|  |  |
| --- | --- |
| Opleiding: Java jan 19  Resultaat ....../ **30** | Naam test: Test Interfaces + Eenvoudige klassen  Datum: 25/3/2019  Lesmodules: H1 – H16  Naam cursist: |

Test Exceptions

Theorie /6

* Geef 2 soorten RunTimeException?

* Wat is het vershil tussen een checked en een unchecked Exception?

* Waar wordt een lokale InnerKlasse gedefineerd en gebruikt?

Praktijk /24

**TAKE​ ​ME​ ​TO​ ​THE​ ​ICECREAM​ ​SHOP**

We​ ​gaan​ ​in​ ​deze​ ​test​ ​een​ ​applicatie​ ​maken​ ​waarbij​ ​we​ ​de​ ​verkoop​ ​van​ ​en​ ​het​ ​eten​ ​van​ ​ijsjes​ ​gaan simuleren.​ ​De​ ​ijsjes​ ​zijn:​ ​magnumijsjes​ ​(verschillende​ ​types),​ ​hoorntjes​ ​(met​ ​verschillende​ ​smaken voor​ ​de​ ​ballen)​ ​en​ ​raketijsjes.

**Maak​ ​hiervoor​ ​een​ ​nieuw​ ​project​ ​aan​ ​met​ ​de​ ​packages:​ ​be.intecbrussel.eatables, be.intecbrussel.sellers​ ​en​ ​be.intecbrussel.application.**

**1) Eatables /5**

Bovenstaand​ ​schema​ ​stelt​ ​de​ ​eatables​ ​package​ ​voor.​ ​We​ ​beschrijven​ ​hier​ ​de​ ​verschillende​ ​ijsjes​ ​die we​ ​later​ ​zullen​ ​verkopen.​ ​Belangrijke​ ​implementatiedetails:

- De​ ​eat​ ​methode​ ​print​ ​af​ ​wat​ ​je​ ​aan​ ​het​ ​eten​ ​bent.​ ​Bij​ ​een​ ​Magnum​ ​is​ ​dit​ ​dus​ ​afhankelijk​ ​van het​ ​magnumtype,​ ​bij​ ​een​ ​Cone​ ​zijn​ ​het​ ​alle​ ​verschillende​ ​soorten​ ​bollen,​ ​bij​ ​de​ ​IceRocket​ ​gaat het​ ​gewoon​ ​over​ ​het​ ​eten​ ​van​ ​een​ ​raketijsje.

**2) Prijslijst /2**



Dit​ ​schema​ ​hoort​ ​bij​ ​de​ ​sellers​ ​package​ ​en​ ​zal​ ​de​ ​prijzen​ ​van​ ​de​ ​ijsjes​ ​bijhouden.​ ​De​ ​getmagnumprice zal​ ​aan​ ​de​ ​hand​ ​van​ ​een​ ​gegeven​ ​MagnumType​ ​en​ ​een​ ​algemeen​ ​opgegeven​ ​magnumStandardPrice teruggeven​ ​wat​ ​de​ ​prijs​ ​van​ ​zo’n​ ​magnum​ ​zal​ ​zijn.​ ​Bv:​ ​Een​ ​alpinenuts​ ​magnum​ ​zal​ ​1.5​ ​keer​ ​de​ ​prijs zijn​ ​van​ ​een​ ​standaard​ ​magnum.

**3) IceCreamSeller​ ​en​ ​het​ ​IceCreamSalon /6**



Dit​ ​schema​ ​past​ ​ook​ ​binnen​ ​de​ ​sellers​ ​package. We​ ​starten​ ​met​ ​het​ ​maken​ ​van​ ​de​ ​interfaces​ ​en​ ​het​ ​IceCreamSalon.​ ​De​ ​IceCreamCar​ ​zullen​ ​we​ ​later behandelen. Een​ ​IceCreamSalon​ ​stelt​ ​een​ ​ijsjeszaak​ ​voor​ ​die​ ​een​ ​ongelimiteerd​ ​aantal​ ​ijsjes​ ​heeft​ ​die​ ​ze​ ​zelf produceren.​ ​Je​ ​kan​ ​er​ ​ijsjes​ ​bestellen​ ​en​ ​je​ ​kan​ ​er​ ​de​ ​winst​ ​van​ ​opvragen.​ ​Ze​ ​implementeert​ ​immers de​ ​IceCreamSeller​ ​interface.

Uitleg​ ​bij​ ​de​ ​implementatie​ ​van​ ​het​ ​IceCreamSalon:

- orderCone​ ​zal​ ​op​ ​basis​ ​van​ ​een​ ​array​ ​van​ ​flavors​ ​een​ ​nieuwe​ ​cone​ ​aanmaken​ ​en​ ​teruggeven. Ook​ ​zal​ ​ze​ ​aan​ ​de​ ​hand​ ​van​ ​de​ ​pricelist​ ​de​ ​totalprofit​ ​waarde​ ​omhoog​ ​laten​ ​gaan.

- orderIceRocket​ ​zal​ ​een​ ​nieuwe​ ​icerocket​ ​aanmaken​ ​en​ ​teruggeven.​ ​Ook​ ​zal​ ​ze​ ​aan​ ​de​ ​hand van​ ​de​ ​pricelist​ ​de​ ​totalprofit​ ​waarde​ ​omhoog​ ​laten​ ​gaan.

