Java programming fundamentals: herhalingsoefeningen

Je maakt alle classes in een package `be.vdab`

PersoonManager

Maak de nodige classes zodat volgende code werkt:

```
Personen personen = new Personen();
personen.add(new Persoon("Joe", "Dalton"));
personen.add(new Persoon("Sarah", "Bernhardt"));
PersoonManager manager = new PersoonManager();
manager.save(personen); // slaat op in een bestand op de harde schijf
Personen personen2 = manager.load(); // leest de data van het bestand terug
System.out.println(personen2);
```

output:

```
Joe Dalton
Sarah Bernhardt
```

Sorteren

Je maakt een verzameling unieke personen. Je toont deze verzameling gesorteerd op familienaam.
Je sorteert daarbij twee personen met dezelfde familienaam op voornaam.

Lokaal

Je maakt de nodige classes, zodat volgende code

```
Traject traject1 = new Traject(".java");
Traject traject2 = new Traject(".net");
Cursist cursist1 = new Cursist("Felix", "De Vliegher", traject1);
Cursist cursist2 = new Cursist("Koen", "Vanhoutte", traject2);
Cursist cursist3 = new Cursist("Serge", "Vereecke", traject2);
Cursist cursist4 = new Cursist("Freddy", "Himpe", traject1);
Trainer trainer = new Trainer("Hans", "Desmet");
Lokaal lokaal = new Lokaal(11, trainer);
lokaal.cursistToevoegen(cursist1);
// een lokaal heeft een variabel aantal cursisten
lokaal.cursistToevoegen(cursist2);
lokaal.cursistToevoegen(cursist3);
lokaal.cursistToevoegen(cursist4);
System.out.println(lokaal);
```

volgende output geeft:

```
Lokaal 11 Hans Desmet
4 cursisten:
Felix De Vliegher volgt java
Koen Vanhoutte volgt .net
Serge Vereecke volgt .net
Freddy Himpe volgt java
```

Landen

De gebruiker tikt per land een landcode en een aantal inwoners, tot hij stop tikt.

Jij toont deze informatie terug, gesorteerd op landcode.
Je toont daarna het totaal aantal inwoners per land.

PacMan

Je maakt de nodige classes zodat volgende code werkt:

```
Doolhof doolhof = new Doolhof();
Pacman pacman = new Pacman();
doolhof.add(pacman);
Monster inky = new Monster("inky");
doolhof.add(inky);
Monster pinky = new Monster("pinky");
doolhof.add(pinky);
Monster blinky = new Monster("blinky");
doolhof.add(blinky);
Monster clyde = new Monster("clyde");
doolhof.add(clyde);
System.out.println(doolhof); // pacman 3 leven(s), inky, pinky, blinky, clyde
blinky.meet(pacman);
System.out.println(doolhof); // pacman 2 leven(s), inky, pinky, blinky, clyde
pacman.eatPowerPill();
pacman.meet(inky);
clyde.meet(pacman);
System.out.println(doolhof); // pacman 2 leven(s), pinky, blinky
pacman.powerPillWorkedOut();
```

Woorden 1

De gebruiker tikt zinnen tot hij `stop` tikt.

De zinnen bevatten geen leestekens (punten, komma's, …) en worden in kleine letters getikt.

Jij toont een alfabetische lijst van de woorden met per woord hoeveel het voorkomt.

Woorden 2

De gebruiker tikt zinnen tot hij `stop` tikt.

De zinnen bevatten geen leestekens (punten, komma's, …) en worden in kleine letters getikt.

Jij toont een alfabetische lijst van de woorden met per woord in de hoeveelste zin(nen) dit voorkomt.

Kasbons

Je maakt de nodige classes zodat volgende code de output 220.2500 geeft.

Een niet-kapitaliseerbare kasbon heeft drie eigenschappen: beginwaarde, jaren en intrest.
Bij zo'n kasbon brengt ieder jaar intrest op als beginwaarde * intrest.

