25 punten

OPGAVE 1

De onderdelen a en b van deze opgave gaan over de klassen Baas en Hond. De definitie van deze klassen volgt hieronder. Het ontwerp is overigens geen afspiegeling van de werkelijkheid.

```
public class Baas {
  private Hond hond;
  private String naam;

  public Baas(String naam, String naamHond) {
    this.naam = naam;
    this.hond = new Hond(naamHond);
    hond.setBaas(this);
  }

  public Hond getHond() {
    return hond;
  }
}

public class Hond {
  private Baas baas;
  private String naam;

  public Hond(String naam) {
    this.naam = naam;
  }

  public void setBaas(Baas baas) {
    this.baas = baas;
  }

  public String getNaam() {
    return naam;
  }

  public Baas getBaas() {
    return baas;
  }
}
```

Het onderstaande fragment wordt uitgevoerd.

```
Baas jan = new Baas("Jan", "Bello");
Baas piet = new Baas("Piet", "Bello");
Hond lassie = new Hond("Lassie");
```

1a Teken het toestandsdiagram na uitvoering van dit fragment. Strings moeten als aparte objecten worden opgenomen in het diagram.

Vervolgens wordt (na uitvoering van het bovenstaande fragment) het onderstaande fragment uitgevoerd.

```
Hond hond1 = jan.getHond();
Hond hond2 = piet.getHond();
Baas baas1 = hond1.getBaas();
hond1.setBaas(piet);
```

1b Geef de waarden van de onderstaande zeven uitdrukkingen na uitvoering van beide fragmenten. Gemakshalve zijn de uitdrukkingen genummerd. U hoeft uw antwoorden niet te motiveren.

```
(1)  lassie.getBaas()
(2)  hond1 == hond2
(3)  hond1.getBaas() == baas1
(4)  baas1 == piet
(5)  piet == hond1.getBaas()
(6)  piet.getHond() == hond1
(7)  jan.getHond() == hond1
```

Gegeven is het onderstaande fragment:

```
int i = 0;
int j = 0;
while (j < 4) {
  if (i % 2 == 0)
    j = j + 2 * i;
  else
    j = j - i;
  i++;
}
```

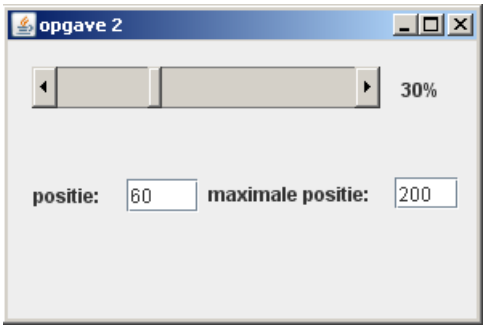
1c Wat is de waarde van i en j na verwerking van dit fragment?
U hoeft uw antwoord niet te motiveren.

25 punten

OPGAVE 2

In deze opgave wordt gebruik gemaakt van de klasse Scrollbar uit package java.awt.Scrollbar. Een deel van de interface van deze klasse staat in de bijlage na de tentamenopgaven.

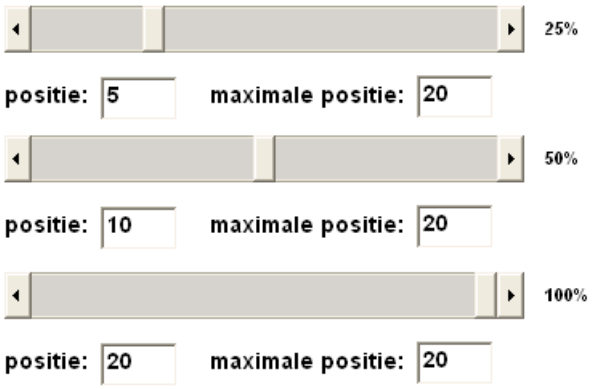
In onderstaande figuur ziet u een applicatie met een scrollbar (attribuut scrollbar), een label met daarin een percentage (attribuut percentageLabel) en twee invoervelden (attributen positieVeld en maxPositieVeld).



De waarde die de scrollbar representeert, wordt bepaald door de positie van het schuifblokje van de scrollbar. In bovenstaand voorbeeld kan het schuifblokje 201 verschillende posities hebben (0,1,2, ..., 200). De actuele positie van het schuifblokje is 60 en dat is tevens de waarde van het positieVeld. De positie van het schuifblokje verandert door het schuifblokje met de muis te verschuiven, door met de muis te klikken op een van de pijltjes of door de gewenste positie in te voeren in het eerste tekstveld. In dit tekstveld wordt tevens de actuele positie van het schuifblokje getoond, zodat de actuele positie van het schuifblokje voortdurend correspondeert met de inhoud van het tekstveld. Als de positie van het schuifblokje wijzigt, verandert ook het percentage achter de scrollbar dat de relatieve positie van het schuifblokje ten opzichte van de maximaal mogelijke positie weergeeft.

