Tijdens het examen mogen geen hulpmiddelen gebruikt worden.

Tijdens het examen mogen deze hulpmiddelen gebruikt worden:

* + - 1. Java documentatie: <https://docs.oracle.com/en/java/javase/14/docs/api/> of eventueel <https://docs.oracle.com/en/java/javase/15/docs/api/> of <https://docs.oracle.com/en/java/javase/16/docs/api/>
      2. JavaFX documentatie: <https://openjfx.io/javadoc/15/>
      3. Java How To Program (Early objects) – 11e editie (of eerder) - Deitel & Deitel Prentice Hall

|  |
| --- |
| Algemene richtlijnen |
| Vul je naam, voornaam en studentennummer in het kader bovenaan in. |
| Je vindt de examenvragen in het bestand Opgave.docx. Beantwoord de vragen op dit apart antwoordformulier. Geef hierbij duidelijk aan op welke vraag je antwoordt. |
| Heb je individuele onderwijs- en examenmaatregelen, noteer dan in de rechterbovenhoek van je antwoordpapier ‘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. |

**ANTWOORDFORMULIER**

Kopieer telkens de inhoud van **de volledige klasse** in het overeenkomstig hokje.

*Je hoeft de package declaratie en de imports niet mee te kopiëren.*

**Vraag 1.**

## De interface Opmeetbaar

|  |
| --- |
| **public interface Opmeetbaar {**    **public abstract double geefOppervlakte();**    **public abstract double geefVrijeOppervlakte();**    **}** |

## De klassen BloemenperkException en ToestelException

|  |
| --- |
| public class BloemenperkException extends Exception{  public BloemenperkException() {  super();  // TODO Auto-generated constructor stub  }  public BloemenperkException(String message, Throwable cause, boolean enableSuppression,  boolean writableStackTrace) {  super(message, cause, enableSuppression, writableStackTrace);  // TODO Auto-generated constructor stub  }  public BloemenperkException(String message, Throwable cause) {  super(message, cause);  // TODO Auto-generated constructor stub  }  public BloemenperkException(String message) {  super(message);  // TODO Auto-generated constructor stub  }  public BloemenperkException(Throwable cause) {  super(cause);  // TODO Auto-generated constructor stub  }  public class ToestelException extends RuntimeException{  public ToestelException() {  super();  // TODO Auto-generated constructor stub  }  public ToestelException(String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace) {  super(message, cause, enableSuppression, writableStackTrace);  // TODO Auto-generated constructor stub  }  public ToestelException(String message, Throwable cause) {  super(message, cause);  // TODO Auto-generated constructor stub  }  public ToestelException(String message) {  super(message);  // TODO Auto-generated constructor stub  }  public ToestelException(Throwable cause) {  super(cause);  // TODO Auto-generated constructor stub  } |

## De abstracte klasse Tuinperceel

|  |
| --- |
| public abstract class Tuinperceel implements Opmeetbaar{  private static final BigDecimal EENHEIDSWAARDE = new BigDecimal(2.5);  protected static final int DEFAULT\_LENGTE\_BREEDTE = 5;    private final double lengte, breedte;    public Tuinperceel() {  this.lengte = DEFAULT\_LENGTE\_BREEDTE;  this.breedte = DEFAULT\_LENGTE\_BREEDTE;  }  public Tuinperceel(double lengte, double breedte) {  controleerLengteEnBreedte(lengte, breedte);  this.lengte = lengte;  this.breedte = breedte;  }      public double geefOppervlakte() {  return this.lengte\*this.breedte;    }    public BigDecimal geefWaarde() {  return EENHEIDSWAARDE.multiply(new BigDecimal(geefOppervlakte()));    }    private void controleerLengteEnBreedte(double lengte, double breedte) {  if (lengte < 1.0 || breedte < 1.0)  throw new IllegalArgumentException("De lengte en breedte moeten strikt positief zijn");  if (lengte < breedte)  throw new IllegalArgumentException("Lengte moet groter dan of gelijk zijn aan breedte");    }    @Override  public String toString() {    return String.format("Tuinperceel van %.1f op %.1f met een waarde van %.2f", this.lengte, this.breedte, this.geefWaarde().doubleValue());  } |
|  |

