

Reflectie

Thema “Overstap naar Java”



28 september 2019

Imad el fetouh

Fontys Hogeschool ICT

Inhoud

[Het toepassen van tenminste 2 design patterns 2](#_Toc20580963)

[Factory pattern 1 2](#_Toc20580964)

[Factory pattern 2 3](#_Toc20580965)

[Observer pattern 3](#_Toc20580966)

[Persistent opslaan van data 5](#_Toc20580967)

[Toepassen van SOLID principes 6](#_Toc20580968)

[Gebruik van GIT en Jenkins/unit-test uivoeren/genereren van coverage rapportage 6](#_Toc20580969)

[Statische code analyse 7](#_Toc20580970)

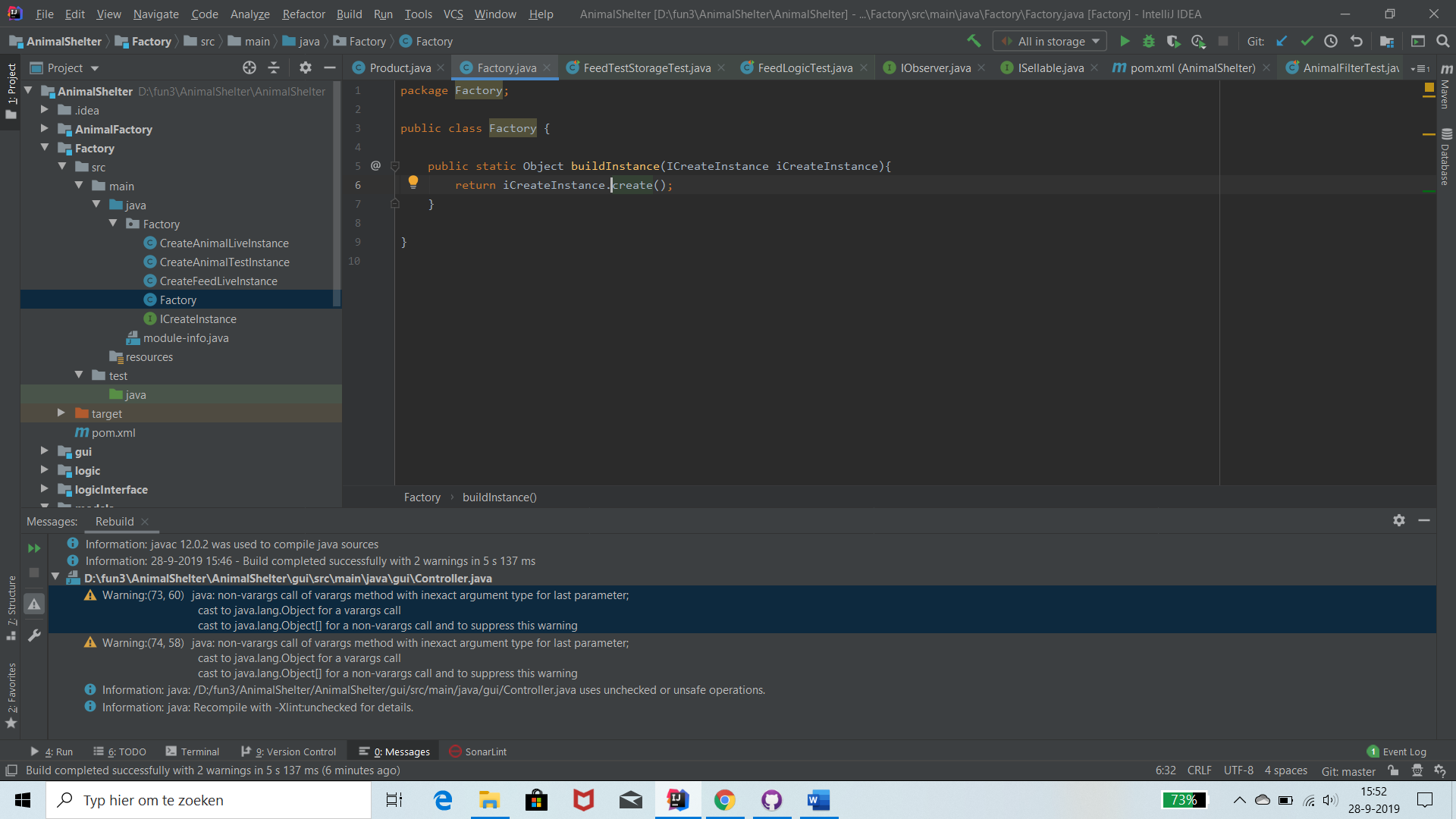
# Het toepassen van tenminste 2 design patterns

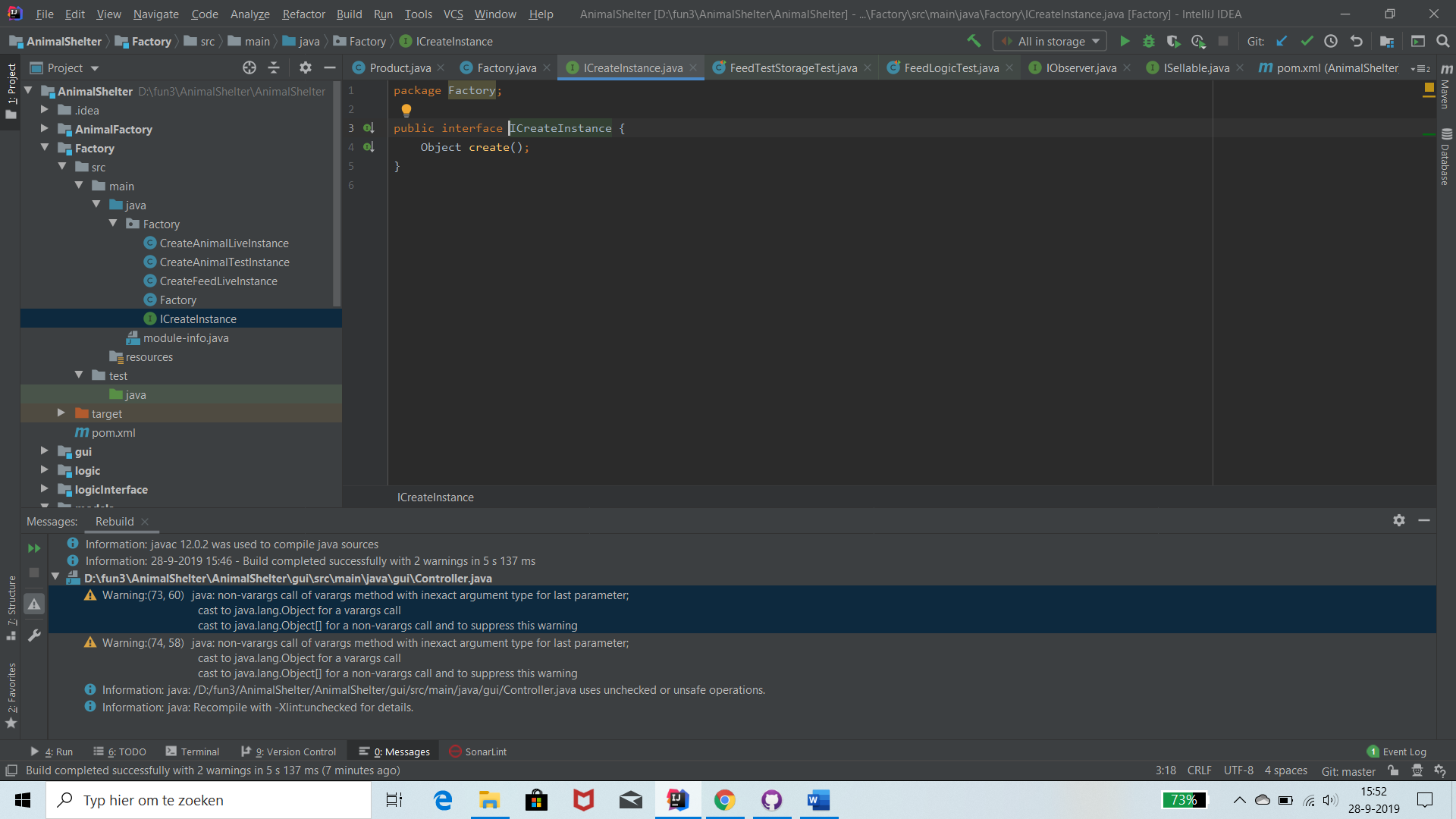
Voor deze leerdoel maak ik gebruik van factory pattern(2 maal) en observer pattern.

