# Algemeen

## 1. This

Er ontbreken veel “this.”. Dit zorgt voor onleesbare code en kan soms ook leiden tot het gebruik van de verkeerde variabelen.

De oplossing voor dit probleem is voor alle keren dat een lokaal field gebruikt wordt, een “this.” Er voor wordt gezet.

Dit zorgt uiteindelijk voor leesbaardere code.

## 2. Beschikbare variabelen

Er zijn beschikbare variabelen. Dit betekend dat deze variabelen overal ge-set kunnen worden.

Deze variabelen mogen alleen gelezen kunnen worden, dus is het beter om ze private te maken en getters toe te voegen. Alle plekken waar deze velden worden opgeroepen zullen dus ook vervangen moeten worden door getters.

Dit zorgt er voor dat niet alle variabelen van buitenaf aangepast kunnen worden en zorgt er voor dat waarden niet per ongeluk aangepast kunnen worden.

## 3. Finals

Er ontbreken veel “final” keywords. Hierdoor kan er per ongeluk geschreven worden naar een variabel terwijl het alleen maar uitgelezen wordt.

De oplossing voor dit probleem is om voor alle fields die niet veranderen, een “final” voor te zetten.

Dit zorgt voor leesbardere code en voorkomt dat de variabelen aangepast worden terwijl dit niet de bedoeling is.

## 4. Nutteloze comments

Er zijn veel nutteloze comments. Zoals bij een field uitleggen waarom het bestaat, of comments bij getters en setters. Deze zijn overbodig.

De oplossing voor dit probleem is om alle nutteloze comments te verwijderen. In plaats hier van zal boven elke functie (exclusief getters, setters) javadoc komen.

Dit zorgt er voor dat de code makkelijker te begrijpen is.

# Styles

## 1. Style array

Styles bevat een array van Styles. Hierdoor krijg je dat je een Style hebt met lege style waarden, met daar binnenin een aalstal Styles met style waarden en met lege arrays.

Dit is op te lossen door een aparte klasse te maken met een array van Styles. Deze klasse zal dan de “StyleFactory” klasse heten. Hierin zal een functie staan waarin op basis van een enum (zie punt 2) een Style object zal worden aangemaakt en gereturned.

Dit zorgt er voor dat er geen lege velden meer zullen ontstaan in de Style klasse.

## 2. Primitive obsession

De style wordt gekozen aan de hand van een level. Dit is een interger. Hierdoor kan het lastig zijn te begrijpen welk getal bij welke style hoort.

Een oplossing voor dit probleem is om de interger om te zetten naar een enum.

Dit zorgt er voor dat je duidelijk kan zien welke style level bij welke style hoort.

## 3. Dode code

Er is een functie genaamd toString die niet gebruikt wordt. Dit is dode code en ook speculative generality.

De oplossing voor dit probleem is om de functie te verwijderen.

Dit zorgt voor beter leesbare en begrijpbare code.

# Slide

## 1. Vectors

Op een aantal locaties worden Vectors gebruikt als collectie. Vectors zijn legacy en zijn over het algemeen ook langzamer dan bijvoorbeeld ArrayLists.

De oplossing voor dit probleem is om alle Vectors te vervangen voor ArrayLists. Er zijn geen speciale features van Vectoren die hier gebruikt, die ArrayLists niet hebben. Ze kunnen dus worden vervangen door een ArrayList.

Dit zorgt uiteindelijk voor nettere, leesbaardere en snellere code.

## 2. Hele objecten

Er wordt aan sommige functies een heel “Presentation” object meegestuurd. Hiervan worden vervolgens alleen een paar waarden gebruikt.

De oplossing voor dit probleem is om alleen de benodigde waarden mee te sturen aan de functie.

Dit verminderd de complexiteit van de code en maakt het makkelijker te lezen.