- orderMagnum​ ​zal​ ​op​ ​basis​ ​van​ ​een​ ​MagnumType​ ​een​ ​nieuwe​ ​Magnum​ ​aanmaken​ ​en teruggeven.​ ​Ook​ ​zal​ ​ze​ ​aan​ ​de​ ​hand​ ​van​ ​de​ ​pricelist​ ​de​ ​totalprofit​ ​waarde​ ​omhoog​ ​laten gaan.

Vergeet​ ​ook​ ​niet​ ​een​ ​constructor​ ​aan​ ​te​ ​maken​ ​waar​ ​je​ ​een​ ​PriceList​ ​aan​ ​kan​ ​meegeven.

**4) Eerste​ ​applicatie**

Maak​ ​in​ ​je​ ​application​ ​package​ ​een​ ​klasse​ ​aan​ ​met​ ​een​ ​main​ ​methode.​ ​(bv:​ ​IceCreamApp) Maak​ ​een​ ​PriceList​ ​instantie​ ​aan. Maak​ ​een​ ​IceCreamSalon​ ​instantie​ ​aan​ ​met​ ​behulp​ ​van​ ​de​ ​pricelist​ ​en​ ​steek​ ​deze​ ​in​ ​een IceCreamSeller​ ​variabele. Bestel​ ​enkele​ ​ijsjes​ ​(order​ ​methoden),​ ​steek​ ​deze​ ​in​ ​een​ ​array​ ​van​ ​Eatable. Roep​ ​van​ ​deze​ ​ijsjes​ ​de​ ​eat​ ​methode​ ​aan. Aan​ ​het​ ​einde​ ​van​ ​je​ ​applicatie​ ​print​ ​je​ ​de​ ​profit​ ​af​ ​van​ ​de​ ​icecreamseller.

**5) Keeping​ ​stock​ ​&​ ​NoMoreIceCreamException /2**

De​ ​Stock​ ​klasse​ ​helpt​ ​ons​ ​in​ ​het​ ​bijhouden​ ​van​ ​een​ ​voorraad​ ​ijsjes.​ ​Deze​ ​stock​ ​zullen​ ​we​ ​nodig hebben​ ​wanneer​ ​we​ ​de​ ​iceCreamCar zullen aanmaken straks

Tevens​ ​maken​ ​we​ ​een​ ​eigen​ ​exception​ ​klasse:​ ​NoMoreIceCreamException.​ ​

**6) IceCreamCar /4**

Zie​ ​het​ ​schema​ ​bij​ ​3. We​ ​gaan​ ​een​ ​2​e​​ ​soort​ ​icecreamseller​ ​definiëren,​ ​de​ ​icecreamcar.​ ​Deze​ ​zal​ ​een​ ​eindige​ ​voorraad​ ​ijsjes hebben.​

​Ze​ ​gaat​ ​dus​ ​gebruik​ ​maken​ ​van​ ​een​ ​stock. Zorg​ ​dus​ ​ook​ ​dat​ ​je​ ​aan​ ​een​ ​constructor​ ​zowel​ ​een​ ​pricelist​ ​als​ ​een​ ​stock​ ​meegeven​ ​kan.

De​ ​orderMethoden​ ​zijn​ ​grotendeels​ ​hetzelfde​ ​als​ ​bij​ ​een​ ​IceCreamSalon,​ ​alleen​ ​dat​ ​je​ ​nu​ ​ook rekening​ ​moet​ ​houden​ ​met​ ​de​ ​stock.​ ​Indien​ ​er​ ​van​ ​een​ ​bepaald​ ​ijsje​ ​geen​ ​voorraad​ ​meer​ ​is,​ ​gooi​ ​je een​ ​NoMoreIceCreamException.​ ​

Deze​ ​vang​ ​je​ ​dan​ ​eventueel​ ​later​ ​op​ ​in​ ​je​ ​programma. De​ ​prepareCone,​ ​prepareRocket​ ​en​ ​prepareMagnum​ ​methoden​ ​zijn​ ​private​ ​en​ ​dienen​ ​eerder​ ​als hulpmethoden​ ​gebruikt​ ​te​ ​worden​ ​binnen​ ​je​ ​ordermethoden​ ​om​ ​het​ ​boeltje​ ​clean​ ​te​ ​houden.

**7) AppV2 /5**

Maak​ ​een​ ​nieuwe​ ​IceCreamApp​ ​aan​ ​in​ ​je​ ​application​ ​package.​ ​(vb:IceCreamAppV2) Deze​ ​werkt​ ​exact​ ​als​ ​je​ ​eerste​ ​applicatie​ ​(kopieren)​ ​op​ ​enkele​ ​verschillen​ ​na: - Je​ ​gebruikt​ ​een​ ​IceCreamCar​ ​instantie​ ​in​ ​plaats​ ​van​ ​een​ ​salon - Je​ ​geeft​ ​een​ ​Stock​ ​en​ ​een​ ​PriceList​ ​mee​ ​aan​ ​je​ ​IceCreamCar​ ​constructor Bestel​ ​nu​ ​meer​ ​raketijsjes​ ​dan​ ​er​ ​in​ ​je​ ​stock​ ​zitten.​ ​Wat​ ​gebeurt​ ​er?​ ​Vang​ ​deze​ ​eventuele uitzonderlijke​ ​gebeurtenis​ ​op​ ​en​ ​run​ ​het​ ​programma​ ​opnieuw.

**( bonusen /2)**

**-** Maak een JavaDoc aan voor dit project /1

**-** Maak aan de hand van de interface Eatable een nieuwe LokaleClasse Waffle aan in de klasse WaffleSeller Gebruik deze om een waffel te verkopen/1

**Veel Succes**