Een kapitaliseerbare kasbon heeft dezelfde eigenschappen.
Bij zo'n kasbon brengt ieder jaar intrest op als beginwaarde * intrest + intrestVorigejaren * intrest

```
Klant klant = new Klant();
klant.add(
  new NietKapitaliseerbareKasbon(BigDecimal.valueOf(100), 2,
    BigDecimal.valueOf(0.05)));
klant.add(
  new KapitaliseerbareKasbon(BigDecimal.valueOf(100), 2,
    BigDecimal.valueOf(0.05)));
System.out.println(klant.getEindWaarde());
```

Time

Je maakt de class Time zodat volgende code volgende output geeft

```
Time hour1 = new Time(9, 40);
Time hour2 = new Time(10, 30);
Time hour3 = hour2.subtract(hour1);
System.out.println(hour3);
Time hour4 = hour1.add(hour2);
System.out.println(hour4);

0:50
20:10
```

Voorbeeldoplossingen

PersoonManager

```
class Persoon implements Serializable {  // uitbreidingen class Persoon
    …
    public String getVoornaam() {
        return voornaam;
    }
    public String getFamilienaam() {
        return familienaam;
    }
    …
}

package be.vdab;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

class Personen implements Serializable {
    private final List<Persoon> personen = new ArrayList();
    public void add(Persoon persoon) {
        personen.add(persoon);
    }
    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        for (Persoon persoon : personen) {
            builder.append(persoon.getNaam()).append("\n");
        }
        return builder.toString();
    }
}
```

```
class PersoonManager {
    private final static String NAAM_BESTAND = "c:/data/personen.dat";
    public void save(Personen personen) {
        try (FileOutputStream fileStream = new FileOutputStream(NAAM_BESTAND);
                ObjectOutputStream objectStream =
                    new ObjectOutputStream(fileStream)) {
            objectStream.writeObject(personen);
        } catch (IOException ex) {
            System.out.println(ex);
        }
    }
    public Personen load() {
        try (FileInputStream fileStream = new FileInputStream(NAAM_BESTAND);
                ObjectInputStream objectStream =
                  new ObjectInputStream(fileStream)) {
            return (Personen) objectStream.readObject();
        } catch (Exception ex) {
            System.out.println(ex);
            return null;
        }
    }
}
```

Sorteren

```
class Persoon implements Serializable, Comparable<Persoon> {
    // uitbreidingen class Persoon
    …
    @Override
    public int compareTo(Persoon ander) {
        return this.getFamilienaam().compareTo(ander.getFamilienaam()) * 10
                + this.getVoornaam().compareTo(ander.getVoornaam());
    }
    …
}
```

```
package be.vdab;
import java.util.Set;
import java.util.TreeSet;

public class SorteerProgramma {
    public static void main(String[] args) {
        Set<Persoon> personen = new TreeSet<>();
        personen.add(new Persoon("Joe","Dalton"));
        personen.add(new Persoon("Sarah","Bernhardt"));
        for (Persoon persoon: personen) {
            System.out.println(persoon.getFamilienaam() + ' ' +
              persoon.getVoornaam());
        }
    }
}
```

Lokaal

```
package be.vdab;
import java.util.ArrayList;
import java.util.List;

class Traject {
    private final String naam;
    public Traject(String naam) {
        this.naam = naam;
    }
    public String getNaam() {
        return naam;
    }
}

class Cursist extends Persoon {
    private Traject traject;
    public Cursist(String voornaam, String familienaam, Traject traject) {
        super(voornaam, familienaam);
        this.traject = traject;
    }
    public Traject getTraject() {
        return traject;
    }
    public void setTraject(Traject traject) {
        this.traject = traject;
    }
}

class Trainer extends Persoon {
    public Trainer(String voornaam, String familienaam) {
        super(voornaam, familienaam);
    }
}

class Lokaal {
    private final int nummer;
    private Trainer trainer;
    private final List<Cursist> cursisten = new ArrayList<>();
    public Lokaal(int nummer, Trainer trainer) {
        this.nummer = nummer;
        this.trainer = trainer;
    }
    public void cursistToevoegen(Cursist cursist) {
        cursisten.add(cursist);
    }
```

```
    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append("Lokaal ").append(nummer)
          .append(' ').append(trainer.getNaam()).append("\n");
        builder.append(cursisten.size()).append(" cursisten\n");
        for (Cursist cursist : cursisten) {
            builder.append(cursist.getNaam())
           .append(" volgt ").append(cursist.getTraject().getNaam()).append("\n");
        }
        return builder.toString();
    }
}
```