## De klasse Siertuinperceel

|  |
| --- |
| public class Siertuinperceel extends Tuinperceel {  private final List<String> bloemenperken;  private final boolean vijverAanwezig;    public Siertuinperceel(List<String> bloemenperken, boolean vijverAanwezig) throws BloemenperkException {  super();  if (bloemenperken == null)  this.bloemenperken = new ArrayList<>();  else this.bloemenperken = bloemenperken;      if (bloemenperken.size() > (this.DEFAULT\_LENGTE\_BREEDTE\*this.DEFAULT\_LENGTE\_BREEDTE))  throw new BloemenperkException("Te veel bloemenperken voor dit perceel");      this.vijverAanwezig = vijverAanwezig;  }  public Siertuinperceel(double lengte, double breedte,List<String> bloemenperken, boolean vijverAanwezig) throws BloemenperkException {  super(lengte, breedte);  if (bloemenperken == null)  this.bloemenperken = new ArrayList<>();  else this.bloemenperken = bloemenperken;    if (bloemenperken.size() > (lengte \* breedte))  throw new BloemenperkException("Te veel bloemenperken voor dit perceel");  this.vijverAanwezig = vijverAanwezig;  }    public List<String> getBloemenperken() {  return bloemenperken;  }    public boolean isVijverAanwezig() {  return vijverAanwezig;  }      public BigDecimal geefWaarde() {  if (isVijverAanwezig())  return this.geefWaarde().add(new BigDecimal(50.0));  return this.geefWaarde();  }    @Override  public double geefVrijeOppervlakte() {  double returnvalue = geefOppervlakte();  double bloemenperkenreductie = 0;  double vijverreductie = 0;    if (bloemenperken.isEmpty() == false)  bloemenperkenreductie = 1 \* this.bloemenperken.size();    if (isVijverAanwezig())  vijverreductie = 5;    return (returnvalue - bloemenperkenreductie - vijverreductie);      }    @Override  public String toString() {  String vijver = "";  if (isVijverAanwezig())  vijver = "met vijver";  else vijver = "zonder vijver" ;      return String.format("%s %s",super.toString(), vijver);  } |

## De klasse Speeltuinperceel

|  |
| --- |
| public class Speeltuinperceel extends Tuinperceel {  private final int aantalToestellen;        public Speeltuinperceel(int aantalToestellen) {  super();    if (aantalToestellen > ((this.DEFAULT\_LENGTE\_BREEDTE\*DEFAULT\_LENGTE\_BREEDTE))/3)  throw new ToestelException("Er past niet meer dan 1 toestel per 3 vierkante meter");  else this.aantalToestellen = aantalToestellen;  }  public Speeltuinperceel(double lengte, double breedte, int aantalToestellen) {  super(lengte, breedte);  if (aantalToestellen > ((lengte\*breedte))/3)  throw new ToestelException("Er past niet meer dan 1 toestel per 3 vierkante meter");  else this.aantalToestellen = aantalToestellen;    }  public BigDecimal geefWaarde() {  return this.geefWaarde().add(new BigDecimal(10\*this.aantalToestellen));    }  @Override  public double geefVrijeOppervlakte() {      return (this.geefOppervlakte() - (1 \* this.aantalToestellen));  }    @Override  public String toString() {    return String.format("%s en met %d toestellen om te spelen", super.toString(), this.aantalToestellen);  } |

