**Portfolio eindproject: Web/Mobile application  
  
Puntenverdeling bij rubrics:**

Voorwaarde: alle onderdelen tenminste 1 punt.

36 punten -> 5.5  
54 punten -> 6.5  
72 punten -> 7.5  
90 punten -> 8.5  
108 punten -> 9.5  
Formule: (#punten - 36)/18 + 5.5

**Beoordeling:  
-** Onvoldoende  
- Matig  
- Voldoende: Toepassen/analyseren  
- Goed: Synthetiseren/evalueren  
- Daarnaast kun je ook een bonus behalen.

**Je portfolio kun je (meerdere keren) inleveren op:**  
<https://dlo.mijnhva.nl/d2l/le/content/329878/viewContent/1025714/View>

**De verschillende rubrics zijn:**  
- UML  
- Web Applications  
- Software Maintenance  
- Design Patterns

Je mag je portfolio op je eigen manier een invulling geven. Het kan zijn dat je dit eerst wilt overleggen met een docent. Voor sommige rubrics zoals UML zijn concrete opdrachten meegegeven die je kunt gebruiken om aan de rubric te voldoen.

**Op DLO staat het volgende bij sommige CodeLabs over de portfolio:**  
Het is de bedoeling om de uitwerking van de opdrachten met korte reflectie op te nemen in je portfolio. Doe dit in de onderstaande vorm.

| Opdrachtnaam | Geleerde lessen | Problemen | Hoe was je leerervaring en waarom (fantastisch/goed/oke/vreselijk)|

Hieronder staan alle opdrachten uit de CodeLabs , dus kijk welke je kunt gebruiken voor je portfolio. Bij veel staan er ook al uitwerkingen, dus waarschijnlijk zijn die niet te gebruiken, maar je hebt dan in ieder geval voorbeelden.

**CodeLab Internet101 opdrachten:**  
- 6. Opdracht: verschil IPv4 en IPv6  
- 8. Opdracht: intern IP-adres  
- 9. Opdracht: extern IP-adres  
- 11. Opdracht: NS lookup  
- 12. Opdracht: DNS Lokaal (optioneel)  
- 13. Opdracht: Ping  
- 14. Opdracht: Traceroute  
- 16. Opdracht: gereserveerde poorten  
- 17. Opdracht: Java Sockets (Optioneel)

**CodeLab: HTTP101 opdrachten:**- 7. Opdracht: GET request  
- 8. Opdracht: statuscodes  
- 9. Opdracht: headers  
- 11. Opdracht: Query string  
- 12. Opdracht: URL-encoding  
- 13. Opdracht: POST Request  
- 14. Opdracht: Bad Request  
- 15. Opdracht: POST Json

**Web 1 - Codelab: Spring Basis:**  
- 4. Opdracht: Constructor Injectie met uitwerking  
- 7. Opdracht: CommandLineRunner met uitwerking  
- 9. Opdracht: Componentscan met uitwerking  
- 11. Opdracht: Autowired (optioneel)  
  
**Web 1 - Codelab: Spring HTTP-Controller:**  
- 9. Opdracht: User resource  
- 11. Opdracht: RequestMapping  
- 13. Opdracht: ResponseBody met uitkwerking  
- 16. Opdracht: RequestParam met uitkwerking  
- 18. Opdracht: RequestBody met uitkwerking  
- 22. Opdracht: De Cirkel Rond met uitkwerking  
- 23. Opdracht: Fat Controller met uitkwerking  
- 26. Opdracht: UserController compleet CRUDL met uitkwerking

**Web 2 – REST - geen CodeLab beschikbaar**  
  
**Web 3 - Codelab: JdbcTemplate met uitkwerking:**- 3. Opdracht: Project setup – configuratie  
- 4. Opdracht: Project setup - Database maken  
- 5. Opdracht: Project setup - Klasse Klant  
- 10. Opdracht: MedewerkerDAO met implementatie  
- 12. Opdracht: JdbcTemplate update, insert, delete  
- 15. Opdracht aanpassen JdbcKlantDAO  
- 17. Opdracht: NamedParameterJdbcTemplate (Optioneel)  
- 19. Opdracht: HTTP en Database met Controller en JdbcTemplate  
  