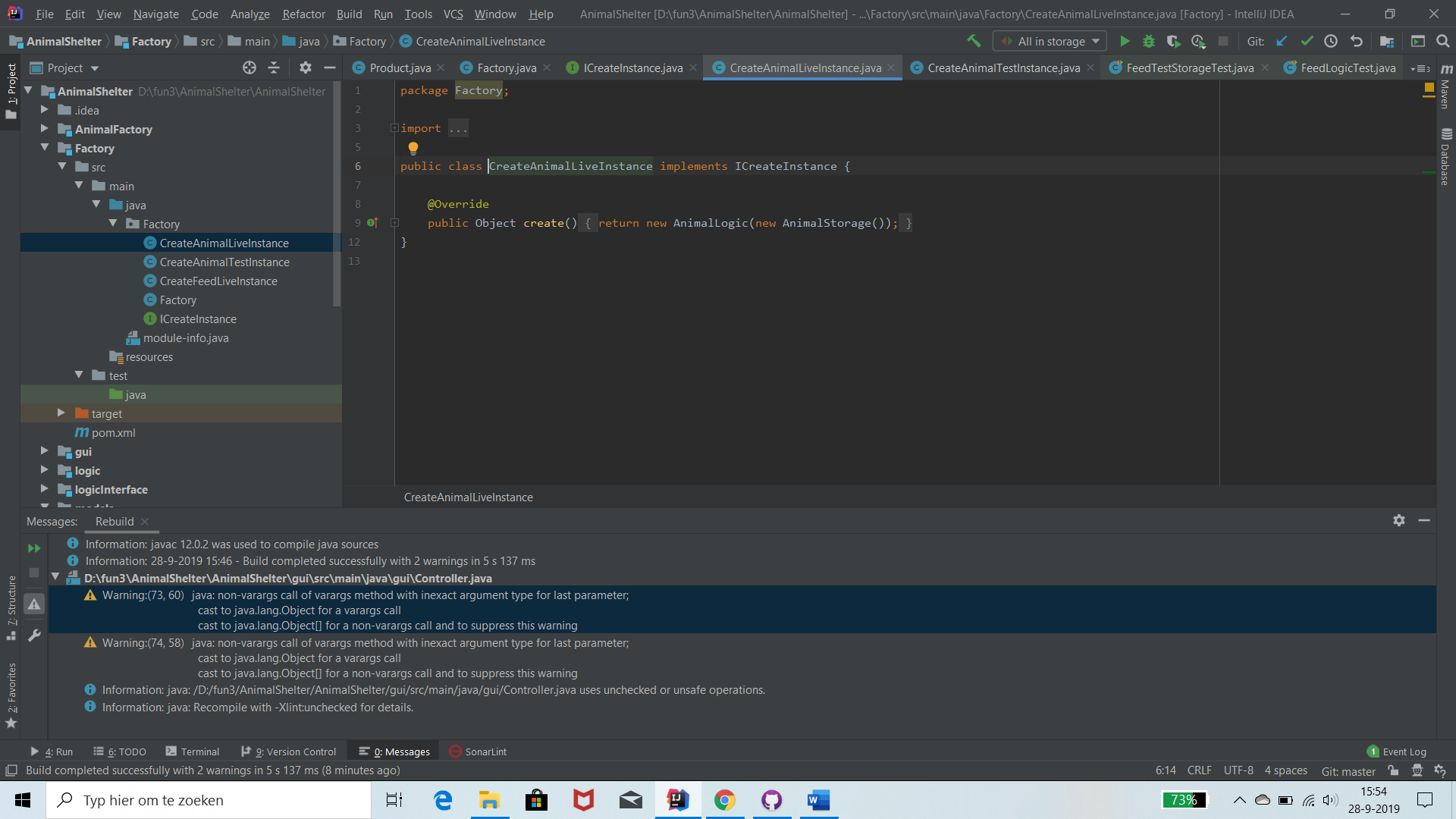
## Factory pattern 1

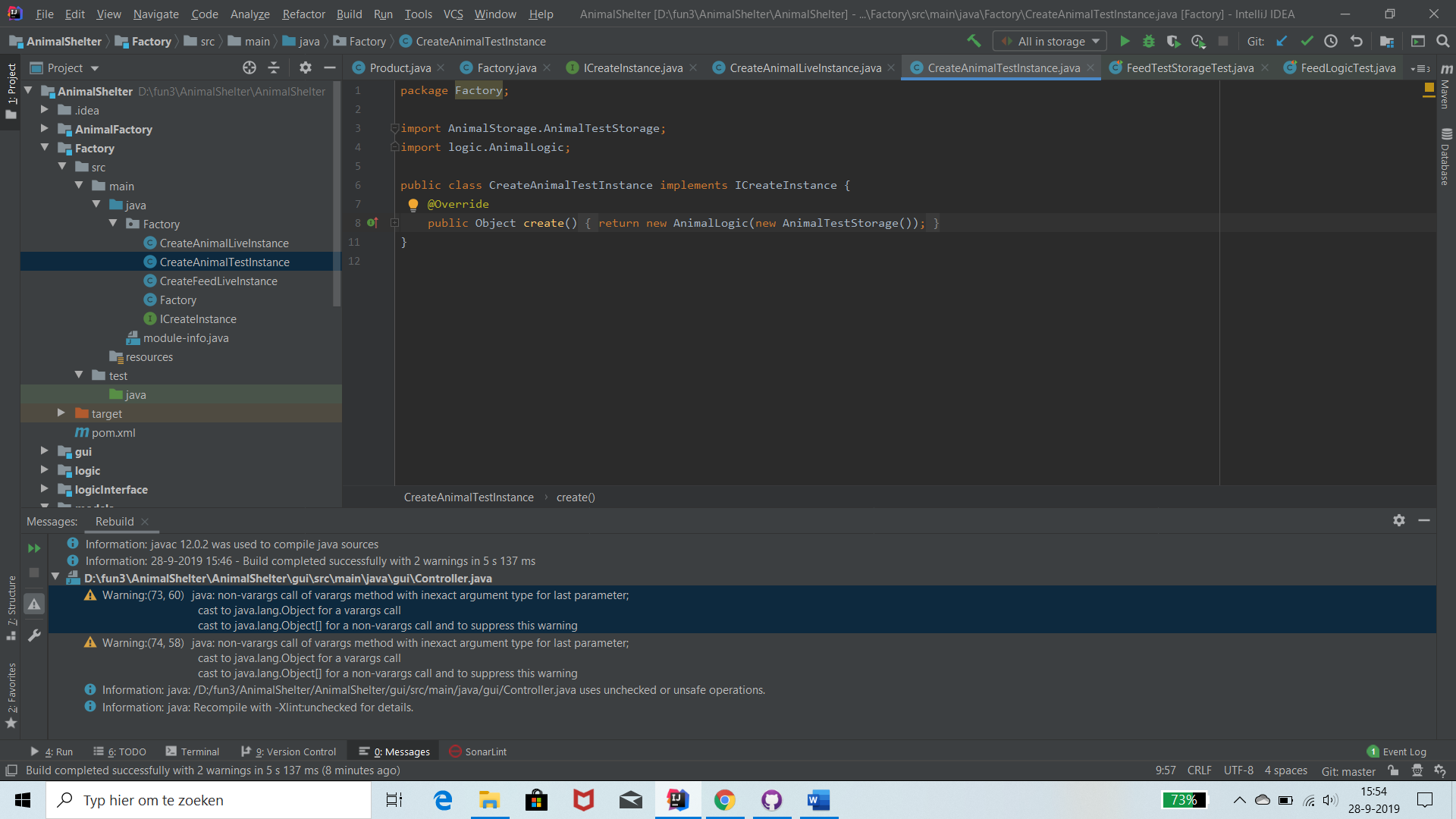
Factory pattern gebruik bij het instantiëren van een context door middel van dependency injection.

Implentatie van factory pattern:

  
*Factory class met een static method. Ik maak deze methode