## De klasse GrootteEnWaardeVergelijker

|  |
| --- |
| public class GrootteEnWaardeVergelijker implements Comparator<Tuinperceel> {  @Override  public int compare(Tuinperceel tp1, Tuinperceel tp2) {  if (Double.valueOf(tp1.geefOppervlakte()).compareTo(tp2.geefOppervlakte()) == 0)  return tp1.geefWaarde().compareTo(tp2.geefWaarde());  else return Double.valueOf(tp1.geefOppervlakte()).compareTo(tp2.geefOppervlakte());  } |

## De klasse Tuin

|  |
| --- |
| public class Tuin implements Opmeetbaar {  private List<Tuinperceel> tuinpercelen;  public Tuin(Collection<Tuinperceel> tuinpercelen) {  super();  this.tuinpercelen = (ArrayList<Tuinperceel>) tuinpercelen;  }  @Override  public double geefOppervlakte() {  double returnwaarde = 0;    for (Tuinperceel tuinperceel : tuinpercelen)  returnwaarde += tuinperceel.geefOppervlakte();    return returnwaarde;      }  @Override  public double geefVrijeOppervlakte() {  double returnwaarde = 0;    for (Tuinperceel tuinperceel : tuinpercelen)  returnwaarde += tuinperceel.geefVrijeOppervlakte();    return returnwaarde;  }    public List<Tuinperceel> geefAlleTuinpercelenGesorteerd() {  List<Tuinperceel> returnlist = tuinpercelen;    Collections.sort(returnlist, new GrootteEnWaardeVergelijker());    return returnlist;    }    public List<Siertuinperceel> geefAlleSiertuinpercelen() {  List<Siertuinperceel> list = new ArrayList<>();    for (Tuinperceel tuinperceel : tuinpercelen)  if (tuinperceel instanceof Siertuinperceel)  list.add((Siertuinperceel)tuinperceel);    return list;    } |

## De klasse DomeinController

|  |
| --- |
| public class DomeinController {  private Tuin tuin;    public void maakTuin(int id) throws BloemenperkException {  tuin = new Tuin(new TuinperceelRepository().geefTuinpercelen(id));    }    public String geefGesorteerdeTuinpercelen() {    String lijst = tuin.geefAlleTuinpercelenGesorteerd().stream().map(e-> String.format("%s%n", e.toString())).collect(Collectors.joining(""));  String output = String.format("Alle tuinpercelen gesorteerd volgens grootte en waarde:%n%s", lijst);    return output;  }    public String geefSiertuinpercelen() {  String lijst = tuin.geefAlleSiertuinpercelen().stream().map(e-> String.format("%s%n", e.toString())).collect(Collectors.joining(""));  String output = String.format("Alle siertuinpercelen:%n%s", lijst);    return output;  }    public double geefOppervlakte() {    return tuin.geefOppervlakte();    }    public double geefVrijeOppervlakte() {    return tuin.geefVrijeOppervlakte();    } |

## De klasse StartUp

|  |
| --- |
| public class StartUp {  public static void main(String[] args) {  new TuinApp(new DomeinController()).start();  } |

