|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum examen 2 juni 2020 | |  | | |
| Academiejaar 2019-2020 | | | | |
| Voornaam en naam | Sibian De Gussem | | | |
| Studentennummer | 201970377 | | | |
| Opleiding | Toegepaste informatica | | **Jaar** 1 TI / 1 TILE | |
| Afstudeerrichting |  | | | |
| Opleidingsonderdeel | OO Programmeren II | | | |
| Lesgever(s) | L. Lewyllie, A. Pierreux, H. Roobrouck, L. Stroobant,  S. Vandermeersch, L. Vuyge | | | |
|  |  | | | |
|  | | | | **Behaald resultaat**  **/** |

Tijdens het examen mogen deze hulpmiddelen gebruikt worden:

* Java How To Program (Early objects) – 11e editie (of eerder) - Deitel & Deitel Prentice Hall
* API-documentatie:

<https://docs.oracle.com/en/java/javase/11/docs/api/index.html>  
<https://openjfx.io/javadoc/11/>

* Ter beschikking gestelde startprojecten

|  |
| --- |
| Algemene richtlijnen |
| Vul het kader bovenaan in (enkel voor het antwoordformulier). |
| Je vindt de examenvragen in het opdrachtformulier. Beantwoord de vragen op het aparte antwoordformulier. Volg hierbij de lay-out van dit antwoordformulier.  Lees voor de start aandachtig de aandachtspunten op volgende pagina! |
| Heb je individuele onderwijs- en examenmaatregelen, noteer dan na je naam en voornaam in het kader bovenaan tussen haakjes ‘IOEM’ (afkorting voor individuele onderwijs- en examenmaatregelen). |
| Je mag geen enkele vorm van communicatie – noch offline noch online – gebruiken tijdens dit examen, tenzij anders aangegeven in de exameninstructies. Mobiele telefoons, smartwatches en dergelijke moeten uitgeschakeld zijn (niet op stil, trillen, vliegtuigstand, …). Ze mogen tijdens het examen ook niet gebruikt worden om de tijd te raadplegen. Het niet volgen van de gedragscode wordt gesanctioneerd als examenfraude. |

**Kopieer elke klasse naar de juiste plaats (geen imports)**

**Vraag 1. (70 punten)**

1. Interface MilieuBelasting

package domein;

public interface MilieuBelasting {

public final int MILEUHEFFING\_ALLEENSTAANDE = 30;

public final int MILEUHEFFING\_GEZIN = 50;

int berekenMilieuheffing();

}

1. Klasse Boot

package domein;

import exceptions.FormaatException;

public class Boot implements Comparable<Boot> {

private final String naam;

private boolean zeewaardig;

public Boot() throws FormaatException {

this("2000\_Onbekend", true);

}

public Boot(String naam, boolean zeewaardig) throws FormaatException {

super();

controleerNaam(naam);

this.naam = naam;

setZeewaardig(zeewaardig);

}

private void controleerNaam(String naam) throws FormaatException {

if (naam == null || naam.isEmpty()) {

throw new IllegalArgumentException("Naam moet ingevuld zijn");

} else {

if (!naam.matches("^\\d{4}\_[A-Z][a-z]{2,}$")) {

throw new FormaatException("Naam voldoet niet aan het opgelegde formaat");

}

}

}

public String getNaam() {

return naam;

}

public boolean getZeewaardig() {

return zeewaardig;

}

private void setZeewaardig(boolean zeewaardig) {

this.zeewaardig = zeewaardig;

}

@Override

public String toString() {

return String.format("%s: naam=%s, zeewaardig=%s", getClass().getSimpleName(), naam, zeewaardig ? "ja" : "nee");

}

@Override

public int compareTo(Boot b) {

int jaar1 = Integer.parseInt(naam.split("\_")[0]);

int jaar2 = Integer.parseInt(b.getNaam().split("\_")[0]);

int result = jaar2-jaar1;

if (result == 0) {

result = naam.compareTo(b.getNaam());

}

return result;