static, omdat de methode niet van het object is maar van de klasse zelf.*

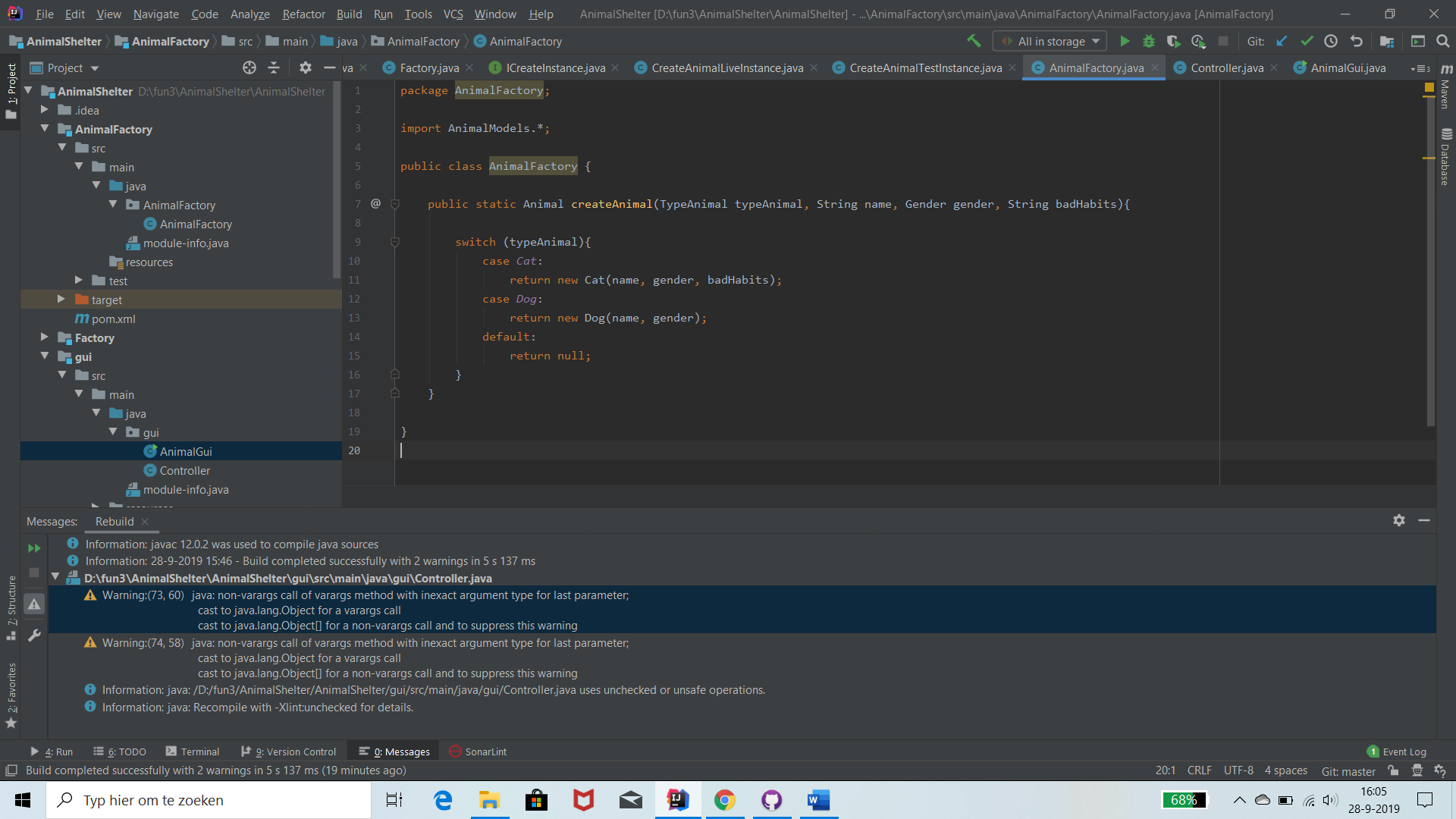
  
*De ICreateInstance interface heeft een “create()” method. Deze methode creëert een instantie van bijvoorbeeld of de live DAL (met database) of met een test DAL (memorycontext).*

