“Code is like humor. When you have to explain it, it’s bad.” - Cory House

NIVEAU 1: MINIMAAL VOOR BEGINNERS

1) code/programma voldoet aan alle opgegeven functionele eisen (en eventuele niet-functionele eisen als er gevraagd wordt om een bepaalde programmastructuur enzo)
2) de code staat op een github repo waar de docenten/begeleiders bijkunnen [uiteraard heb je het adres daarvan gecommuniceerd!]
*handig is mogelijk een ITvitae-repo met een package per opdracht, waarbij de packagenaam gelijk is aan de naam van de opdracht.

NIVEAU 2: MINIMAAL VOOR AFSTUDEREN

1) identifiers (namen van variabelen, methoden, klassen en eventuele packages) in het Engels, en met standaard-Java hoofdlettergebruik (ClassName vs methodName, variableName en ENUM_OR_CONSTANT)
2) intelliJ geeft geen errors of warnings bij het compileren
3) buiten loop-indexes en lambda-parameters: variabelennamen een "woordenboekwoord" of combinatie daarvan. Dus geen "bkCnt" maar "bookCount" enzovoorts. En liefst uiteraard ook een naam die naar je beste vermogen aangeeft wat er in de variabele zit.
4) geen uitgecommentarieerde of dode (ongebruikte) code
5) regels moeten niet van het scherm aflopen (onpraktisch bij checken en wijzigen van code)
6) zorg voor naar Java/IDEA-standaarden geformatteerde code. Dus of voor inchecken Ctrl+Alt+L voor elke Java-file, of File->Settings->Tools->Actions on Save en dan reformat code en run code cleanup

NIVEAU 3: PROFESSIONEEL

1) methodes niet langer dan 25 regels, kijk ook kritisch of een klasse niet erg veel methodes krijgt, of een package erg veel klassen
2) Cyclomatische complexiteit van een methode niet hoger dan 10, en liefst 5 of lager
3) DRY toepassen: codefragmenten die duplicaten zijn (in de zin dat als één verandert, de ander precies hetzelfde moet veranderen) uitfactoren
4) messages van IntelliJ ook opgelost (vraag het als je ze niet begrijpt/dat niet lukt)
5) geen loze witregels (bijvoorbeeld tussen functieheader en body)
6) overbodige code/features vermijden, in elk geval in de hoofdbranch (je mag altijd experimenteren in een zijbranch). Want 'code is not an asset, it's a liability'
7) voor zover mogelijk rekening houden met rare input en security (SQL-injectie en andere OWASP-dingen)
8) vermijd magische constanten (getallen anders dan -2/-1/0/1/2) en 'magische' herhaalde strings
9) tenzij daar sterke redenen tegen zijn, je zoveel mogelijk aan conventionele programma-structuur houden (bv controller-service-repository) en standaardfuncties/bibliotheken gebruiken ipv “homebrew”
10) omdat je hier veel moet leren eisen we niet dat alles 80% of meer unittestcoverage heeft; maar af en toe een unit test in je code zetten is wel een goede oefening!

Algemene tips bij het leren programmeren

Als je op professioneel niveau wilt leren programmeren, heb je een 'gebalanceerd dieet' nodig:

1) Daadwerkelijk programmeren is noodzakelijk om te leren programmeren
    1) Maar als je niet opschiet (15 regels of minder in een half uur) zoek naar/vraag om iets makkelijkers om mee te trainen.
    2) vraag desnoods eerst om voorbeeldcode; probeer die eerst te begrijpen, dan te voorspellen
    3) maak eerst pseudocode (beschrijving in normale tekst) als probleem (te) moeilijk is. En/of splits het op in stappen/kleinere problemen 'verdeel en heers'

2) Experimenteren met nieuwe technieken (als ik dit doe, wat gebeurt er dan) kan erg helpen

3) Informatie opdoen van boeken, lessen (offline of online: Youtube, Udemy, Pluralsight, filmpjes van Devoxx of GOTO conferences of NLJUG) kan helpen heel erg goed te worden als je basisprogrammeren beheerst

4) Aantekeningen maken en af en toe doorlezen/jezelf overhoren kan nuttig zijn. Als je iets moeilijk vindt, kan het ook helpen voor jezelf (of eventueel zogenaamd voor iemand anders) een 'handleiding tot feature X' te schrijven.

5) Je code laten reviewen door anderen en zelf andermans code reviewen zijn ook erg nuttig!

6) Praten met anderen over programmeren kan enorm helpen, ook qua motivatie (kan in groep, of op een van de vele programmeursmeetups, zie bv meetup.com)

Mag ik Googelen/chat-gpten als ik iets niet weet?

-ja, maar je bent hier om te leren, niet om oefenprogramma's te produceren (hoe meer je leert, hoe meer/betere keuze je hebt met banen en hoe sneller je geplaatst kan worden)
-dus als je iets Googelt/chatGPT, zorg dat je elke regel begrijpt die je in je code zet.
-en probeer dat stuk code de volgende dag en na een week ZONDER Google/chatGPT