# WeekOpdracht 1

Opgave 1

A \*

B: De initialisatie zorgt er voor dat het object juist wordt ingesteld zodra het gemaakt is.

Bij een declaratie maak je een nieuwe variabele aan(je verteld dat je een variabele gaat gebruiken), dit kan zowel in de constructor als in een methode.

C: \*

D: \*

Opgave 2

\*

Opgave 3

\*

Opgave 4

Object types werken met verwijzingen. Als jij een variabele aan een klasse toekent verwijst deze altijd naar een object in plaats van naar de geheugenruimte waarin een primitieve type wordt opgeslagen.

Opgave 5

\*

# Weekopdracht 2

Opgave 1

\*

Opgave 2

\*

Opgave 3

\*

Opgave 4

\*

Opgave 5

A

Het gebruik van een while-lus is hier handiger omdat je nooit weet hoe vaak de lus doorlopen moet worden. Het einde van en while lus kun je aangeven door een voorwaarde in de while-lus mee te geven.

Bij een for-lus geef je in het begin al aan hoe vaak de lus doorlopen moet worden.

B

\*

Opgave 6

\*

Opgave 7

\*

# Weekopdracht 3

Opgave 1

A

In de klassen Kantine en Kassa worden de volgende methodes dubbel aangeroepen:

* public int hoeveelHeidGeldInKassa
* public int aantalArtikelen

Opgave 2

A

Deze methodes maken deel uit van de implementatie van de klasse KantineAanbod, ze hoeven dus niet belangrijk te zijn voor programmeurs van buitenaf, sterker nog soms wil je juist dat programmeurs deze methodes zien. Daarom is het goed om de methodes private te maken zodat je ze verbergt van de interface die zichtbaar is voor andere programmeurs.

B

Een HashMap kan je gebruiken als je zelf keys invoert, ook accepteerde de HashMap meerdere null waarden voor zowel de key als de value.

De HashSet gebruik je als je geen keys invoert, je kan dus alleen over de HashSet itereren als je een for-each loop gebruikt. De HashSet mag minimaal 1 null object bevatten ook mag het geen dubbele values bevatten.

Opgave 3

A

Er wordt een instantie van de klasse Kantine aangemaakt.

Er wordt een instantie van de klasse Random aangemaakt. Overigens komt de klasse uit de javabibliotheek. Ook halen we een instantie van de klasse Kassa op zodat we de dagtotalen kunnen weergeven. Dan maken we een array van het type int, in deze array staat een lijst met willekeurige getallen die de voorraad per artikel voor moet stellen.

Dan wordt er een instantie van de klasse KantineAanbod geïnitialiseerd zodat de Kantinesimulatie deze klasse kan benaderen alsof een persoon een artikel van de stapel af pakt, waardoor de voorraad in slinkt. Ook bepalen de parameters van de KantineAanbod hoe hoog de voorraad en de prijs per product zijn.

De laatste regel in de constructor maakt via een methode in de klasse Kantine een instantie van de klasse KantineAanbod aan.

B

De methode nextInt(int n) van de klasse Random, heeft als parameter int n, met deze waarde kan je het bereik van het pseudorandom getal bepalen. Echter moet er rekening mee gehouden worden dat het bereik dan van 0 tot n-1 loopt, daarom wordt er in de paratmeter 1 bij opgeteld zodat n gegenereerd kan worden. Dus het bereik is exclusief n. Om ook een minimum te stellen moet je een kleine berekening uitvoeren. De berekening luidt: nextInt(max-min +1)+min, omdat je + min doet zal het getal altijd boven het minimum uit komen.

# Weekopodracht 4

Opdracht 1

C

De reden dat je wel *new Administratie()* aan kan roepen is, java maakt automatisch een constructor aan als je deze niet definieert.

D

De reden dat deze twee methodes static kunnen zijn is, net zoals bij een klassenvariabele is de klassenmethode gelijk voor alle instanties die van die bepaalde klassen wordt gemaakt. Zo hoef je bij het aanroepen van een methode niet de instantienaam er voor te schrijven, hier voldoet de klasse- naam ook.

E

Door het private maken van de constructor is het onmogelijk geworden om een instantie van de klasse Administratie aan te maken. Door het private maken van de constructor zorg je er dus voor dat je geen instantie kan maken en verklein je dus de afhankelijkheid van de klasse.

G

\*\*zie a4-tje

H

Door een veld als final te declaren zorg je er voor dat een waarde niet mag veranderen.

