Opdracht uitwerkingen

1.1.B

Vraag: Je hebt hierboven twee instantie variabelen gedeclareerd. Voordat je ze zinnig zou kunnen gebruiken moet je ze wel initialiseren. Leg uit wat de begrippen declaratie en initialisatie betekenen. Let op: in het BlueJ boek word declaratie en initialisatie vaak in één keer gedaan.

Antwoord: Wanneer een variabele word gedeclareerd, word verteld dat hij bestaat, maar nog niks mee gedaan kan worden. Bij het declareren van een variabele, word de naam (bijv. voorNaam) en het type (bijv. String) mee gegeven.

Wanneer een variabele word geïnitialiseerd, word deze bruikbaar. De waarde van de variabele word van NULL (waar hij standaard heen gezet word op het moment dat de variabele word gedeclareerd) naar het gene gezet wat mee gegeven word (bijv. *voorNaam = "Pietje").*

Extra: Zoals in de vraag aangegeven staat, word soms declaratie en initialisatie in één keer gedaan. Dit kan op de volgende manier gedaan worden; "*String voorNaam = "Pietje*".

1.2.C

Vraag: Maak een constructer die de gegevens uit vraag a) als parameters heeft en de instantievariabelen de meegegeven waarden geeft. Let op: de controles uit de vorige vraag moet je ook hier uitvoeren! Hoe doe je dat? Er zijn meerdere mogelijkheden, maar voorkom in ieder geval dubbele code!

Antwoord: Eerder zijn er een aantal setter functie gemaakt in de klasse persoon. De constructer word aangepast van;

public Persoon()

Naar:

public Persoon(int BSN, String Voornaam, String Achternaam, int GeboorteDatumDag, int GeboorteDatumMaand, int GeboorteDatumJaar, char Geslacht)

Vervolgens worden de setter functies aangeroepen in de constructer zodat de controles, die in de setter functies staan, uitgevoerd worden. Dit word op de volgende manier uitgevoerd.

this.setGeboorteDatum(GeboorteDatumDag, GeboorteDatumMaand, GeboorteDatumJaar);

1.4

Vraag: Primitieve typen (zie 3.7 en bijlage B BlueJ boek) en objecten zijn twee fundamen-

teel verschillende soorten typen. Het belangrijkste verschil in gedrag zie hieronder

geïllustreerd aan de hand van twee code fragmenten:

Verklaar dit verschil in gedrag. Gebruik in je uitleg de uitdrukkingen “directe op-

slag” en “referentie (verwijzing)”. In het college is hier ook aandacht aan besteed.

Antwoord:

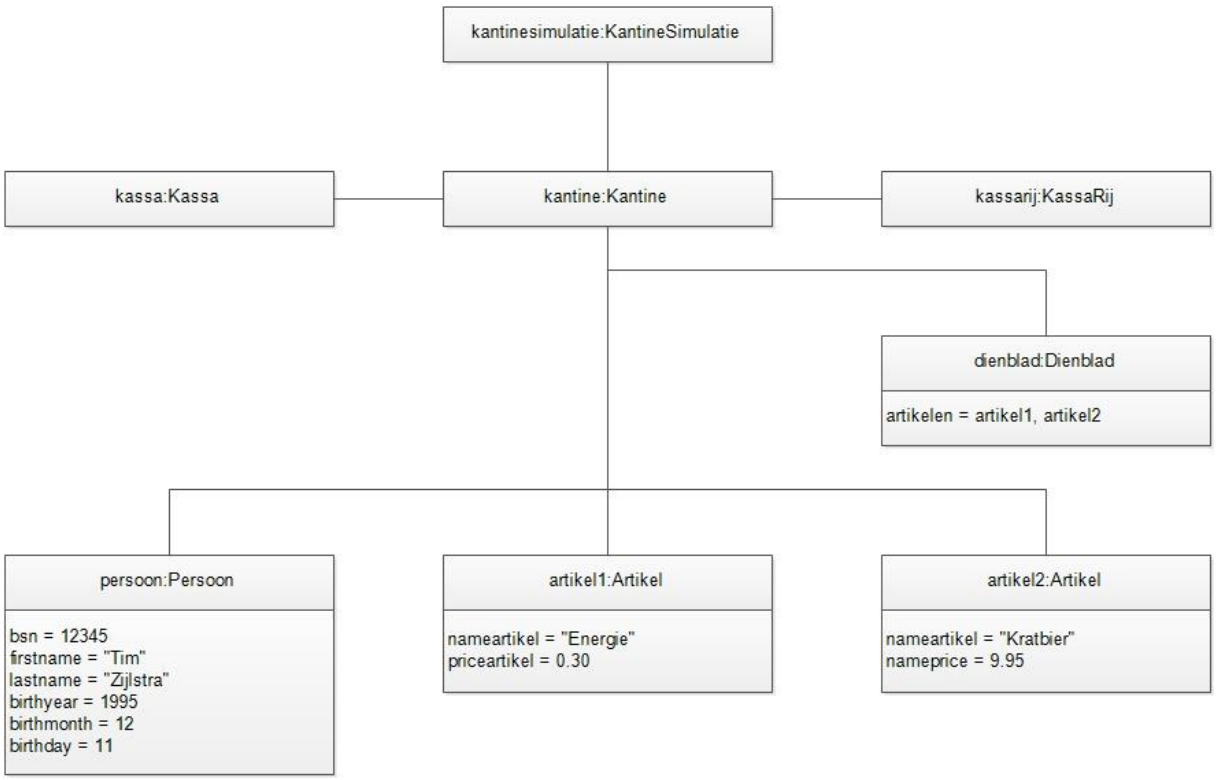
2.5.A

Vraag: Leg uit waarom het gebruik van een while lus in de methode verwerkRijVoorKassa() handiger is dan een for lus.

Antwoord: Stel een nieuwe klant (of Persoon) word in de wachtrij geplaatst op het moment dat de rij verwerkt word. Indien er gebruik gemaakt word van een for loop, word van te voren uitgerekend hoevaak de loop moet loopen. Indien er 5 man in de rij staan, word er van te voren berekend dat de loop 5x moet loopen. Indien er bij de 2^{e} loop een persoon bij komt, blijft het aantal loops op 5 staan en word de laatste loop dus niet uitgevoerd.

Bij een while loop, word na elke loop gecontroleerd of de statement een false of een true is. Indien de statement een true is, word er nog een loop uitgevoerd. In de statement staat in die geval een controle functie die checkt of er minimaal nog één persoon in de wachtrij staat. Indien dat het geval is, word er een true terug gegeven waardoor de while loop nogmaals uitgevoerd word.

2.6 objectdiagram



3.1.A

Vraag: Bij welke methodes in Kassa en Kantine komt dit voor?

Antwoord: Bij de methodes “hoeveelheidGeldInKassa” en “aantalArtikelen”.

3.1.A

Vraag: Leg uit waarom het goed is om de methodes ArrayList<Artikel> getArrayList<String productnaam) en Artikel getArtikel(ArrayList<Artikel>) private te maken.

Antwoord: Zodat er controles binnen de methode kunnen worden uitgevoerd voordat er items in of uitgehaald worden.

3.1.B

Vraag: In welke situatie gebruik je een HashMap en wanneer een HashSet?

Antwoord: Een HashMap word gebruikt indien bij de ingevoerde keys, values moeten worden gekoppeld. HashSet word gebruikt als er alleen indexes of keys worden geplaatst.

3.3.A

Vraag: Leg de werking van de constructor uit.

Antwoord: De constructor wordt aan geroepen op het moment dat een object van de classe (waar de constructer in staat) word aangemaakt, waar eventueel variables in worden geinitialiseerd.

3.3.B

Vraag: Leg de werking van int getRandomValue(int min, int max) uit en met name waarom er +1 in voorkomt. Gebruik de Java API. Hint: denk aan de betekenis van inclusief en exclusief.

