Igor Baeyens & Jari Miers
Reflectieverslag development 4

De verschillen tussen OO t.o.v. FP:

Object-Oriented Programming (OO)	Functional Programming (FP)
Parallel programming niet mogelijk daardoor deden we aan pair programming.	Makkelijk samen te werken met git flow.
We moeten steeds een nieuw object aanmaken om ons project uit te breiden.	We kunnen steeds verder bouwen op ons project zonder nieuwe objecten aan te maken.
We konden zeer “snel” en “efficiënt” te werk gaan om alle gebruikte design patterns met elkaar te laten werken.	Moesten we dit doen in FP zou dit veel langer duren.
Moeilijker om direct de error te begrijpen.	Makkelijker om error te vinden en op te lossen.

De voor- en nadelen van OO versus FP:

Object-Oriented Programming (OO)	Functional Programming (FP)
Zeer makkelijk om classes uit te breiden (extends, implements)	Je mag geen eerder gebruikte functie aanpassen.
Het gebeurt vaak dat men het overzicht verliest tussen alle files of klasses.	Mogelijk om alles in één file te schrijven.
Je kan eenmaal iets schrijven en deze meerdere keren gebruiken. (een shopper kunnen we meerdere keren gebruiken)	Bij het schrijven van pure functies wordt het soms moeilijker om de code te lezen en te begrijpen.
Zeer fijn om methodes goed te beveiligen (protected , private,…)	Na FP veel te gebruiken/oefenen zal het zeer makkelijk zijn om snel en efficiënt een oplossing te vinden op jouw probleem.
Objecten worden soms heel groot waardoor het wel wat werk kost.	

Voordelen = Groen
Nadelen = Rood

Pedamkar, P. (2021, 24 maart). *Functional Programming vs OOP*. EDUCBA. https://www.educba.com/functional-programming-vs-oop/