## De klasse TuinApp

|  |
| --- |
| public class TuinApp {  private DomeinController dc;  public TuinApp(DomeinController dc) {  super();  this.dc = dc;  }    public void start() {  try {  dc.maakTuin(1);  } catch (IllegalArgumentException e) {  System.err.printf("Algemeen probleem: %s", e.getMessage());  System.exit(1);  } catch (BloemenperkException e) {  System.err.printf("Probleem met bloemenperken: %s", e.getMessage());  System.exit(1);  } catch (ToestelException e) {  System.err.printf("Probleem met toestellen: %s", e.getMessage());  System.exit(1);  }    toonEnReageerOpMenu();  }    private void toonEnReageerOpMenu() {  String[] keuzes = { "Toon alle siertuinpercelen", "Toon alle percelen gesorteerd op grootte en waarde",  "Geef totale oppervlakte van de tuin", "Geef vrije oppervlakte in de tuin", "Stop" };  int keuze = maakKeuzeUitMenu(keuzes);  switch (keuze) {  case 1:  System.out.println("\nAlle siertuinpercelen:");  System.out.println(dc.geefSiertuinpercelen());  break;  case 2:  System.out.println("\nAlle tuinpercelen gesorteerd volgens grootte en waarde:");  System.out.println(dc.geefGesorteerdeTuinpercelen());  break;  case 3:  System.out.printf("%nDe totale oppervlakte van de tuin is %.2f vierkante meter", dc.geefOppervlakte());  break;  case 4:  System.out.printf("%nDe vrije oppervlakte in de tuin is %.2f vierkante meter groot",  dc.geefVrijeOppervlakte());  break;  default:  break;  }    }    private int maakKeuzeUitMenu(String[] keuzes) {    List<String> keuzelijst = new ArrayList<>(Arrays.asList(keuzes));    String menu = "";    for (String string :keuzelijst)  menu += String.format("%s. %s%n", keuzelijst.indexOf(string)+1, string);    Scanner sc = new Scanner(System.in);  boolean loopflag = true;    int input = 0;      do {  try {  System.out.println(menu);  System.out.println("Je keuze (1-5): ");  input = sc.nextInt();  if (input < 1 || input > 5)  throw new IllegalArgumentException();  else  loopflag = false;    } catch (InputMismatchException e) {  System.out.println("Geef een getal in!");  sc.next();  } catch (IllegalArgumentException e) {  System.out.println("Geef een getal tussen 1 en 5.");  }  } while (loopflag);    return input;  } |

**Vraag 2.**

## De klasse UitgaveRepository

|  |
| --- |
| public class UitgaveRepository {  private List<Uitgave> uitgaveLijst;  private final UitgaveMapper uitgaveMapper;  public UitgaveRepository() {  uitgaveMapper = new UitgaveMapper();  // Maak een lijst van Uitgave objecten aan op basis van  // de elementen vanuit de mapper (uitgaveMapper.geefUitgaven())  // Deze lijst van Uitgave objecten MOET uitbreidbaar zijn.  uitgaveLijst = new ArrayList<>(Arrays.asList(uitgaveMapper.geefUitgaven()));    // Sorteer hier de uitgaveLijst volgens 'natuurlijke orde' (= volgens het jaar, oud naar nieuw)  // (--> hiervoor dien je ook ook de klasse Uitgave aan te passen).    uitgaveLijst = uitgaveLijst.stream().sorted().collect(Collectors.toList());    }  public List<Uitgave> getUitgaveLijst() {  return uitgaveLijst;  }    // Geef een gesorteerde lijst van de uitgaven terug,  // alfabetisch gesorteerd op titel en bij gelijke titel op categorie.  public List<Uitgave> geefGeordendOpTitelCategorie() {    Comparator<Uitgave> comp = new Comparator<Uitgave>() {  @Override  public int compare(Uitgave o1, Uitgave o2) {  if (o1.getTitel().compareTo(o2.getTitel()) ==0)  return o1.getCategorie().compareTo(o2.getCategorie());  else return o1.getTitel().compareTo(o2.getTitel());  }  };    return uitgaveLijst.stream().sorted(comp).collect(Collectors.toList());    }  // Als twee uitgave objecten hetzelfde isbn nummer hebben, dan gaat het om dezelfde uitgave.  // Pas de Uitgave klasse zodanig aan dat je twee Uitgave objecten met elkaar kan vergelijken.  // Haal de dubbels (uitgaves met hetzelfde isbn) uit de lijst  public Collection<Uitgave> geefZonderDubbels() {    return uitgaveLijst.stream().distinct().collect(Collectors.toList());    }  // Tel het aantal uitgaves waarvan de categorie het opgegeven woord bevat.  // Let op: Bij het zoeken of een woord voorkomt in een categorie moet je  // weten dat de categorie van een uitgave altijd in hoofdletters staat.  public long geefAantalCategorie(String woord) {  String nieuwwoord = woord.toUpperCase();    return uitgaveLijst.stream().filter(e-> e.getCategorie().matches(nieuwwoord)).count();  }  // Geef een lijst van de unieke titels van alle uitgaven, de recentste bovenaan  public List<String> geefTitels() {    return uitgaveLijst.stream().sorted(Comparator.comparing(Uitgave::getJaar).reversed()).distinct().map(e-> e.getTitel()).collect(Collectors.toList());      }  // Geef de titel van de uitgave met een bepaald isbn nummer. Als er geen  // resultaat gevonden wordt, geef je "GEEN RESULTAAT" terug.  public String geefTitelMetISBN(String isbn) {    return uitgaveLijst.stream().filter(e-> e.getISBN() == isbn).findAny().orElse(null).toString();    } |