  
*Creëren van een animal live instance. Deze gebruik ik bij de applicatie zelf.*

  
*Creëren van een test instantie. Deze gebruik ik bijvoorbeeld voor de JUnit tests.*

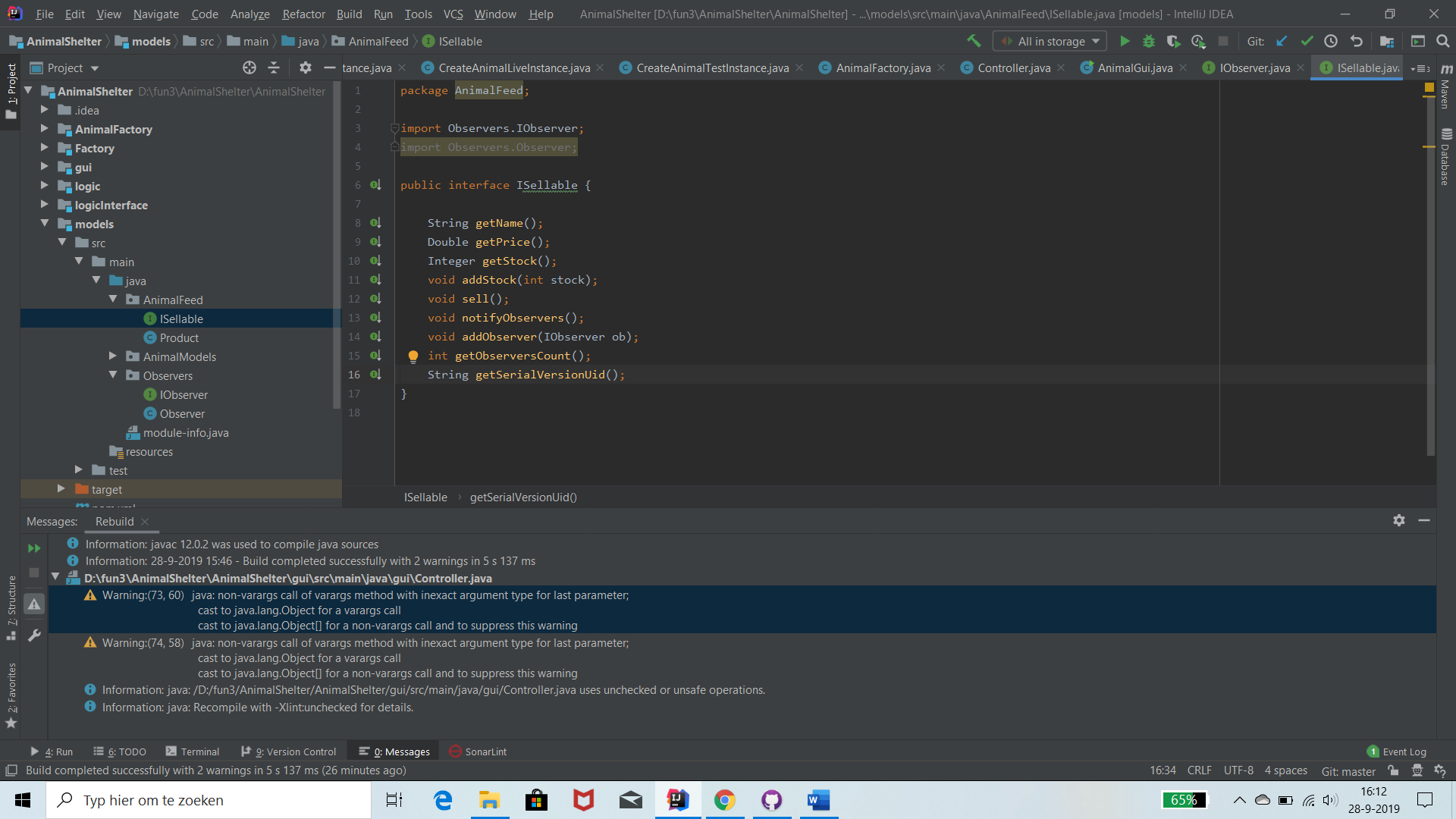
## Factory pattern 2

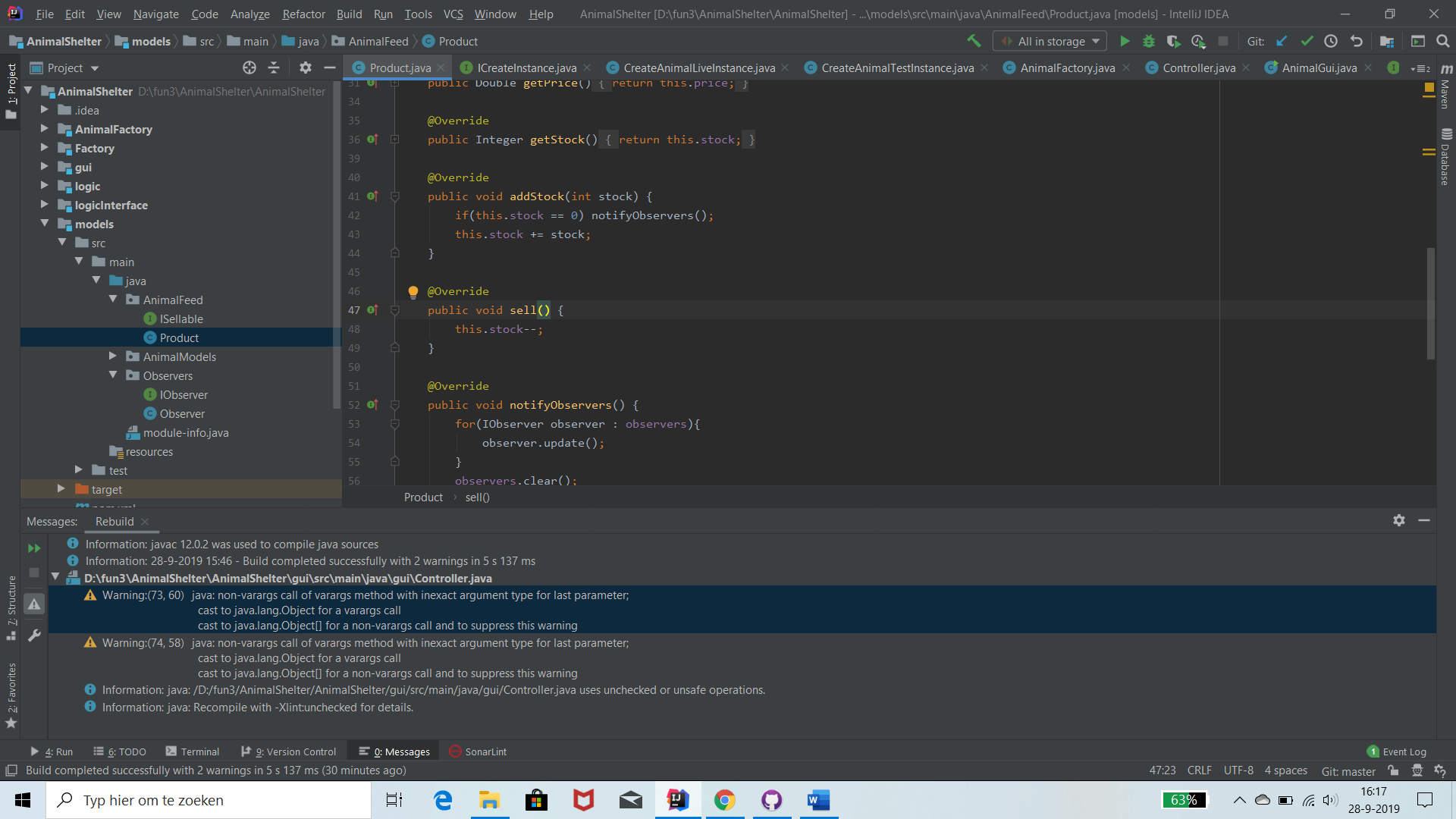
De 2de factory pattern gebruik ik voor het aanmaken van dieren.

