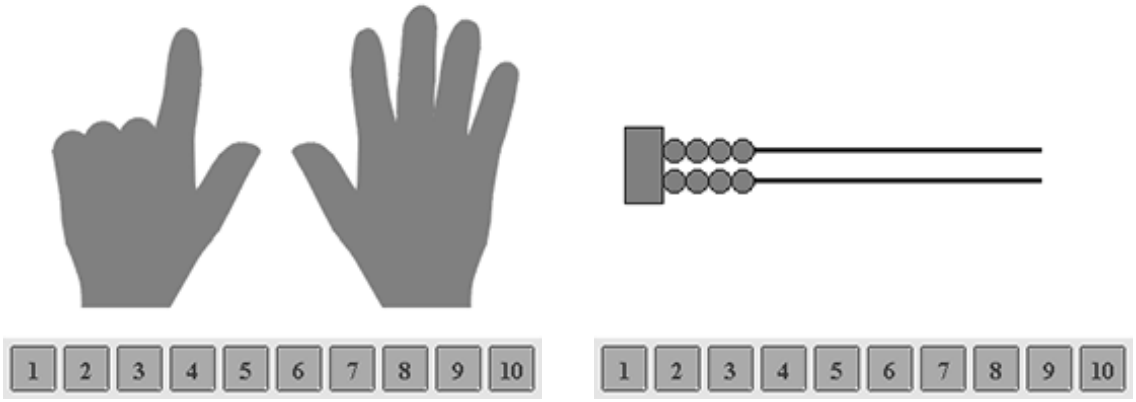
OOPD Oefentoets 2

Opgave 1: Arrays bij elkaar optellen

Maak de publieke methode telElementenOp die twee integer arrays met dezelfde lengte als invoer heeft. De methode telt de getallen van elk element uit beide arrays bij elkaar op en slaat de resultaten op in een nieuwe array. Vervolgens retourneert de methode de nieuwe array.
Schrijf de methode in javacode

Opgave 2: Flitsbeelden

Gegeven de definitie van het Flitsbeeldspel. In dit spel krijgen leerlingen kort een beeld te zien waarin ze het aantal getoonde vingers, kralen op een telraam, eieren, etc moeten aanklikken in de balk eronder. Hieronder zie je enkele screenshots uit het (daadwerkelijke ontwikkelde) spelletje.



In deze opgave wordt ervan uitgegaan dat het spel getallen tussen de 1 en 10 genereert Hieronder is de klasse gegeven.

```
public class FlitsbeeldSpel
{
    private String beeld = "Handbeeld";

    //Code weggelaten

    public void speelSpel()
    {
        int aantal = maakGetal(1, 10);
        toonBeeld(aantal);

        // controleer antwoord het aangeklikte antwoord,
        // niet van belang in deze opgave
    }

    public void toonBeeld(int n)
    {
        if (beeld == "Handbeeld")
```

```
        {
            // toon beeld van handen
        }
        else if (beeld == "Telraambeeld")
        {
            // toon beeld van telraam
        }
    }

     /**
     * Retourneert een willekeurig geheel getal dat tussen
     * de meegegeven paramaters ligt
     */
    private int maakGetal(int laagste, int hoogste)
    {
        //Code niet van belang voor deze opgave
    }
}
```

Hoewel deze klasse werkt, is het toevoegen van een nieuw flitsbeeld te veel gedoe, bijvoorbeeld als je een nieuw flitsbeeld wilt toevoegen, zoals een eierdoos.

Daarom moet de code uit deze klasse opnieuw verdeeld worden over andere klassen en het if-statement weggewerkt worden.

Opgave A

Schrijf javacode voor de drie klassen die nodig zijn om met behulp van polymorfie het if-statement weg te kunnen werken uit de methode `toonBeeld`.
NB: je hoeft geen constructoren te schrijven. Ook de body's van de methoden hoef je niet te schrijven, maar plaats de comments uit bovenstaande code wel op de juiste plaats. Verder hoef je geen klasse te maken voor het flitsbeeld met de eierdoos.