**Web 3 - Codelab: Domain mapping en DAO's:**  
- 8. Opdracht: Van DAO naar Repository

**Web 4 - Codelab: Authentication – Security met uitkwerking:**  
- 16. Opdracht: Wachtwoorden opslaan 1 : Hashen  
- 17. Opdracht: Wachtwoorden op slaan 2 : Salt toevoegen  
- 18. Opdracht: Wachwoorden op slaan 3: Pepper toevoegen  
- 19. Opdracht: Service-klassen – overzicht  
- 20. Opdracht: Service-klassen - Pepper Service  
- 21. Opdracht: Service-klassen - Hash Service  
- 22. Opdracht: Service-klassen - Registratie Service  
- 23. Opdracht: Service klassen - Authenticatie Service  
- 25. Opdracht: Token Based Authentication  
- 28. Opdracht: Wachtwoord opnieuw instellen  
- 29. Opdracht: RestController en Authenticatie  
- 35. Opdracht: Learn Java Encryption (optioneel)

**Web 5 - Codelab: HTTP-HTML-FORM met uitwerking:**- 9. Opdracht: eerste pagina  
- 10. Opdracht: Browsertools  
- 11. Opdracht: DOM-inspector  
- 13. Opdracht: Tempconvert FORM  
- 14. Opdracht: Inlogformulier  
- 15. Opdracht: Rekeningoverzicht – Table

**Web 6 - Codelab: CSS Basis met uitkwerkingen:**  
- 3. Opdracht: selecteren – basis  
- 4. Opdracht: selecteren – klassen  
- 5. Opdracht: Flexbox  
- 6. Opdracht: Rekeningoverzicht – Div  
- 7. Opdracht: NavBar (optioneel)

**Web 7 - Codelab: Javascript Basis met uitkwerkingen en uitwerkingen van opdrachten die niet apart als opdrachten aangemerkt zijn:**- 8. Opdracht: typeof operator  
- 11. Opdracht: variabelen  
- 13. Opdracht: BMI  
- Opdrachten CSS  
- Opdrachten JavaScript  
  
**Web 8 - Codelab: AJAX met uitkwerkingen:**  
- 5. Opdracht: Postcode  
- 7. Opdracht: CORS  
- 9. Opdracht: Postcode en Weer (Optioneel)

**UML (workshop UML 1 en 2)**UML 1 op 24-11-2021: Use cases diagram, Class diagrams  
UML 2 op 26-11-2021: Sequence diagrams, State diagrams

**Wat je moet kunnen:**Voor de behandelde UML diagrammen (use-case, user-stories, klassen-, sequence-, state-, ~~component-, package-, deployment-diagrammen(~~deze worden niet meer behandeld):   
- meerdere versies van de diagrammen genereren, gegeven een incomplete casus.  
- complexe diagrammen omzetten in java-code  
- diagrammen genereren vanuit complexe java-code  
- evalueren welke diagrammen beter zijn

**Hoe je dit moet bewijzen:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Diagram** | **Leerdoel** | **Suggestie** | **Afvinken** |
| Use-case diagram | Beschrijving => diagram | groeps-opdracht |  |
|  | Beschrijving => use-case scenario | groeps-opdracht |  |
|  | Beschrijving => code | individueel (\*)  opdracht 3 van Makelaar |  |
| Klassen diagram | Beschrijving => diagram | groeps-opdracht |  |
|  | Diagram => code | groeps-opdracht |  |
|  | Code => diagram | individueel (\*)  opdracht 1 van Makelaar |  |
| Sequence diagram | Beschrijving => diagram | individueel |  |
|  | Diagram => code | individueel (\*)  opdracht 2 van Makelaar |  |
|  | Code => diagram | individueel (\*)  opdracht 4 van Makelaar |  |
| State diagram | Beschrijving => diagram | individueel (\*)  opdracht 5 van Makelaar |  |
|  | Diagram => code | optioneel |  |
| Package/component/deployment diagrammen | Beschrijving => diagram | optioneel |  |
|  | Code => diagram | optioneel |  |
| Activity diagram | Beschrijving => diagram | optioneel |  |
|  | Diagram => code | optioneel |  |
|  | Code => diagram | optioneel |  |