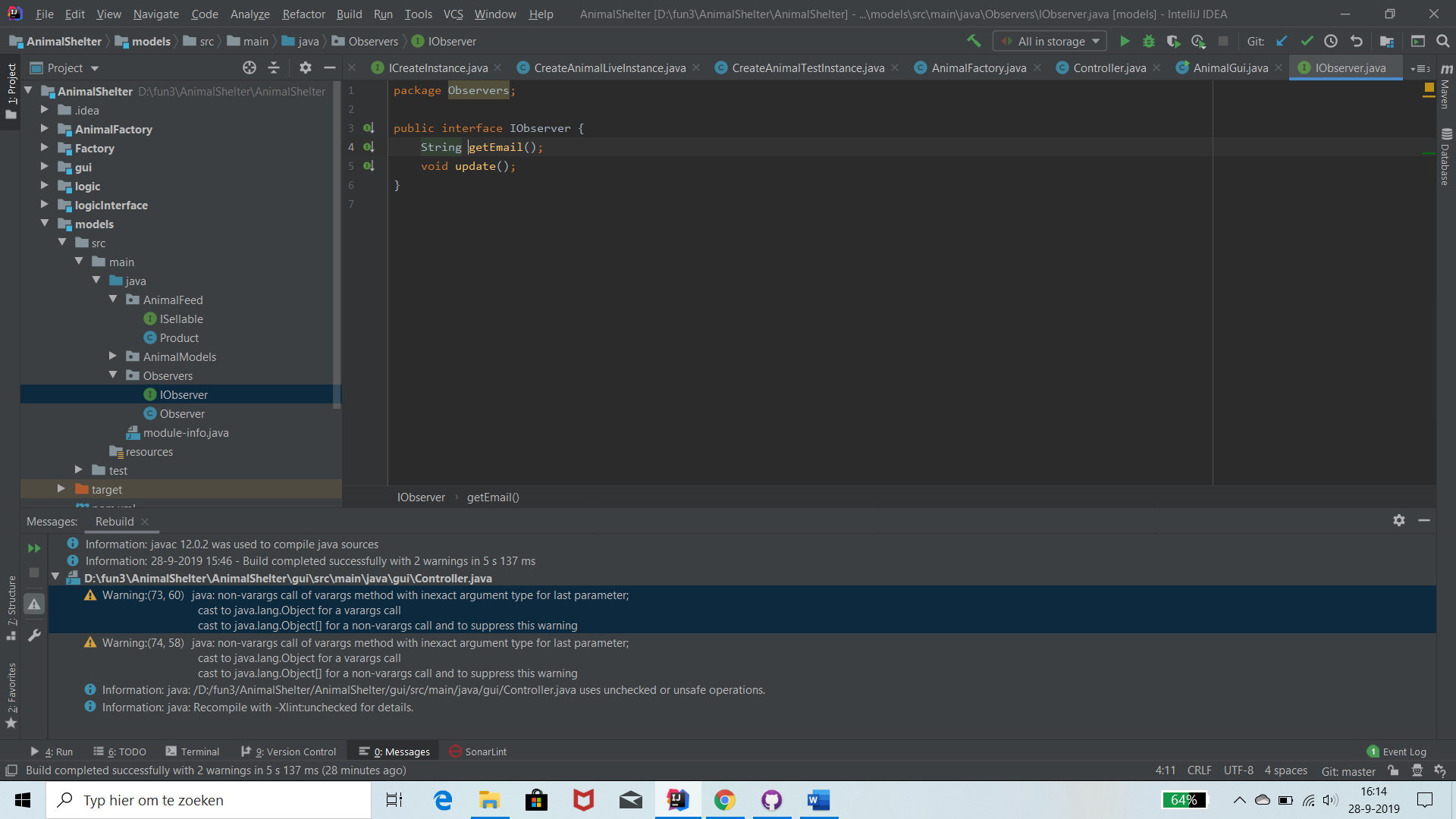


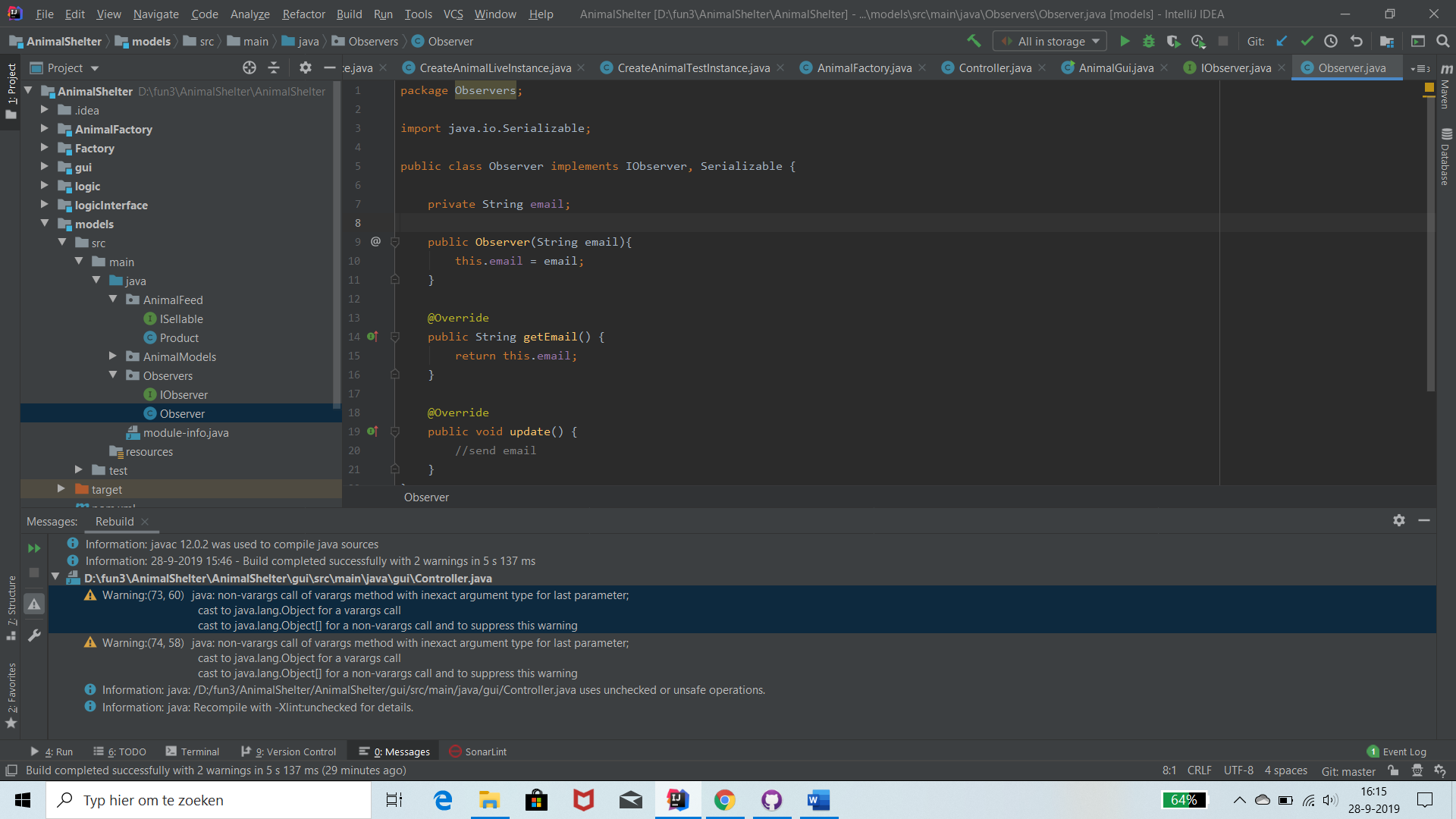
## Observer pattern

Observer pattern gebruik ik bij de webshop bij de verkoop van dierenvoer. Zodra de voorraad van een product bijgevuld is stuur ik een mail naar alle observers met daarin dat de voorraad weer bijgevuld is en verwijder ik alle observers van de lijst.

  
*ISellable interface. Deze interface heeft een lijst van observers.*