Opgave B

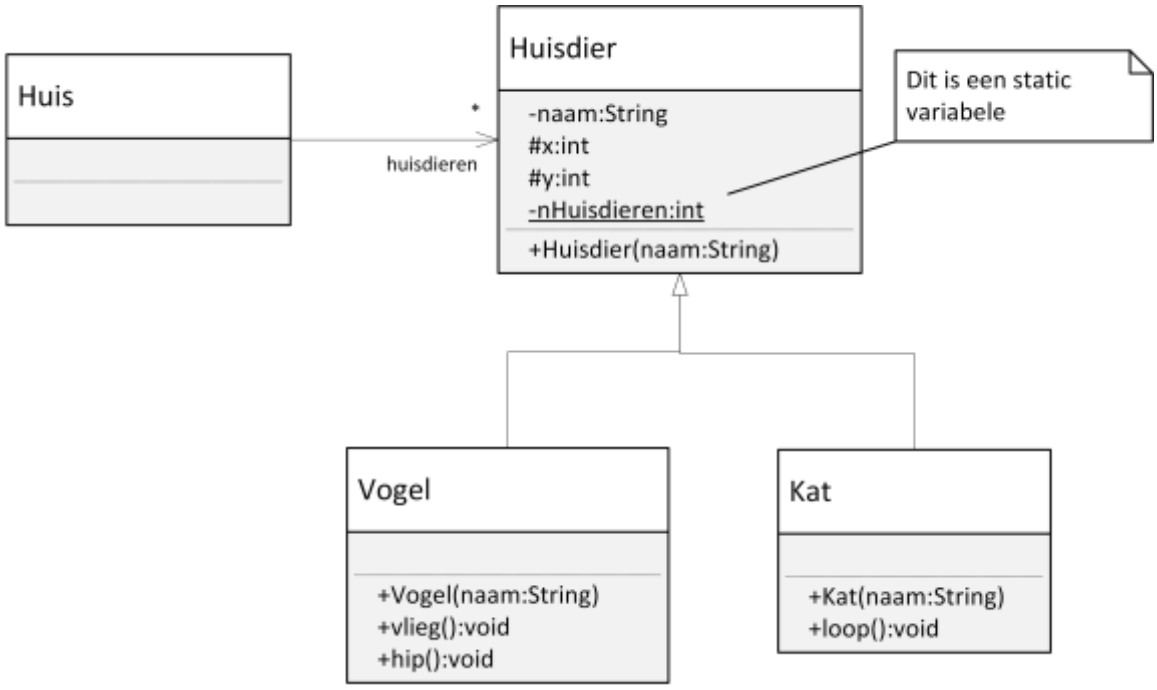
Vervang de declaratie en initialisatie van de instantievariabele `beeld` uit de klasse `FlitsbeeldSpel` door een soortgelijke regel, waarin je een van de nieuwe objecten aanmaakt. Ga ervan uit dat je Handbeelden wilt tonen.

Opgave C

Pas de methode `speelSpel` uit de klasse `FlitsbeeldSpel` aan, zodat dit flitsbeelden kan tonen, ongeacht het type flitsbeeld dat geselecteerd is. Geef javacode van de complete nieuwe methode, het commentaar hoef je niet op te nemen in de uitwerking.

Opgave 3: Klassendiagram huisdieren

Gegeven onderstaan klassendiagram



Opgave A

Geef de complete implementatie van dit klassendiagram in javacode.

Opgave B

De variabele `nHuisdieren` is een static variabele. Schrijf zo duidelijke mogelijk op wat de consequentie is van deze beslissing.

Opgave 4: Huisdierenlijst

Gegeven onderstaande main-methode

```
public class Main {
    public static void main(String[] args) {
        Huisdier[] huisdieren = {
            new Kat("Poekie"),
            new Vogel("Lorre")
        };

        for (Huisdier h : huisdieren) {
          ...
        }
    }
}
```

Opgave A

In de for each lus moet elk huisdier bewegen volgens onderstaande regels:

- als `h` een kat is, dan moet de methode `loop` uitgevoerd worden
- als `h` een vogel is, dan moeten de methoden `hip` en `vlieg` uitgevoerd worden

Schrijf javacode binnen de for-each lus om dit voor elkaar te krijgen.

Opgave B

Nu willen we de alle huisdieren graag laten bewegen in de for-each lus zonder dat we daarbinnen hoeven te controleren of `h` een kat, of een vogel is.

Pas de klassen `Huisdier`, `Kat` en `Vogel` zo aan dat je in de for-each lus alle huisdieren kunt laten bewegen volgens het gedrag dat bij opgave A is gespecificeerd. Gebruik polymorfie.

Geef javacode van de aanpassingen aan de klassen en de nieuwe for-each lus.

Opgave 5: Geheugenmodel huisdieren

Gegeven onderstaande (bekende) klassen

```
public abstract class Huisdier
{
    private String naam;
    protected int x, y;

    public Huisdier(String naam)
    {
        this.naam = naam;
        x = 0;
        y = 0;
    }

    public abstract void beweeg();
}
```

```
public class Kat extends Huisdier
{
    public Kat(String naam)
    {
        super(naam);
    }

    public void beweeg()
    {
        loop();
    }

    public void loop()
    {
        x += 10;
    }
}
```

```
public class Vogel extends Huisdier {
    public Vogel(String naam) {
        super(naam);
    }

    public void beweeg() {
        hip();
        vlieg();
    }

    public void hip() {
        x += 2;
    }

    public void vlieg() {
       y += 10;
    }
}
```

Klasse met main methode

```
01  public class Huis {
02      public static void main(String[] args) {
03          Huisdier[] huisdierenlijst = {
04              new Kat("Poekie"),
05              new Vogel("Lorre")
06          };
07
08          for (int i = 0; i < huisdierenlijst.length; i++) {
09              huisdierenlijst[i].beweeg();
10          }
11      }
12  }
```

Opgave

Teken het geheugenmodel op het moment dat het programma zich voor de tweede keer in de for-lus (regels 08 – 10) bevindt (de variabele `i` heeft waarde 1) en de methode `beweeg` net is uitgevoerd, maar nog niet van de stack is verwijderd.