**Toelichting:**

* De rechterkolom geeft suggesties hoe je zo'n bewijsstuk kunt toevoegen in je portfolio:  
  - groep: maak een (gezamenlijk) bewijsstuk als onderdeel van je groeps-project  
  - individueel: maak zelfstandig een bewijsstuk (mag bv uit je groeps-project komen). De punten met (\*) zijn ook terug te vinden in de Rubric-opdracht bij de module UML. Zie hieronder.  
  - optioneel: niet verplicht voor het behalen van de rubrics voor deze module maar kunnen bonus opleveren/compenseren voor niet geheel correct gemaakte rubrics in deze module
* Voor elke leerdoel staat:  
  - beschrijving => diagram - maak vanuit de context (probleembeschrijving) een diagram  
  - diagram => code - maak vanuit een diagram bijbehorende (Java) code  
  - code => diagram - maak vanuit (java) code een bijbehorende diagram

**\*** Voor deze rubric kun je dus gebruik maken van de opdracht op DLO en het project Makelaar en de meegegeven UML. Dan heb je alles meegenomen van wat je moet weten/begrijpen over UML voor deze rubric voor je portfolio. Je levert dan het aangepaste project in (opgave 2 en 3) en de modellen (opgave 1, 4 en 5). Let op: bewaar zowel de pdf-bestanden als de uxf-bestanden (Umlet) in je portfolio.

Voor de opdracht zie: <https://dlo.mijnhva.nl/d2l/le/content/329878/viewContent/917372/View>

**Info:**Alle niet-optionele rubrics moeten aan bod komen in je portfolio om te slagen. Hoe goed ze gemaakt zijn bepaalt de inschaling in de rubrics: matig, voldoende, goed.

**Webapplications (workshop Web 1 t/m 9)**Web 1 op 23-11-2021: Spring  
Web 2 op 23-11-2021: REST  
Web 3 op 25-11-2021: Spring JDBC  
Web 4 op 29-11-2021: Security + Authentcation  
Web 5 op 13-12-2021: HTML CSS  
Web 6 op 13-12-2021: HTML CSS  
Web 7 op 14-12-2021: Client side scripting – Javascript Basis  
Web 8 op 16-12-2021: Ajax, Cors  
Web 9 op 11-01-2022: ?

**Wat je moet kunnen:**Kennis hebben van, inzicht in en relevante technieken kunnen toepassen met betrekking tot onderstaande onderdelen:  
- Web applications client side architectuur  
- HTML/DOM, CSS, Javascript, AJAX  
- Structuur en eigenschapen van het Internet  
- MVC-architectuur toepassen in webapplicaties  
- Web applications: Gebruik maken van een webframework