Antwoord: De methode getRandomValue geeft een random int terug die tussen de ingevulde int min en int max zit. Dit +1 word erin gezet, omdat negatieve getallen te voorkomen.

4.1.C

Vraag: Er is geen constructor gedefinierd voor Administratie terwijl je gewoon new Administratie() kan aanroepen. Leg uit waarom dat kan.

Antwoord: Er is wel een constructer gedefineerd in de class Administratie. Deze is private, zodat er geen Administratie object aangemaakt kan worden. Omdat de methodes van de classe Administratie static zijn, hoeft er geen object aangemaakt te worden.

4.1.D

Vraag: Leg uit waarom de twee al bestaande methoden van Administratie static kunnen zijn. Verander ze in static.

Antwoord: De methodes kunnen static zijn omdat er geen object van aangemaakt word.

4.1.E

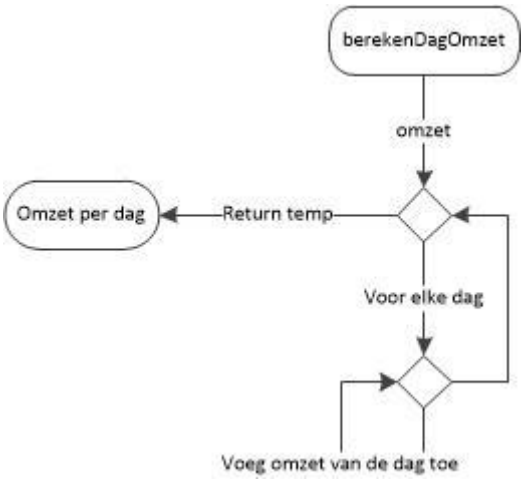
Vraag: We hebben door het static maken van de twee methodes geen instantie meer nodig van Administratie. Het is echter wel mogelijk om een instantie van Administratie aan te maken en daar de static methoden op aan te roepen. Als je dat wil voorkomen kun je een private constructor voor Administratie maken. Doe dat en leg uit waarom je je doel nu bereikt.

Antwoord: Door de constructer private te maken, kan de constructor methode niet van buiten af aangeroepen worden. Hierdoor kan er geen instantie aangemaakt worden, en zal er een foutmelding als resultaat volgen.

4.1.G

Vraag: Maak een activiteitendiagram van je implementatie van de vorigevraagd.

Antwoord:



4.1.H

Vraag: Leg uit wat final doet.

Antwoord: Final zorgt ervoor dat een variable niet meer veranderd kan worden.

4.1.I

Vraag: Als het goed is klaagt de compiler over zoiets als “Cannot make a static reference to the non-static field…”. Leg uit waarom de compiler hierover klaagt.

Antwoord: Er kunnen geen instantie velden of methodes worden aangeroepen vanuit een statische methode. Dit is wel het geval, en daar komt de foutmelding vandaan.

4.1.J

Vraag: Een manier om het probleem te verhelpen is om het woord final te vervangen door static. Waarschijnlijk compileert het werk nu wel weer, maar is het niet meer goed. Welk “probleem” heb je nu geïntroduceerd? Hint: wat was nou ook alweer de oorspronkelijke aanleiding om days_in_week te introduceren?

Antwoord: Nu is de ingestelde waarde aan te passen, en dat is niet wat de bedoeling is.

4.2.B

Vraag: Waarom moet een super aanroep in de constructor altijd bovenaanstaan?

Antwoord: Omdat dan het parent object aangemaakt wordt.

5.1.C

Vraag: Maak een klasse PersoonsVergelijker en gebruik een `public static void main(String[] args)` methode waarin je twee identieke Persoon objecten maakt via new; identiek wil zeggen dat de waarden van alle eigenschappen van de objecten hetzelfde is. Test op == en equals. Wat concludeer je over het verschil tussen het van deze twee manieren?

