Command-Line Interfaces og TextIO

Disse opgaver bygger videre på et løsningsforslag til modulet 11 (Arv og polymorfi 2).

I vælger selv om i vil løse opgaverne på jeres egen løsning eller anvende løsningsforslaget som ligger tilgængeligt på Blackboard.

Opgave 1

For at sikre, at produkter altid instantieres som enten et ”FoodProduct”-objekt eller et ”NonFoodProduct”-objekt skal i gøre klassen Product gøres abstrakt.

Hvad sker der, hvis i forsøger at lave et objekt at typen Product ved at kalde constructoren på Product direkte?

Opgave 2

I skal nu lave et interface kaldet ”Expireable”, med en metode der har følgende signatur:

public boolean isExpired();

Interfacet skal i herefter implementere i Product. At ”implementere” et interface vil sige, at i skal anvende ”implements” keywordet i Product, og i skal override metoden isExpired() fra interfacet. Bodyen på isExpired() i Product skal kun indeholde en linje kode:

throw new UnsupportedOperationException(”Product does not support this operation.”);

Opgave 3

Da FoodProduct og NonFoodProduct nedarver fra Product, nedarver de også isExpired() metoden. I skal nu override denne metode i klassen FoodProduct, så den udnytter attributten expireDate til at afgøre, om produktet er for gammel.

Opgave 4

I ProductDatabase skal i implementere en metode, public void removeExpiredFoods(), der itererer igennem listen af Products og kalder isExpired() på hvert af Product-objekterne. Hvis isExpired returnerer true, skal Product-objektet fjernes fra listen af Product-objekter. Da Product i sig selv ikke implementerer isExpired() men kaster en exception, skal i sørge for at anvende try/catch udenom kaldet til isExpired, for at sikre jer at programmet ikke crasher, når i har at gøre med NonFoodProduct-objekter.