Landen: eerste mogelijke oplossing

```
package be.vdab;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Scanner;
import java.util.TreeMap;

public class LandenProgramma {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Map<String, Integer> aantalInwonersPerLand = new TreeMap<>();
        String code = scanner.next();
        while (!code.equals("stop")) {
            int aantalInwoners = scanner.nextInt();
            aantalInwonersPerLand.put(code, aantalInwoners);
            code = scanner.next();
        }
        int totaal = 0;
        for (Entry<String, Integer> entry : aantalInwonersPerLand.entrySet()) {
            System.out.println(entry.getKey() + ':' + entry.getValue());
            totaal += entry.getValue();
        }
        System.out.println(totaal);
    }
}
```

Landen:tweede mogelijke oplossing

```
package be.vdab;
public class Land implements Comparable<Land>{
    private final String code;
    private int aantalInwoners;
    public Land(String code, int aantalInwoners) {
        this.code = code;
        this.aantalInwoners = aantalInwoners;
    }
    public String getCode() {
        return code;
    }
    public int getAantalInwoners() {
        return aantalInwoners;
    }
    @Override
    public boolean equals(Object object) {
        if (! (object instanceof Land)) {
            return false;
        }
        return code.equalsIgnoreCase(((Land) object).code);
    }
    @Override
    public int hashCode() {
        return code.toUpperCase().hashCode();
    }
    @Override
    public int compareTo(Land land) {
        return code.compareTo(land.code);
    }


}
package be.vdab;
import java.util.Scanner;
import java.util.Set;
import java.util.TreeSet;
public class LandenProgramma {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Set<Land> landen = new TreeSet<>();
        String code = scanner.next();
        while (!code.equals("stop")) {
            int aantalInwoners = scanner.nextInt();
            landen.add(new Land(code, aantalInwoners));
            code = scanner.next();
        }
        int totaal = 0;
        for (Land land : landen) {
            System.out.println(land.getCode() + ':' + land.getAantalInwoners());
            totaal += land.getAantalInwoners();
        }
        System.out.println(totaal);
    }
}
```

Pacman

```
package be.vdab;
import java.util.LinkedHashSet;
import java.util.Set;

abstract class Inwoner {
    protected Doolhof doolhof;
    public void setDoolhof(Doolhof doolhof) {
        this.doolhof = doolhof;
    }
}

class Monster extends Inwoner {
    private final String naam;
    public Monster(String naam) {
        this.naam = naam;
    }
    public String getNaam() {
        return naam;
    }
    public void meet(Pacman pacman) {
        pacman.meet(this);
    }
    @Override
    public boolean equals(Object object) {
      if (!(object instanceof Monster)) {
          return false;
      }
    return naam.equalsIgnoreCase(((Monster) object).naam);
    }
    @Override
    public int hashCode() {
        return naam.toUpperCase().hashCode();
    }
}

class Pacman extends Inwoner {
    private boolean powerPillEaten;
    private int levens = 3;
    public int getLevens() {
        return levens;
    }
    public void eatPowerPill() {
        powerPillEaten = true;
    }
    public void powerPillWorkedOut() {
        powerPillEaten = false;
    }
    public void meet(Monster monster) {
        if (powerPillEaten) {
            doolhof.remove(monster);
            monster.setDoolhof(null);
        } else {
            levens--;
        }
```

```
    }
}

class Doolhof {
    private Pacman pacman;
    private final Set<Monster> monsters = new LinkedHashSet<>();
    public void add(Pacman pacman) {
        this.pacman = pacman;
        pacman.setDoolhof(this);
    }
    public void add(Monster monster) {
        monsters.add(monster);
        monster.setDoolhof(this);
    }
    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append("pacman ").append(pacman.getLevens()).append(" levens, ");
        for (Monster monster : monsters) {
            builder.append(monster.getNaam()).append(", ");
        }
        if (!monsters.isEmpty()) {
            builder.deleteCharAt(builder.length() - 1);
            builder.deleteCharAt(builder.length() - 1);
        }
        return builder.toString();
    }
    public void remove(Monster monster) {
        monsters.remove(monster);
    }
}
```

Woorden 1

```
package be.vdab;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Scanner;
import java.util.TreeMap;

class WoordenProgramma {
    public static void main(String[] args) {
        Map<String, Integer> woordenEnAantallen = new TreeMap<>();
        Scanner scanner = new Scanner(System.in);
        for (String zin; !(zin = scanner.nextLine()).equals("stop");) {
            String[] woorden = zin.split(" ");
            for (String woord : woorden) {
                Integer aantal = woordenEnAantallen.get(woord);
                if (aantal == null) {
                    woordenEnAantallen.put(woord, 1);
                } else {
                    woordenEnAantallen.put(woord, aantal + 1);
                }
            }
        }
        for (Entry<String, Integer> entry : woordenEnAantallen.entrySet()) {
            System.out.println(entry.getKey() + ':' + entry.getValue());
        }
    }
}
```