## De klasse Uitgave

|  |
| --- |
| package domein;  public class Uitgave implements Comparable<Uitgave> {    private int jaar;  private String titel;  private String isbn;  private String categorie;    public Uitgave(int jaar, String titel, String isbn, String categorie) {  this.jaar = jaar;  this.titel = titel;  this.isbn = isbn;  this.categorie = categorie;  }  public int getJaar() {  return jaar;  }  public void setJaar(int id) {  this.jaar = id;  }  public String getTitel() {  return titel;  }  public void setTitel(String titel) {  this.titel = titel;  }  public String getISBN() {  return isbn;  }  public void setISBN(String isbn) {  isbn = isbn;  }  @Override  public int hashCode() {  final int prime = 31;  int result = 1;  result = prime \* result + ((isbn == null) ? 0 : isbn.hashCode());  return result;  }  @Override  public boolean equals(Object obj) {  if (this == obj)  return true;  if (obj == null)  return false;  if (getClass() != obj.getClass())  return false;  Uitgave other = (Uitgave) obj;  if (isbn == null) {  if (other.isbn != null)  return false;  } else if (!isbn.equals(other.isbn))  return false;  return true;  }  public String getCategorie() {  return categorie;  }  public void setCategorie(String categorie) {  this.categorie = categorie;  }  @Override  public String toString() {  return "titel: " + titel + ", ISBN: " + isbn + ", categorie: " + categorie + ", Jaar: " + jaar;  }  @Override  public int compareTo(Uitgave o) {  return Integer.valueOf(this.jaar).compareTo(o.getJaar());    }  } |

## De klasse DomeinController

|  |
| --- |
| private List<String> zetUitgavenOmNaarTekst(Collection<Uitgave> list) {    return list.stream().map(e-> e.toString()).collect(Collectors.toList());    } |

## De klasse ConsoleApplicatie

|  |
| --- |
| // Zet een willekeurige collectie van String objecten om naar een String,  // de beschrijving van elke element uit de collectie komt op een nieuwe lijn.  private String zetLijstOmNaarTekst(Collection<String> collectie) {    return collectie.stream().map(e-> String.format("%s%n", e)).collect(Collectors.joining(""));    }    // Een titel opgevraagd via het ISBN nummer wordt getoond op het scherm.  // Een titel is echter niet altijd correct ingegeven, soms staan er meerdere  // spaties of tabs tussen de woorden.  // Om een titel mooi te presenteren koos men ervoor om elk woord in een titel  // te laten starten met een hoofletter. De methode formatteerTitel zorgt hiervoor.  // Voorbeeld:  // De titel: "Python crash course" ziet er na formattering zo uit:  // "Python Crash Course"  private String formatteerTitel(String titel) {    String output;  List<String> list = new ArrayList<>(Arrays.asList(titel.split(" ")));  List <StringBuilder> sblist = new ArrayList<>();    list.stream().forEach(e-> sblist.add(new StringBuilder(e)));    sblist.stream().forEach(e-> e.setCharAt(0, Character.toUpperCase(e.charAt(0))));    output = sblist.stream().map(e-> e.toString()).collect(Collectors.joining(" "));  output = output.trim();    return output;    } |