In het tweede invoerveld kan de gebruiker invoeren wat de maximaal mogelijke positie van het schuifblokje is.

Hieronder volgen ter verduidelijking nog drie toestanden van de applicatie.



De gui van de applicatie bevat drie event handlers: Eén voor invoer van de gewenste positie, één voor invoer van de maximale positie en één voor als de positie van het

schuifblokje verandert. Verder bevat de gui een hulpmethode voor het berekenen en tonen van het percentage.

De specificaties van de event handlers en van de hulpmethode zijn als volgt:

```
/**
 *  Berekent en toont het percentage in percentageLabel
 */
private void toonPercentage()

/**
 *  Toont de positie van het schuifblokje in positieVeld
 *  en werkt het percentage bij
 */
private void scrollbarAction()

/**
 * Zet de actuele positie van het schuifblokje gelijk aan de
 * waarde in positieVeld en werkt het percentage bij
 */
private void positieVeldAction()

/**
 * Zet de maximaal mogelijke positie van het schuifblokje
 * gelijk aan de waarde in maxPositieVeld en werkt het
 * percentage bij
 */
private void maxPositieVeldAction()
```

Opmerking: In de documentatie van de klasse Scrollbar wordt regelmatig gesproken over de waarde van de scrollbar. Hiermee wordt de positie van het schuifblokje bedoeld.

2a De scrollbar is standaard verticaal georiënteerd. Welke regel moet aan de programmacode worden toegevoegd om te zorgen dat de scrollbar horizontaal georiënteerd wordt?

2b Geef een implementatie van toonPercentage().

2c Geef een implementatie van de event handler positieVeldAction().

2d Heeft u in uw antwoord op een van de vragen a, b of c gebruik gemaakt van een klassenmethode? Zo ja, van welke klasse? Geef de bijbehorende regel code.

25 punten

OPGAVE 3

In deze opgave ontwerpen we een klasse Routeplanner voor het bepalen van de kortste route tussen een beperkt aantal locaties. Het moeilijkste aspect daarvan, namelijk de implementatie van een algoritme om deze route te berekenen, hoeft u niet te maken.

Klasse Routeplanner gebruikt onderstaande hulpklasse Verbinding. Een Verbinding verbindt twee locaties met elkaar.

```
public class Verbinding {
  private String van;
  private String naar;
  private int afstand;

  public Verbinding(String van, String naar, int afstand){
    this.van = van;
    this.naar = naar;
    this.afstand = afstand;
  }

  public String getVan(){
    return van;
  }

  public String getNaar(){
    return naar;
  }

  public int getAfstand(){
    return afstand;
  }
}
```

Klasse Routeplanner bevat een ArrayList van Verbindingen en het maximaal aantal verbindingen. Naast een constructor waarmee het maximum aantal verbindingen wordt vastgelegd, beschikt deze klasse over een aantal methoden:
1. Een methode om een Verbinding toe te voegen. Dat lukt alleen wanneer het maximum aantal verbindingen nog niet is bereikt.
2. Een methode die, gegeven de namen van twee locaties, de kortste route daartussen berekent en deze retourneert als een ArrayList van Verbindingen. Attribuut 'naar' van een element van de geretourneerde ArrayList is steeds gelijk aan het attribuut 'van' van het daaropvolgende element. Indien er geen route berekend kan worden (omdat een locatie onbereikbaar is of onbekend) wordt null geretourneerd. (Deze methode hoeft u niet te implementeren).
3. Een methode die gegeven een route (ArrayList van Verbindingen) een ArrayList van Strings retourneert met namen van te passeren locaties in de goede volgorde.
4. Een methode die gegeven een route (ArrayList van Verbindingen) het aantal af te leggen kilometers geeft.

3a Geef een gedetailleerd klassendiagram voor het domeinmodel. Neem daarin ook de constructoren op.

3b Geef een implementatie van methode genummerd 1 in de probleembeschrijving.

3c Geef een implementatie van methode genummerd 3 in de probleembeschrijving.

