1. Leg uit wat het verschil is tussen een klasse en een object

2. Leg uit wat bij object georiënteerd programmeren inkapseling inhoudt. Geef daarnaast een voorbeeld.

3. Leg uit wat bij object georiënteerd programmeren overerving inhoudt. Geef daarnaast een voorbeeld.

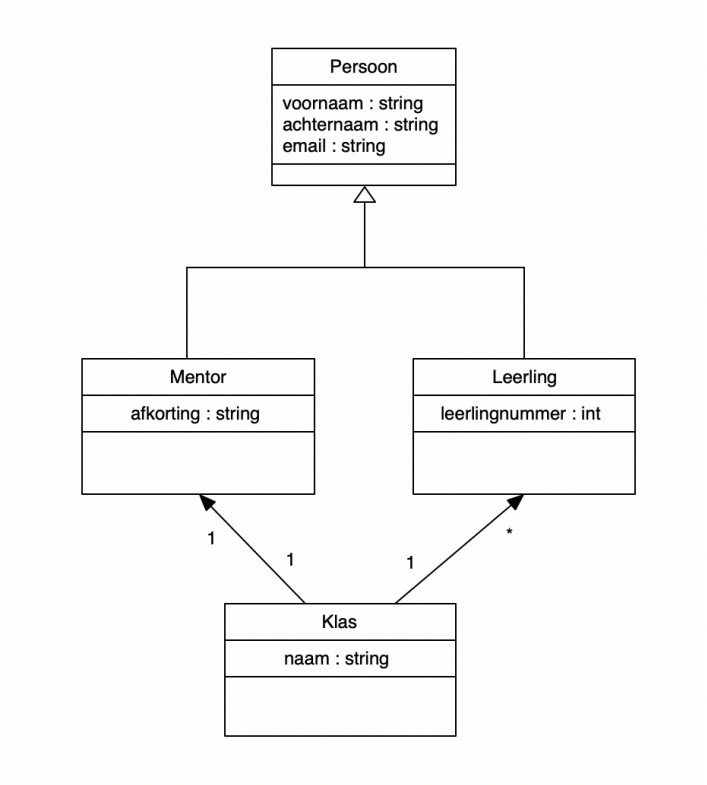
4. Hieronder staat de code voor een JavaScript klasse waarin gegevens voor een persoon kunnen worden opgeslagen. Alle attributen zijn nu public. Welke attribu(u)t(en) zou(den) beter private kunnen zijn? Leg per attribuut uit waarom.

```
class Persoon {
  voornaam;
  achternaam;
  geboortedatum;
  leeftijd;

  constructor(_voornaam, _achternaam, _geboortedatum) {
    this.voornaam = _voornaam;
    this.achternaam = _achternaam;
    this.geboortedatum = _geboortedatum;

    // neem aan dat de functie jarenTotNu() het verschil
    // in hele jaren geeft tussen de meegegeven datum en nu
    this.leeftijd = jarenTotNu(geboortedatum);
  }
}
```

5. Hieronder zie je de code van meerdere C++ klassen. Teken het bijbehorende klassendiagram

```
class Thermostaat {
  private:
    TemperatuurSensor tempSensor;
    LCD display;
    int serienummer;
    bool verwarmen;

    void update() {
      // deze methode leest de sensor uit en werkt het display
      // en het attribuut ‘verwarmen’ bij. Exacte invulling onbelangrijk.
    }

    int readSerialNumber() {
      // deze methode leest het unieke serienummer van het apparaat uit.
      // Exacte invulling onbelangrijk.
    }

  public:
    Thermostaat() {
      this.serienummer = this.readSerialNumber();
      this.tempSensor = new TemperatuurSensor();
      this.display = new LCD();
      this.verwarmen = false;
    }

    int getSerienummer() {
      return this.serienummer;
    }

    bool getVerwarmen() {
      return this.verwarmen;
    }
}

class TemperatuurSensor {
  public:
    TemperatuurSensor() {
      // initialiseer de LCD voor gebruik
      // exacte invulling onbelangrijk.
    }

    int geefTemp() {
      // leest de temperatuur en geeft deze terug
    }
}

class LCD {
  public:
    LCD() {
      // initialiseer de LCD voor gebruik
      // exacte invulling onbelangrijk.
    }

    toonTekst(String tekst) {
      // code om de tekst op het display te tonen.
      // exacte invulling onbelangrijk.
    }
}
```

6. Hieronder zie je een klassendiagram. Maak de bijbehorende JavaScript-programmeercode.



7. Hieronder zie je JavaScript-code van een ‘spel’ waarin twee ballen tegen de wanden stuiteren. Teken een objectdiagram voor beide ballen voor de toestand waarin ze verkeren nadat de draw-methode tweemaal is uitgevoerd.

```
class Game {
  ballen;
  #width;
  #height;

  constructor() {
    this.#width = 800;
    this.#height = 600;
    this.ballen = [new Bal (40, 40, 2, 5), new Bal (795, 20, 2, 10)];
  }

  // draw wordt telkens opnieuw uitgevoerd
  draw() {
    for(var i = 0; i < this.ballen.length(); i++) {
      var bal = this.ballen[i];

      bal.x = bal.x + bal.speedX;
      bal.y = bal.y + bal.speedY;

      if (bal.x <= 0 || bal.x >= bal.width) {
        bal.speedX = bal.speedX * -1;
      }

      if (bal.y <= 0 || bal.y >= bal.height) {
        bal.speedY = bal.speedY * -1;
      }

      bal.show();
    }
  }
}

class Bal {
  x;
  y;
  speedX;
  speedY;

  constructor (_x, _y, _speedX, _speedY) {
    this.x = _x;
    this.y = _y;
    this.speedX = _speedX;
    this.speedY = _speedY;
  }

  show() {
    fill(255, 0, 0);
    ellipse(this.x, this.y, 20, 20);
  }
}
```