# Kritische reflectie werkstuk

## Verschillen

|  |  |
| --- | --- |
| Object Oriented Programming | Functional Programming |
| Maakt gebruik van mutable data | Maakt gebruik van immutable data |
| Loop over iteratieve data | Gebruikt herhaling over iteratieve data |
| Wordt gebruikt bij programmeren van veel verschillende items bij weinig operaties | Wordt gebruikt bij programmeren van weinig verschillende items maar bij vele operaties |
| Hierbij worden classes gebruikt waarbij instanties van objecten worden gecreëerd | Hierbij voert elke functie een bepaalde opdracht uit |
| Ideal voor grote projecten en modellen | Gericht op data manipulatie |

## Voordelen

|  |  |
| --- | --- |
| Object Oriented Programming | Functional Programming |
| Makkelijker denkwijze in objecten omdat dit lijkt op de realiteit | Wordt gebruikt als er kleine opdrachten zijn met grote stappen |
| Je kan zelf bepalen in welke volgorde opdrachten worden uitgevoerd | Ondersteund parallel coderen |
| Helpt om irrelevante data te verbergen en geen acces te geven die niet aanvaard is | Eenvoudiger denkwijze, functies zijn gebaseerd op de argumenten binnen geen extra’s achter de schermen |
| Unit testen is met objecten gemakkelijker om aan te roepen | Werkt goed bij het coderen wanneer er geen of vastgestelde grenzen of doelen zijn |
| Ideaal om iets mutable te maken zoals elke keer een user aanmaken en hem te wijzigen | Kan gebruik maken van geneste functies |
| Makkelijk toevoegen van nieuwe data en functies | Elke functies kan op gelijk welk moment aangeroepen worden |

## Nadelen

|  |  |
| --- | --- |
| Object Oriented Programming | Functional Programming |
| Geen ondersteuning voor parallel coderen | Heeft geen blueprint van bepaalde data zoals User die een klasse is |
| Veel files voor verschillende klasses in java | Heeft geen access specifiers zoals ‘Public, Private & Protected’ |
| Minder efficient en veel complexer om te werken met de functies in classes | Uitvoeren van opdrachten moet in een specifiek order |