**Hoe je dit moet bewijzen:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Programming skills** | **Leerdoel** | **Suggestie** | **Afvinken** |
| Web applications client side architectuur. | rol leren html, css, javascript, ajax in webapplicaties | individueel: beschrijving wat het werkingsgebied van gegeven onderdelen is mbt webapplicaties |  |
| Client side techieken: HTML/DOM. | opbouw webpagina van html-bestand naar representatie in het geheugen van de browser, relatie css en javascript mbt DOM | individueel: beschrijving vertaling HTML naar de DOM met rollen html, css en javascript hierin verwerkt |  |
| Client side techieken: CSS. | Webpagina stylen met CSS | groepsopdracht: ieder groepslid maakt (deel) css eigen pagina en is duidelijk onderscheidbaar in git commits **of**  individueel: in eigen project |  |
| Client side techieken: Javascript | Wegpagina dynamisch aanpassen met behulp van javascript | groepsopdracht: ieder groepslid maakt (deel) javascript eigen pagina en het werk is duidelijk onderscheidbaar in git commits **of** individueel: in eigen project |  |
| Client side techieken: AJAX | Data asynchroon ophalen en opslaan en met opgehaalde data de pagina dynamisch aanpassen | groepsopdracht: ieder groepslid verwerkt AJAX eigen pagina en is duidelijk onderscheidbaar in git commits **of** individueel in eigen project |  |
| Web applications: Structuur en eigenschapen van het Internet | Het leren werken met het HTTP-protocol, netwerken, ip-adressen, DNS en netwerktools | individueel: screenshots van het werken met ping, traceroute, curl, Postman met beschrijving wat er gebeurt en waar de tools voor gebruikt worden. |  |
| Web applications: Spring Boot MVC | Het leren werken met een framework om webapplicaties te maken | groepsopdracht **of** individueel: in eigen project |  |
| Web applications: Model | Het leren werken met een domeinlaag in een MVC-framework.  Het leren werken met een ORM en domeinlaag. | groepsopdracht **of**  individueel: in eigen project |  |
| Web applications: View | Het leren werken met een viewlaag in een MVC-framework.  Het leren werken met templating engine om html-pagina's te genereren | groepsopdracht: ieder groepslid eigen pagina en is duidelijk onderscheidbaar in git commits **of** individueel: in eigen project |  |
| Web applications: Controller | Het leren werken met een controllerlaag in een MVC-framework. | groepsopdracht: ieder groepslid heeft een method in een controller gemaakt, of controller bijhorend eigen pagina en is duidelijk onderscheidbaar in git commits **of** individueel: in eigen project |  |

**Toelichting:**

* De rechterkolom geeft suggesties hoe je zo'n bewijsstuk kunt toevoegen in je portfolio:   
  - groep: maak een (gezamenlijk) bewijsstuk als onderdeel van je groepsproject  
  - individueel: maak zelfstandig een bewijsstuk (mag bv uit je groepsproject komen).  
  - optioneel: niet verplicht voor het behalen van de rubrics voor deze module maar kunnen bonus opleveren/compenseren voor niet geheel correct gemaakte rubrics in deze module

**Info:**Alle niet-optionele rubrics moeten aan bod komen in je portfolio om te slagen. Hoe goed je de rubrics maakt bepaalt de beoordeling: matig, voldoende, goed.

**Software Maintenance (workshop software maintenance 1 en 2)**Design Patterns 1 op 17-12-2021: Strategy, Pattern catalogus, Case remote control, Singleton, Pattern talks  
Design Patterns 2 op 20-12-2021: Observer, MVC, Abstract factory, Decorator  
Design Patterns 3 op 10-01-2022: ?  
 **Wat je moet kunnen:**- Richtlijnen 1 t/m 8 en 10 volledig kunnen toepassen in (eigen) code (beheersen van de mogelijke oplossingen per richtlijn).  
- Concreet advies kunnen uitbrengen hoe aan de richtlijnen 1 t/m 8 en 10 te voldoen voor bestaande complexe code.  
- De Bettercodehub en/of andere code analyse tools gebruiken om je code te analyseren op software maintainability, zoals het analyseren van guidelines 1 t/m 8 en 10.  
- Nauwelijks violations van richtlijn 10 (R1 t/m R7).  
  