}

}

1. Klasse ZeilBoot

package domein;

import exceptions.FormaatException;

public class ZeilBoot extends Boot {

private int aantalMasten;

public ZeilBoot() throws FormaatException {

this("2000\_onbekend", true, 1);

}

public ZeilBoot(String naam, boolean zeewaardig, int aantalMasten) throws FormaatException {

super(naam, zeewaardig);

setAantalMasten(aantalMasten);

}

public int getAantalMasten() {

return aantalMasten;

}

private void setAantalMasten(int aantalMasten) {

if (aantalMasten <= 0) {

throw new IllegalArgumentException("Een zeilboot heeft minimaal 1 mast");

}

this.aantalMasten = aantalMasten;

}

@Override

public String toString() {

return String.format("%s, aantal masten=%d", super.toString(), aantalMasten);

}

}

1. Klasse WoonBoot

package domein;

import exceptions.FormaatException;

public class WoonBoot extends Boot implements MilieuBelasting {

private int aantalPersonen;

private int aantalBadkamers;

public WoonBoot() throws FormaatException {

this("2000\_Onbekend", false, 1, 1);

}

public WoonBoot(String naam, boolean zeewaardig, int aantalPersonen, int aantalBadkamers) throws FormaatException {

super(naam, zeewaardig);

setAantalPersonen(aantalPersonen);

setAantalBadkamers(aantalBadkamers);

}

public int getAantalPersonen() {

return aantalPersonen;

}

private void setAantalPersonen(int aantalPersonen) {

if (aantalPersonen <= 0) {

throw new IllegalArgumentException("Een woonboot heeft minimaal 1 persoon aan boord");

}

this.aantalPersonen = aantalPersonen;

}

public int getAantalBadkamers() {

return aantalBadkamers;

}

private void setAantalBadkamers(int aantalBadkamers) {

if (aantalBadkamers < 0) {

throw new IllegalArgumentException("Een woonboot kan geen negatieve badkamers hebben");

}

this.aantalBadkamers = aantalBadkamers;

}

@Override

public String toString() {

return String.format("%s, aantal personen=%d, aantal badkamers=%d, milieuheffing=%d", super.toString(), aantalPersonen, aantalBadkamers, berekenMilieuheffing());

}

@Override

public int berekenMilieuheffing() {

if (aantalPersonen == 1) {

return MILEUHEFFING\_ALLEENSTAANDE;

} else {

return MILEUHEFFING\_GEZIN;

}

}

}

1. Klasse BootRepository

package domein;

import java.util.Arrays;

import java.util.Comparator;

import java.util.List;

import exceptions.FormaatException;

import persistentie.BootMapper;

public class BootRepository {

private BootMapper bootMapper;

private List<Boot> botenlijst;

public BootRepository() throws FormaatException {

bootMapper = new BootMapper();

botenlijst = bootMapper.geefBoten();

}

public List<Boot> getBotenlijst() {

return botenlijst;

}

public void setBotenlijst(List<Boot> botenlijst) {

this.botenlijst = botenlijst;

}

public void voegBootToe(Boot boot) {

botenlijst.add(boot);

}

public void voegZeilBootToe(ZeilBoot zeilBoot) {

botenlijst.add(zeilBoot);

}

public void voegWoonBootToe(WoonBoot woonboot) {

botenlijst.add(woonboot);

}

public List<Boot> geefGesorteerdeOverzichtBoten(){

Boot[] botenArray = new Boot[botenlijst.size()];

botenArray = botenlijst.toArray(botenArray);

Arrays.sort(botenArray);

return Arrays.asList(botenArray);

}

}

1. Klasse DomeinController

package domein;

import java.util.List;

import exceptions.FormaatException;

public class DomeinController {

private BootRepository bootRepository;

public DomeinController() throws FormaatException {

bootRepository = new BootRepository();

}

public void voegBootToe(String[] gegevens) throws FormaatException {

if (gegevens.length == 2) {

bootRepository.voegBootToe(new Boot(gegevens[0], Boolean.parseBoolean(gegevens[1])));