25 punten

OPGAVE 4

Met behulp van onderstaande klassen Wisselpaar en Codeer is het mogelijk teksten te coderen en gecodeerde teksten weer te decoderen.

```
public class Wisselpaar {
  private char kar1;
  private char kar2;

  /**
   * Creeert een nieuwe Wisselpaar object.
   * @param kar1 het ene karakter
   * @param kar2 het andere karakter
   */
  public Wisselpaar(char kar1, char kar2) {
    this.kar1 = kar1;
    this.kar2 = kar2;
  }

  /**
   * Bepaalt of een karakter k voorkomt in dit Wisselpaar.
   * @param k het karakter
   * @return true als k voorkomt in dit Wisselpaar, anders false
   */
  public boolean bevat(char k) {
    return k == kar1 || k == kar2;
  }

  /**
   * Bepaalt de vervanger van een karakter k, zijnde:
   * kar2 als k gelijk is aan kar1
   * kar1 als k gelijk is aan kar2
   * k zelf als k niet voorkomt in dit Wisselpaar.
   * @param k het karakter
   * @return de vervanger van k
   */
  public char vervanger(char k) {
    if (k == kar1)
      return kar2;
    if (k == kar2)
      return kar1;
    return k;
  }
}


public class Codeer {
  private ArrayList<Wisselpaar> lijst;
  private int maxLengte;

  /**
   * Creeert een nieuw Codeer object.
   * @param maxLengte de maximale lengte van lijst
   */
  public Codeer(int maxLengte) {
    lijst = new ArrayList<>();
    this.maxLengte = maxLengte;
  }

  /**
   * Voegt een nieuw Wisselpaar toe aan de lijst mits
   * de betreffende karakters nog niet voorkomen in
   * een van de Wisselparen in de lijst en mits
```

```
     * er nog ruimte in de lijst is.
     * @param kar1 het eerste karakter van het Wisselpaar
     * @param kar2 het tweede karakter van het Wisselpaar
     */
    public void voegToe(char kar1, char kar2) {
        ... implementeren in opgave 4a
    }

    /**
     * Bepaalt de gecodeerde waarde van een karakter k:
     * als een Wisselpaar van de lijst het karakter k bevat
     * wordt diens vervanger geretourneerd,
     * als geen van de Wisselparen het karakter k bevat
     * wordt k zelf geretourneerd
     * @param k het karakter
     * @return de gecodeerde waarde van k
     */
    public char gecodeerd(char k) {
        ... implementeren in opgave 4b
    }

    /**
     * bepaalt de gecodeeerde waarde van een string s door van
     * elk karakter van s de gecodeerde waarde te bepalen
     * @param s de string
     * @return de gecodeerde waarde van s
     */
    public String gecodeerd(String s) {
        ... implementeren in opgave 4c
    }
}
```

Het coderingsproces werkt door karakters onderling te verwisselen. Welke karakters verwisseld moeten worden wordt bijgehouden in een lijst met zogenoemde Wisselparen. Elk Wisselpaar bevat twee karakters die onderling verwisseld moeten worden.

Een voorbeeld van een toepassing van een Codeer-object:

```
Codeer codeer = new Codeer (10);
codeer.voegToe('a', 'e'); 'a' en 'e' onderling verwisselen
codeer.voegToe('p', 'k'); 'p' en 'n' onderling verwisselen
codeer.voegToe('k', 'm'); //genegeerd want 'k' komt al voor
char c = codeer.gecodeerd('e'); // c is nu 'a'
String s1 = codeer.gecodeerd("pek"); // s1 is nu "kap"
String s2 = codeer.gecodeerd(s1); // s2 is weer "pek"
```

Bij de uitwerking van opgave 4c kunt u gebruik maken van de methode charAt uit de klasse String die als volgt is gespecificeerd:

public char `charAt(int index)`
Returns the character at the specified index. An index ranges from 0 to length() - 1. The first character of the sequence is at index 0, the next at index 1, and so on.

4a Geef een implementatie van de methode `voegToe(char kar1, char kar2)`

4b Geef een implementatie van de methode `gecodeerd(char k)`

4c Geef een implementatie van de methode `gecodeerd(String s)`

Bijlage bij opgave 2

Klasse Scrollbar

Field Summary

`static int`	**`HORIZONTAL`** A constant that indicates a horizontal scroll bar.
`static int`	**`VERTICAL`** A constant that indicates a vertical scroll bar.

Constructor Summary

`Scrollbar``()` Constructs a new vertical scroll bar.

Method Summary

`int`	**`getMaximum`**`()` Gets the maximum value of this scroll bar.
`int`	**`getMinimum`**`()` Gets the minimum value of this scroll bar.
`int`	**`getValue`**`()` Gets the current value of this scroll bar.
`void`	**`setMaximum`**`(int newMaximum)` Sets the maximum value of this scroll bar.
`void`	**`setMinimum`**`(int newMinimum)` Sets the minimum value of this scroll bar.
`void`	**`setOrientation`**`(int orientation)` Sets the orientation for this scroll bar.
`void`	**`setValue`**`(int newValue)` Sets the value of this scroll bar to the specified value.