**Hoe je dit moet bewijzen:**

* Laat voor richtlijnen 1 t/m 5 en 10 concrete stukjes code (code-snippets) zien waar je demonstreert hoe je de richtlijnen hebt toegepast en licht je keuzes toe door te laten zien wat er "fout" is en welke refactor-methodes je hebt gebruikt om het op te lossen! Laat daarbij de originele code zien en de gerefactorde code (houdt het wel beknopt, denk aan max. 1 pagina per richtlijn!).
* Geef voor richtlijnen 6 t/m 8 in ieder geval een concreet advies hoe je deze guidelines zou toepassen in een concreet stuk code en beargumenteer waarom dat zou werken. Let op, ook als aan de guideline al is voldaan kun je nog steeds een advies geven om het "te verbeteren" of "dat in de toekomst zo te houden". Je mag natuurlijk ook de "concrete" oplossing geven, maar indien dat te veel werk is, mag je dus ook alleen een beargumenteerd advies geven. Wees in je advies wel "concreet", dus adviezen als: "dit stuk code moet je opsplitsen" is niet voldoende, geef concreet aan hoe je het gaat opsplitsen en beschrijf ook de "gevolgen" van zo'n opsplitsing voor andere delen van je code!
* Je mag bovenstaande aantonen met code vanuit je eigen project, of andere code. Bv, Umlet, het UML-diagram tekenprogramma staat ook op github en kun je ook door BetterCodeHub laten analyseren.
* Hint: gebruik regelmatig BetterCodeHub op je eigen project.

**Info:**Om te slagen moeten minimaal guidelines 1 t/m 5 en 10 worden toegepast in je bewijsstukken. Voor een betere beoordeling moet ook advies over de andere guidelines gegeven worden. Hoe goed ze behandeld worden en of ook correcte adviezen voor de overige guidelines als bewijsstukken worden aangeleverd bepaalt de inschaling in de rubrics: matig, voldoende, goed.

**Design Patterns (workshop Design Patterns 1 t/m 3)**

**Wat je moet kunnen:**- Voor willekeurige contexten de juiste DP kunnen identificeren en toepassen  
- De geschiktheid van een DP kunnen beargumenteren in een context  
- Meerdere DPs kunnen combineren  
- Voor alle aan bod gekomen DPs java code kunnen produceren in context  
- Zelfstandig nieuwe DPs kunnen toepassen  
- Bestaande code kunnen aanpassen en een geschikte DP kunnen toepassen  
  
**Hoe je dit moet bewijzen:**Omdat niet alle criteria in deze rubric even makkelijk in het groeps-project te behalen zijn, kun je deze rubric ook halen door onderstaande opdrachten in je portfolio te stoppen (zoals behandeld in de module). Bij elke opdracht staat of je het als groep of als individu doet. Je mag ook iets anders als bewijsstuk aanleveren, geef dan duidelijk aan waarom je vindt dat dat de rubric in dekt.

1. In groep van max 3 pers. een design pattern bestuderen en een presentatie maken waarin je het nut, de werking en de toepassing van deze design patterns uitlegt. Vermeldt daarbij in ieder geval: naam en synoniemen, categorie, intent, UML diagram (structuur, participants, collaborations), casus met voorbeeld source code. (groeps-opdracht, en wordt in module als opgave gegeven.)
2. Maak voor bovenstaande design pattern een opgave (beschrijving en eventueel een start-project met source code) waarbij dat design pattern moet worden toegepast. Gebruik daarvoor niet een bestaande opgave van internet; maak het ook niet te makkelijk; het maken van de opdracht moet de maker helpen het design pattern te begrijpen. (groeps-opdracht)
3. Maak voor elk van bovenstaande opgaves ook een uitwerking. (individueel, wees in ieder geval in staat aan te tonen dat je de uitwerking begrijpt!)
4. Geef bewijsstukken waarbij je andere design patterns toepast. Dit mag jouw uitwerking zijn van een opgave bedacht door een andere groep. Design patterns waar de uitwerkingen op DLO beschikbaar zijn tellen niet, tenzij ze substantieel zijn uitgebreid. (individueel)

**Info:**  
Om te slagen moeten minimaal punten 1 t/m 3 worden uitgevoerd. Hiermee kun je een beoordeling matig behalen. Voor een hogere beoordeling moet punt 4) voor andere design patterns worden toegepast waarbij in ieder geval een design pattern uit elke categorie (creational, behavioural, structural) is uitgewerkt. Meer design patterns hebben een uiteraard een positief effect op de beoordeling.