Mål: Få en förståelse för hur en webbserver är uppbyggd, hur HTTP-protokollet fungerar och hur vi kan designa en modulär applikation i Java. Hur hanterar vi förfrågningar till en webb tjänst och hur ger vi ett svar i json format.

Deadline: 13/6 23.55

Redovisningen: Demonstrera applikationen via teams + ladda upp länk till gitrepo för uppgiften på ithsdistans.se.
Om uppgiften utförs i grupp laddar alla deltagare upp länk till koden och skriver en egen text som beskriver vad ni bidragit med i projektet och då speciellt med tanke på g och vg kraven.
Försök hålla koden ren med avseende på namngivning, storlek av metoder och klasser samt testbarhet av koden.

Betygskriterier:

- Integrationer mellan system
- Webbtjänster
- Integrera javasystem med andra system
- Skapa en klient som konsumerar tjänster
- Skapa en webbtjänst som klienter kan anropa
- Identifiera integrationsrelaterade krav
- Skapa ett specificerat interface
- Manuellt testa kod
- Hämta data från externa system
- Bygga och publicera webbtjänster
- Konsumera en webbtjänst

Litteratur: Föreläsningsmaterial och kodexempel från vecka 1 och 2.

Med laborationen vill vi få en ökad förståelse för hur vi kan utveckla en större och komplicerad applikation i Java. Vi kommer att titta speciellt på följande saker:

Modulära applikationer. Maven modules/Java Modules/SPI
Nätverkskommunikation.
Filer.
Trådade applikationer.
HTTP-protokollet för överföring av information.
Köra vår applikation i docker container.
Json-konvertering av information
Koppla oss mot databas

Länkar:
https://www.baeldung.com/a-guide-to-java-sockets
https://www.baeldung.com/thread-pool-java-and-guava
https://www.baeldung.com/java-9-http-client
https://www.baeldung.com/java-io

https://www.baeldung.com/java-9-modularity
https://www.baeldung.com/maven-multi-module-project-java-jpms

Uppgift:

G-nivå

- Arbeta i grupp eller individuellt.
- Versionshantera projektet med Git och publicera på github.
- Konfigurera pom.xml för att kunna bygga med maven som ett multi module projekt med minst 2 moduler.

- Bygg en enkel HTTP-server som använder version 1.1 av protokollet.

- Servern ska vara multitrådad och kunna hantera flera förfrågningar samtidigt utan att problem uppstår.

- Servern ska kunna hantera enklare GET HEAD och POST förfrågningar.

- Kunna serva statiska filer från disk av formaten html, css, js, pdf samt ett eller flera bildformat med rätt content-type satt i headern.
- Hantera default filer tex index.html om url slutar med /

- Det ska finnas minst en url som ger oss möjlighet att skicka in information till webbservern via url parametrar tillsammans med GET samt minst en url med POST och body text.
  - Den inskickade informationen ska lagras i en **databas,** lämpligen MySql och kunna skickas tillbaka som ett **json dokument** när det efterfrågas via en GET mot rätt URL.
  - https://www.baeldung.com/java-json
  - Använd URLEncoding/URLDecoding för url parametrar.

- Om en adress efterfrågas som inte finns ska en 404 sida returneras med rätt felkod.

- Skapa antingen en console applikation eller tester som provkör servern med hjälp av HttpClient.

VG:

- Utöver kraven för G ska:
- Skapa en **Dockerfile** som kan bygga en image av projektet och köra det.

- Webb servern kunna hantera pluginklasser skrivna i Java som kan laddas in dynamiskt med hjälp av **ServiceLoader.**
  - https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/util/ServiceLoader.html
  - Konfiguration av routing som behövs för pluginen ska göras med runtime annotations.
- Skapa minst ett plugin som exponerar en url som går att anropa och som returnerar dynamiskt skapad html eller json som svar.
  - Går även att implementera hela serverns funktionalitet som olika plugin inklusive statiska filer.
  - Använd lämpligen Request/Response objekt som skickas från servern till pluginet.
  - Inspireras gärna av https://jakarta.ee/specifications/servlet/5.0/apidocs/jakarta/servlet/http/HttpServlet.html