I

De compiler klaagt omdat days\_in\_week een instantievariabele is die als final gedefinieerd is. De reden hiervoor is dat we deze variabele gebruiken in een statische methode. Als we met een statische methode werken, zijn er geen objecten/instantie dus kun je ook geen instantievariabele gebruiken.

J

De reden voor het maken van days\_in\_week is het vervangen van een onduidelijke 7 naar een duidelijk veld boven aan de code. Hierdoor vergroot je de cohesie van de klasse. Echter nu je deze variabele gedeclareerd hebben kan je hem ook veranderen wat we juist niet willen. Daarom is het raadzaam om een static final DAYS\_IN\_WEEK te declareren, daardoor kan je de variabele ook niet meer aan passen.

Opdracht 2

B

De super aanroep moet altijd boven aan staan zodat eerst de velden van de superklasse geinitialiseerd kunnen worden. Als je de aanroep niet bovenaan in de constructor zet zal java zelf een lege superaanroep doen waarbij geen parameters worden doorgegeven.

# Weekopdracht 5`

Opdracht 1

C

Ik concludeer uit de resultaten van deze methode dat de methode equals() controleert of twee objecten identiek zijn(in dit geval dezelfde gegevens in persoon). == kijkt alleen of de twee objecten op dezelfde geheugenplaats staan.

D

Als je strings inhoudelijk met elkaar wilt vergelijken moet je de methode equals() gebruiken. Dit komt doordat equals kijkt naar wat er op de geheugenplaats staat van beide objecten en dit dan vergelijkt. Als je de andere mogelijkheid(dus: ==) gebruikt vergelijkt het de geheugenplaatsen met elkaar, niet wat er op die specifieke plek in het geheugenstaat.

Opdracht 2

B

title Authentication Sequence

Pinpas->Pinautomaat: Insert pass

Pinautomaat->Pinpas: Start session

loop

note right of Pinautomaat: Look for errors

Pinpas->Pinautomaat: Password

Pinautomaat->Bank: Password authentication

Bank->Pinautomaat: Accept Pass

Pinautomaat->Pinpas: Provide bank account acces

opt pay

Pinpas->Bank: Enough money to pay?

note right of Bank: Check bank account

Bank->Pinpas: Enough money: Yes/no

Pinpas->Pinautomaat: Cancel Payment

Pinautomaat->Pinpas: End session

Pinpas->Kassa: Pay the price

Pinautomaat->Bank: Decrease bank account

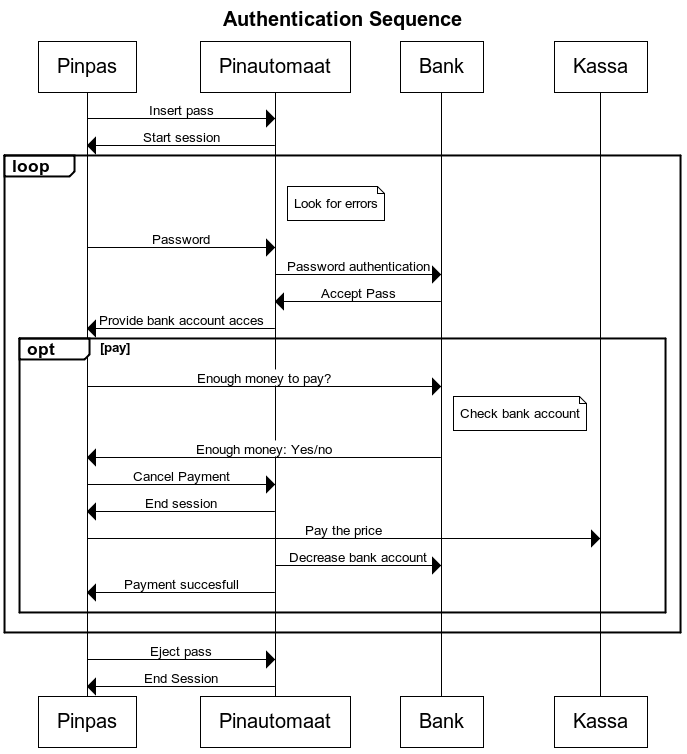
Pinautomaat->Pinpas: Payment succesfull

end

end

Pinpas->Pinautomaat: Eject pass

Pinautomaat->Pinpas: End Session



C

De variabele is protected gemaakt zodat je het alleen maar kan gebruiken binnen de klasse. Zo dat er ook geen andere klasse naar kan verwijzen.