Antwoord: Als er gebruikt gemaakt word van equals, dan word de inhoud van de instanties vergeleken. Als er gebruik gemaakt word van ==, dan word het geheugen adres met elkaar vergeleken, en deze zijn altijd verschillend. Indien er gebruik gemaakt word van == om instanties te vergelijken, zal er altijd false uit komen.

5.1.D

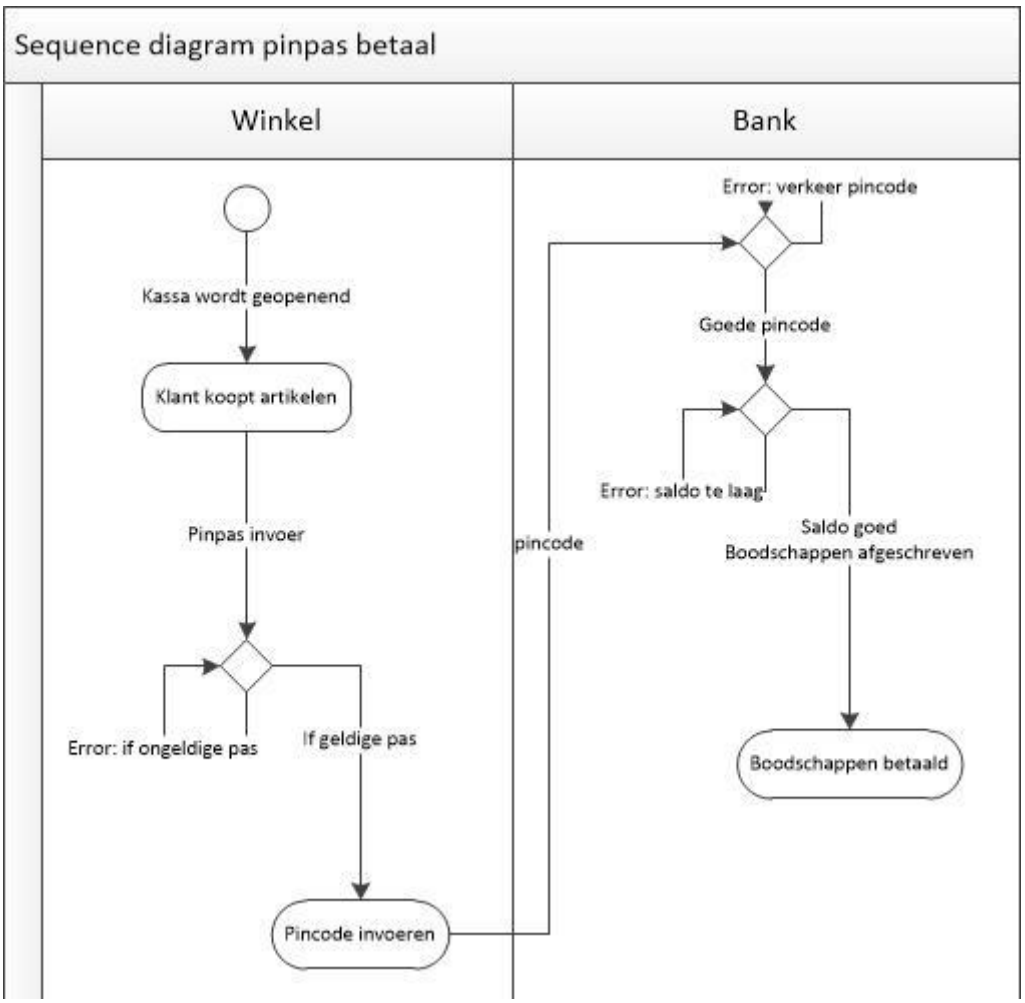
Vraag: Als je twee string inhoudelijk met elkaar wil vergelijken, moet je dan == of equals(Object object) gebruiken. Licht je antwoord toe. Geef aan wat er gebeurt als je de andere mogelijkheid zou kiezen.