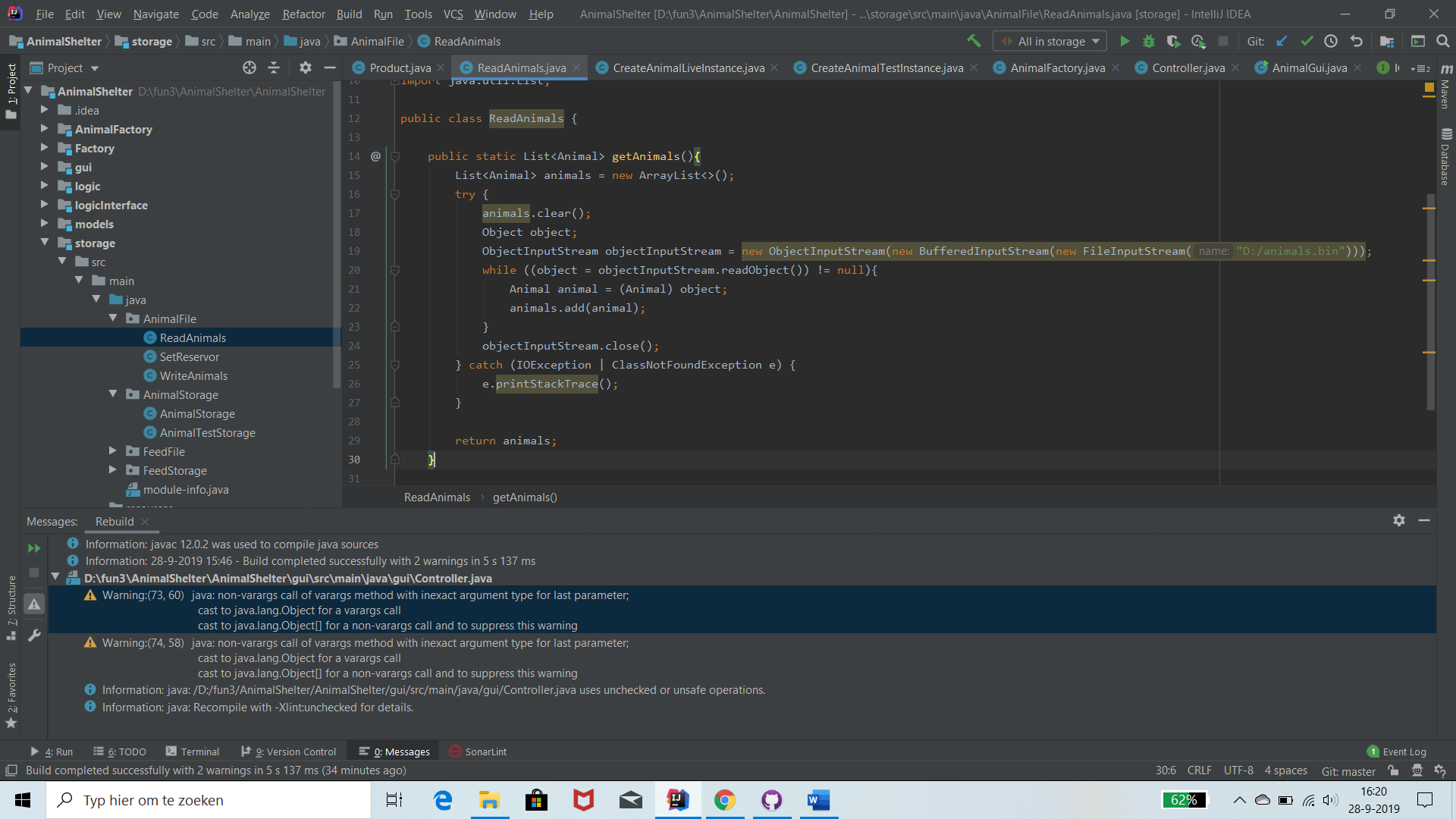
 *Product klasse. Deze klasse implementeert de ISellable interface.*

 *IObserver inteface.*

 *Observer klasse. Deze klasse implementeert IObserver interface.*

## Persistent opslaan van data

Voor deze leerdoel maak ik gebruik van een bestand. Alle objecten die ik naar het bestand schrijf implementeren de Serializable interface. Ik maak gebruik van 2 bestanden. 1 voor de dieren en 1 voor de dierenvoer.