} else if (gegevens.length == 3) {

bootRepository.voegZeilBootToe(new ZeilBoot(gegevens[0], Boolean.parseBoolean(gegevens[1]), Integer.parseInt(gegevens[2])));

} else if (gegevens.length == 4) {

bootRepository.voegWoonBootToe(

new WoonBoot(gegevens[0], Boolean.parseBoolean(gegevens[1]), Integer.parseInt(gegevens[2]), Integer.parseInt(gegevens[3]

)));

}

}

public String[] geefGesorteerdOverzichtBoten() {

List<Boot> botenList = bootRepository.geefGesorteerdeOverzichtBoten();

String[] botenString = new String[botenList.size()];

int index = 0;

for (Boot b: botenList) {

botenString[index++] = b.toString();

}

return botenString;

}

}

1. Klasse in de package exceptions

package exceptions;

public class FormaatException extends Exception {

public FormaatException() {

this("respecteer interval");

}

public FormaatException(String msg) {

super(msg);

}

public FormaatException(Throwable cause) {

super(cause);

}

public FormaatException(String msg, Throwable cause) {

super(msg, cause);

}

}

1. Klasse StartUp

package main;

import domein.DomeinController;

import exceptions.FormaatException;

import ui.RederijApplicatie;

public class StartUp {

public static void main(String[] args) throws FormaatException {

new RederijApplicatie(new DomeinController()).start();

}

}

1. Klasse RederijApplicatie

package ui;

import java.util.InputMismatchException;

import java.util.Scanner;

import domein.DomeinController;

import exceptions.FormaatException;

public class RederijApplicatie {

private DomeinController dc;

public RederijApplicatie(DomeinController dc) {

this.dc = dc;

}

public void start() throws FormaatException {

drukTabelAf(dc.geefGesorteerdOverzichtBoten());

int keuze;

do {

keuze = maakKeuzeUitMenu();

switch (keuze) {

case 1:

voegBootToe();

break;

case 2:

voegZeilBootToe();

break;

case 3:

voegWoonBootToe();

break;

case 0:

drukTabelAf(dc.geefGesorteerdOverzichtBoten());

break;

}

} while (keuze != 0);

}

public void voegBootToe() throws FormaatException {

dc.voegBootToe(new String[]{leesGegeven("naam", "string"), leesGegeven("zeewaardig", "boolean")});

}

public void voegZeilBootToe() throws FormaatException {

dc.voegBootToe(new String[]{leesGegeven("naam", "string"), leesGegeven("zeewaardig", "boolean"), leesGegeven("aantal masten", "int")});

}

public void voegWoonBootToe() throws FormaatException {

dc.voegBootToe(new String[]{leesGegeven("naam", "string"), leesGegeven("zeewaardig", "boolean"), leesGegeven("aantal personen", "int"), leesGegeven("aantal badkamers", "int")});

}

private String leesGegeven(String wat, String datatype) {

Scanner scanner = new Scanner(System.in);

System.out.printf("Geef de waarde voor %s: ", wat);

String string = scanner.next();

if (datatype.equals("int")) {

// niet duidelijk wat dit moet doen? Voorbeeld zou handig geweest zijn.

} else if (datatype.equals("boolean")) {

// niet duidelijk wat dit moet doen? Voorbeeld zou handig geweest zijn.

} else if (datatype.equals("string")) {

// niet duidelijk wat dit moet doen? Voorbeeld zou handig geweest zijn.

}

return string;

}

private void drukTabelAf(String[] overzicht) {

for(String bootString: overzicht) {

System.out.println(bootString);

}

}

private int maakKeuzeUitMenu() {

Scanner scanner = new Scanner(System.in);

boolean herhalen = true;

int keuze = -1;

while (herhalen) {

System.out.printf("Menu%n1. Voeg boot toe%n2. Voeg zeilboot toe%n3. Voeg woonboot toe%n0. Stoppen%nMaak je keuze: ");

try {

keuze = scanner.nextInt();

if (keuze >= 0 && keuze <= 3) {

herhalen = false;

}

} catch (InputMismatchException ime) {

System.err.println("Voer een getal in!");

}

}

return keuze;

}

}