Opdracht ...

# Case

Een school wilt de studenten kunnen uitprinten in een volgorde die bepaald kan worden door een medewerker (lees programmeur). Bijv. de ene keer wilt de administratieve medewerker kunnen sorteren op naam, de andere keer op leeftijd, en weer een andere keer op een nog volledig onbekende verzonnen manier.

# Opdracht

Als programmeur wil je dat een wijziging in de requirements (bijv. de manier van het sorteren van studenten) zo min mogelijk impact heeft op bestaande classen. Je wilt dus niet zomaar bestaande classen moeten wijzigen omdat iemand een nieuwe requirement heeft. In deze opdracht ga je de mogelijk nieuwe requirements (lees: de manier van sorteren) loskoppelen van andere klassen (de Student en School class) d.m.v. een interface i.c.m. implementaties van die interface (elke implementatie kan studenten op een andere manier laten sorteren).

Schrijf de volgende classen:

**Student class**

Elke student heeft een studentNummer (String), naam (String) en leeftijd (int). Schrijf een constructor die deze field/instance variables initialiseert, en schrijf getter methods en de toString() method.

(Zie <http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#toString%28%29> waaraan de toString() method moet voldoen, indien je deze method nog niet kent. Laat de toString method de fields teruggeven in een String.)

**School class**De School class bevat een verzameling (lijst) van studenten. Schrijf een *void add(Student student)* method waarmee je studenten kan toevoegen aan de lijst van studenten van de school. Noem deze lijst/variable “studenten”.

Voeg de volgende method toe aan de School class:

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Comparator;**

**public void** printStudenten(final StudentenVergelijker vergelijker) {  
 Comparator<Student> comp = **new** Comparator<Student>() {  
 @Override  
 **public int** compare(Student o1, Student o2) {  
 **return** vergelijker.isGroter(o1, o2) ? -1 : 1;  
 }  
 };  
  
 List<Student> copy = **new** ArrayList<Student>(studenten);  
 Collections.*sort*(copy, comp);  
  
 **for** (Student s : copy) {  
 System.***out***.println(s);  
 }  
}

Deze printStudenten method zorgt ervoor dat iemand een StudentenVergelijker object aan de school kan meegeven, waardoor de method dit object kan gebruiken om de studenten te sorteren (op een willekeurige en van de voren onbekende manier) en vervolgens kan uitprinten.  
Het mooie van deze oplossing is dat de School class niet hoeft worden aangepast voor een nieuwe manier van sorteren van studenten. Een medewerker/programmeur (of beter gezegd een client de School class) hoeft nu alleen maar een instantie/implementatie van StudentenVergelijker te maken, met daarin gespecificeerd hoe er vergeleken moet worden, zodat de method printStudenten de studenten in de gespecificeerde volgorde kan uitprinten. (D.m.v. \*vergelijken\* kan worden bepaald of de ene student groter of kleiner is in volgorde dan een andere student. Hoe dit vergelijken gebeurt, bijv. o.a. van alfabetische naam of leeftijd, is aan de StudentenVergelijker.)

**StudentenVergelijker interface**

Schrijf een StudentenVergelijker interface. Dit interface heeft een method die als return type een boolean heeft. Deze boolean geeft aan of de ene student \*groter\* is dan de andere student, wat vervolgens door de printenStudenten method wordt gebruikt om de studenten in de gevraagde volgorde te zetten. Probeer zelf a.d.h.v. de printStudenten() method te achterhalen welke parameters deze method moet hebben en hoe deze method heet.

**App class**

De App class gebruik je om de applicatie te starten en te testen. Schrijf er dus een main method in.

Zorg ervoor dat de main method een school maakt, er wat studenten in plaatst. Schrijf vervolgens in de main method twee implementaties van StudentenVergelijker: één implementatie die sorteert op basis van leeftijd, en een ander die sorteert op basis van naam.

(Strings kan standaard al in Java vergeleken worden of de één groter is dan de ander. Gebruik daarvoor de compareTo method die is ingebouwd in de String class: <http://docs.oracle.com/javase/7/docs/api/java/lang/String.html#compareTo%28java.lang.String%29> )