  
*Voorbeeld van het ophalen van dieren.*

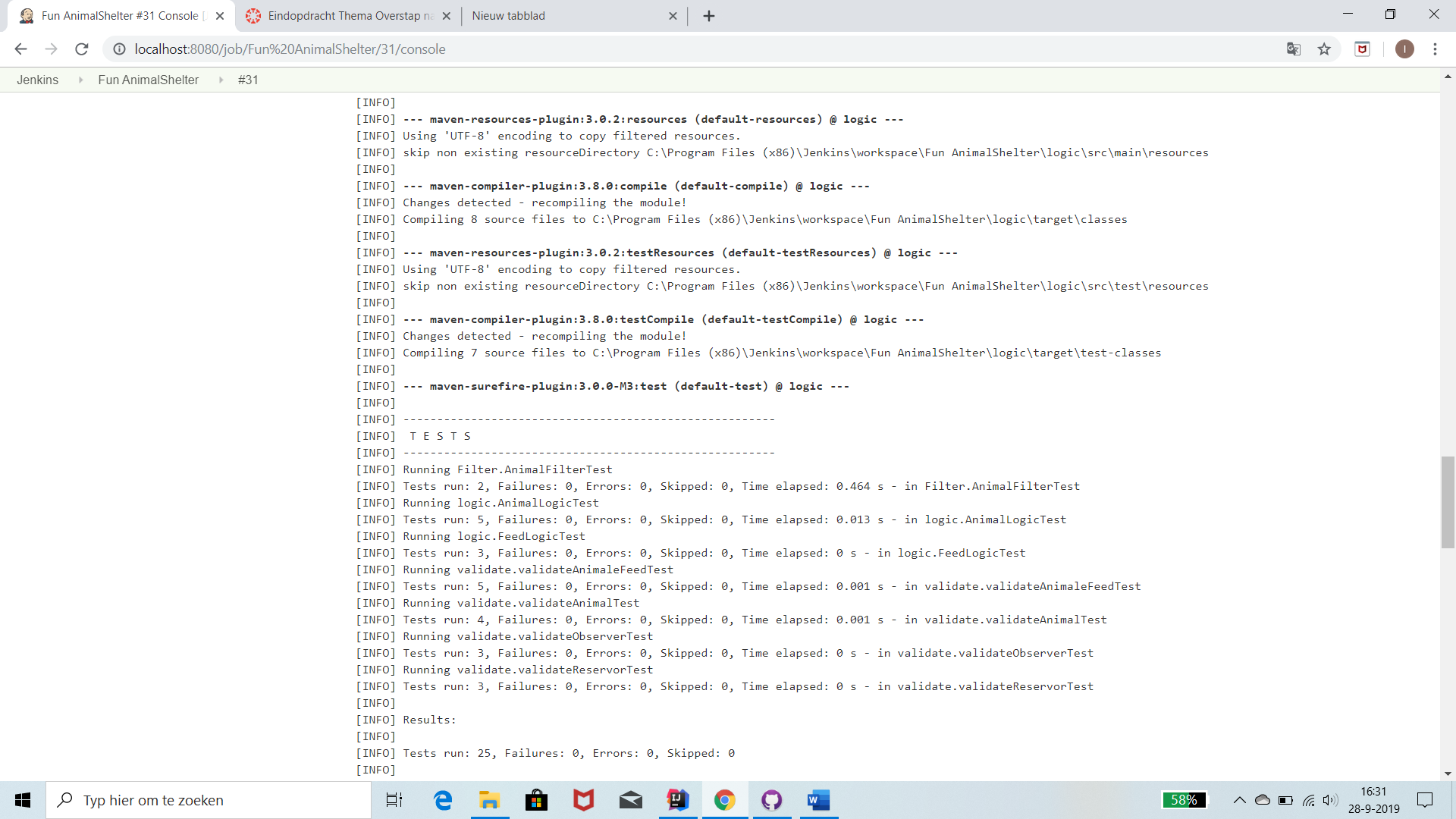
## Toepassen van SOLID principes

Ikzelf denk dat mijn applicatie wel solid is gemaakt. Ik denk dat ik met de design patterns en het gebruik van modules al goed bezig ben met SOLID. Naar mijn gevoel is mijn applicatie goed onderhoudbaar maar uiteraard kan het natuurlijk altijd beter.

