyliczona wartość jest wartością w złotówkach. Jeżeli lot nie jest eligible - wartość powinna wynosić zero.
    i. Jako parametr przyjmij walutę. Przelicz odszkodowanie na odpowiednią walutę. Wspierane waluty
        i. Euro = 4.43
        ii. Złoty = 1
        iii. Dolar = 3.64
        iv. Rubel = 0.05
        v. Brazillian real = 0.73
    j. Domyślna zwracana waluta powinna być konfigurowalna
    k. Waluty oraz ich wartości powinny być konfigurowalne*
    l. Przykłady:
        i. JFK-LHR-20650 -> ineligible, 0
        ii. JFK-LHR-30750 -> eligible-delayed, ~132.1
        iii. JFK-LHR-40850 -> eligible-canceled, ~132.1
        iv. JFK-LHR-50950 -> ineligible, 0