Antwoord: Er moet gebruik gemaakt worden van equals (Object object). Kijk voor uitleg bij de vorige vraag.

5.2.B sequence-diagram

Vraag: Stel dat bij de pinpasbetaling zou worden gecommuniceerd met een Bank object. Teken een sequentie-diagram waaruit blijkt hoe je de methode `boolean betaal(double tebetalen)` in Pinpas implementeert. Hint: vergeet niet hoe de klasse Pinpas een referentie naar een Bank object heeft gekregen.

Antwoord:



5.2.C

Vraag: Waarom is de instantie variable saldo protected gemaakt? Waarom is dat handig?

Antwoord: Zodat de variable van buiten de huidige en subclasses niet aangepast kan worden of worden opgehaald zonder methode erbij. Dit is handig om eventuele foutieve informatie erin te vookomen.

5.5 A

Vraag: Kun je een instantie maken van een interface via new? Leg uit waarom het logisch is dat het wel of niet kan.

Antwoord: Nee dit is niet mogelijk, een klasse implementeert een interface.

5.5 B

Vraag: Herhaal de vorige vraag met abstracte klassen.

Antwoord: Nee, dit is niet mogelijk. Een abstract klas functioneert als een basis voor een subclass.

5.5 C

Vraag: Kan een klasse meerdere klasse overerven?

Antwoord: Nee

5.5 D

Vraag: Kan een klasse meerdere interfaces implementeren?

Antwoord: Ja

5.5 E

Vraag: Kan een klasse tegelijk een klasse overerven en interfaces implementeren?

Antwoord: Ja

5.5 F

Vraag: Klopt de stelling dat elke methode in een interface abstract is? Licht je antwoord toe.

Antwoord: Nee, interface klassen en abstracte klassen zijn twee verschillende dingen. Abstracte interfaces bestaan niet.

5.5 G

Vraag: Moet een klasse abstract zijn als minstens één methode abstract is? Licht je antwoord toe.

Antwoord: Ja, de klasse moet abstract zijn om foutmeldingen tijdens het compilen te voorkomen.

5.5 H

Vraag: Leg het begrip polymorfisme van klassen uit en geef twee voorbeelden (één met abstracte klassen, en één met interfaces).

Antwoord:

5.6 A

Vraag: Kan een klasse abstract zijn als geen enkele methode abstract is in die klasse? Probeer het eens uit. Leg waarom het logisch is dat dit wel of niet kan.

Antwoord: Ja, dit is mogelijk. Een abstract klasse is een basis voor een subklasse. Als een methode altijd het zelfde uitvoert, hoeft deze niet abstract te zijn. Als dit voor alle methodes geld, bestaan er geen abstracte methodes in de abstract klas.

5.6

Vraag: Moet een subklasse van een abstracte klasse altijd alle abstracte methodes implementeren? Leg uit waarom het logisch is dat dit wel of niet kan.

Antwoord: Ja, de methodes worden als het ware doorverwezen naar de subklasse.

5.6 C

Vraag: Als een klasse niet alle methode van een interface implementeert kun je iets doen om een (compiler)fout te voorkomen. Wat? Waarom is de oplossing logisch?

Antwoord: Er een abstracte klasse van maken. Hierdoor word een subklasse aangemaakt die de missende methodes aanvult.

5.6 D

Vraag: Leg uit waarom het logisch is dat een instantie variable niet abstract kan zijn.

Antwoord: Abstracte klassen dienen vanuit een andere klasse te worden geëxtend. Er kan geen instantie gemaakt worden van een abstracte klassen. Hierdoor is het onmogelijk om een instantie variable abstract te maken.

5.6 E

Vraag: (Uitdaging) Zoek uit wat een final methode is. Leg daarna uit waarom het logisch is dat een methode niet tegelijkertijd abstract en final kan zijn.

Antwoord: Als bijvoorbeeld de constructer een final methode is, kan er geen subklassen van gemaakt worden. Abstract methodes werken met subklassen, waardoor de combinatie hiervan niet goed loopt.