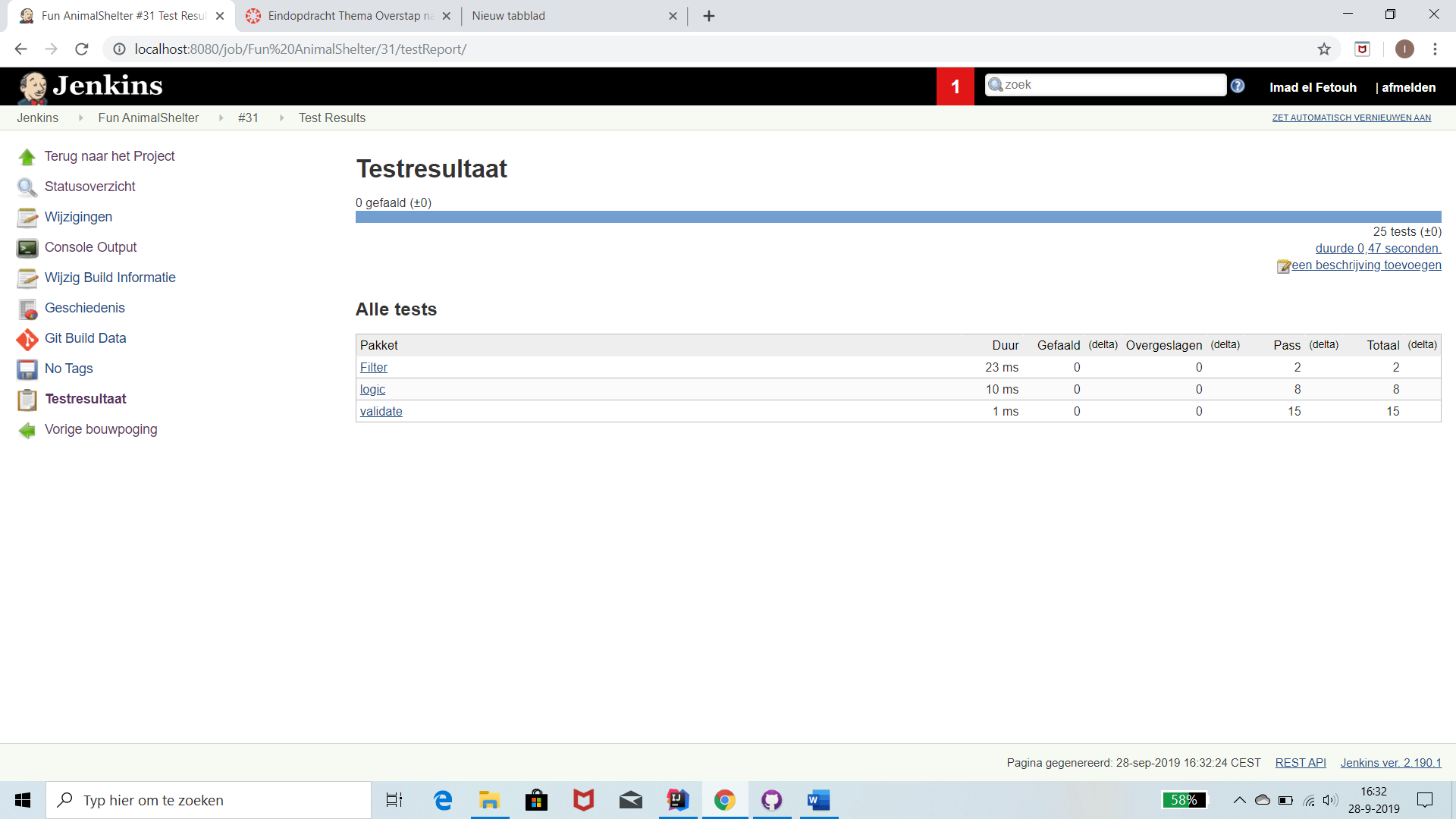
## Gebruik van GIT en Jenkins/unit-test uivoeren/genereren van coverage rapportage

Ik gebruik GitHub en geen GitLab, omdat ik problemen heb met GitLab van school. Ik krijg iedere keer een forbidden error en bij de helpdesk kwamen ze er ook niet uit. Voor CI gebruik ik Jenkins.

Automatisch uitvoeren van uit test doe ik door middel van de maven command: “clean test”.

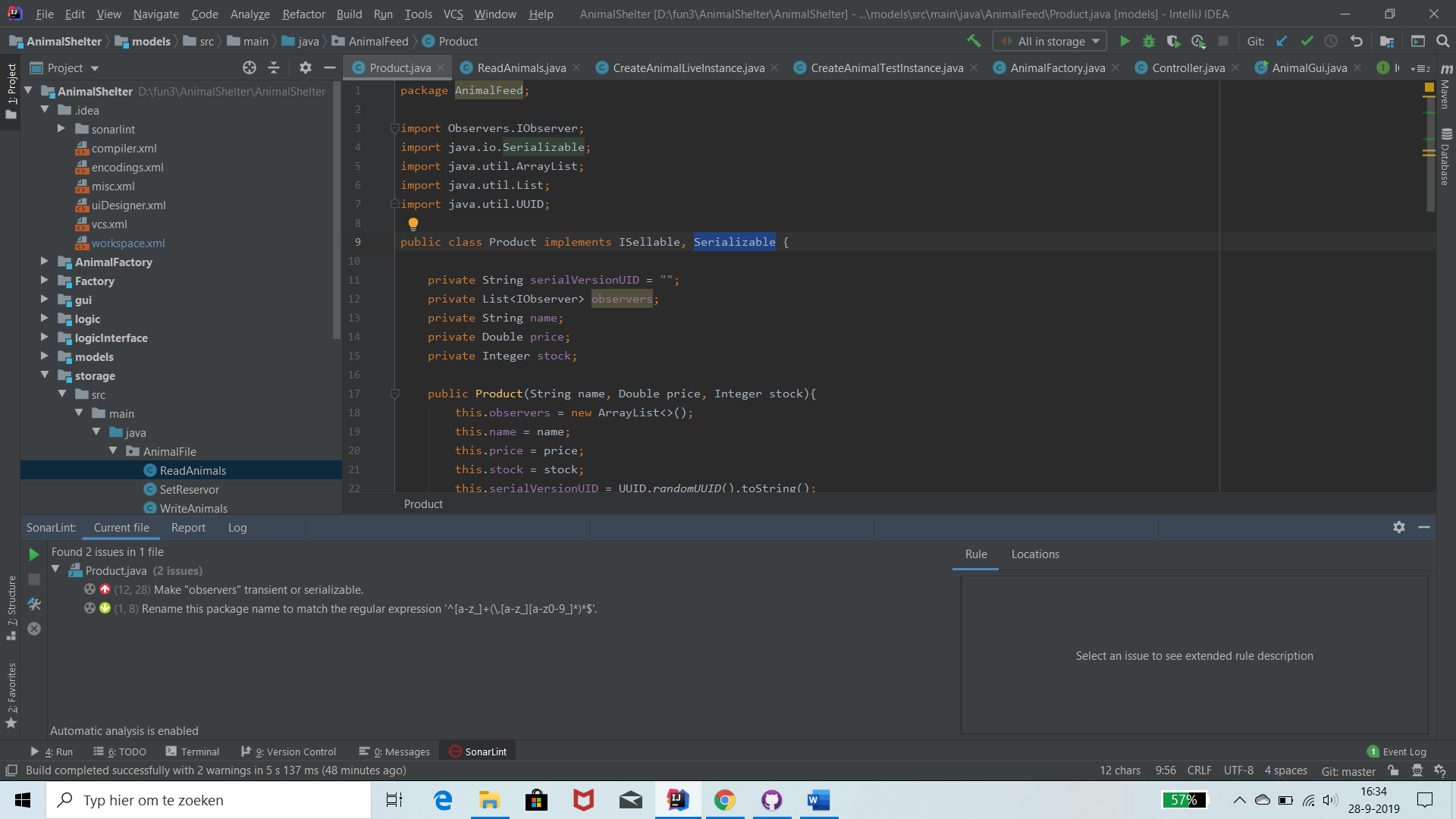


Voor het genereren van coverage rapportage publiceer ik de jUnit test resultaten in Jenkins.



## Statische code analyse

Hierbij maak ik simpel gebruik van SonarLint. Ik heb SonarLint geïnstalleerd in IntelliJ en hij vertelt me steeds wat er beter kan qua code

  
*Voorbeeld van SonarLint output*