Woorden 2

```
package be.vdab;
import java.util.LinkedHashSet;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Scanner;
import java.util.Set;
import java.util.TreeMap;

class WoordenProgramma {
  public static void main(String[] args) {
    Map<String, Set<Integer>> woordenEnAantallen = new TreeMap<>();
    Scanner scanner = new Scanner(System.in);
    int zinIndex = 0;
    for (String zin; !(zin = scanner.nextLine()).equals("stop");) {
      zinIndex++;
      String[] woorden = zin.split(" ");
      for (String woord : woorden) {
        Set<Integer> aantal = woordenEnAantallen.get(woord);
        if (aantal == null) {
          Set<Integer> value = new LinkedHashSet<>();
          value.add(zinIndex);
          woordenEnAantallen.put(woord, value);
        } else {
          aantal.add(zinIndex);
        }
      }
    }
    for (Entry<String, Set<Integer>> entry : woordenEnAantallen.entrySet()) {
      System.out.print(entry.getKey() + ':');
      for (int aantal : entry.getValue()) {
        System.out.print(aantal + " ");
      }
      System.out.println();
    }
  }
}
```

Kasbons

```
package be.vdab;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

abstract class Kasbon {
  protected final BigDecimal beginWaarde;
  protected final int jaren;
  protected final BigDecimal intrest;
  public Kasbon(BigDecimal waarde, int jaren, BigDecimal intrest) {
    this.beginWaarde = waarde;
    this.jaren = jaren;
    this.intrest = intrest;
  }
  public abstract BigDecimal getEindWaarde();
}

class NietKapitaliseerbareKasbon extends Kasbon {
  public NietKapitaliseerbareKasbon(BigDecimal beginWaarde, int jaren,
    BigDecimal intrest) {
    super(beginWaarde, jaren, intrest);
  }
  @Override
  public BigDecimal getEindWaarde() {
    return beginWaarde.add(
      beginWaarde.multiply(intrest).multiply(BigDecimal.valueOf(jaren)));
  }
}

class KapitaliseerbareKasbon extends Kasbon {
  public KapitaliseerbareKasbon(BigDecimal beginwaarde,int jaren,
    BigDecimal intrest) {
    super(beginWaarde, jaren, intrest);
  }
  @Override
  public BigDecimal getEindWaarde() {
    BigDecimal totaleIntrest = BigDecimal.ZERO;
    for (int jaar = 0; jaar != jaren; jaar++) {
      totaleIntrest = totaleIntrest.add
        (beginWaarde.multiply(intrest)).add(totaleIntrest.multiply(intrest));
    }
    return beginWaarde.add(totaleIntrest);
  }
}

class Klant {
  private final List<Kasbon> kasbons = new ArrayList<>();
  public void add(Kasbon kasbon) {
    kasbons.add(kasbon);
  }
```

```
  public BigDecimal getEindWaarde() {
    BigDecimal totaal = BigDecimal.ZERO;
    for (Kasbon kasbon : kasbons) {
      totaal = totaal.add(kasbon.getEindWaarde());
    }
    return totaal;
  }
}
```

Time

```
package be.vdab;
public class Time {
  private int hours;
  private int minutes;
  public Time(int hours, int minutes) {
    if (hours < 0) {
      throw new IllegalArgumentException();
    }
    if (minutes < 0 || minutes > 59) {
     throw new IllegalArgumentException();
    }
    this.hours = hours;
    this.minutes = minutes;
  }
  public Time(int hours) {
    this(hours, 0);
  }
  public Time subtract(Time hour2) {
    int hoursResult = this.hours - hour2.hours;
    int minutes = this.minutes - hour2.minutes;
    if (minutes < 0) {
      hoursResult--;
      minutes += 60;
    }
    return new Time(hoursResult, minutes);
  }
  public Time add(Time hour2) {
    int hoursResult = this.hours + hour2.hours;
    int minutesResult = this.minutes + hour2.minutes;
    if (minutesResult > 59) {
      hoursResult++;
      minutesResult -= 60;
    }
    return new Time(hoursResult, minutesResult);
  }
  public String toString() {
    return hours + ":" + minutes;
  }
}
```