In BlueJ

Doe onderstaande met muis/menu's:

- Download en unzip het bestand `sklok-run.zip` en open het in BlueJ
- Maak een `Schaakklok`-object
- Voer `herstart()` uit voor dit object
- Vraag de resterende tijd op voor spelers 1 en 2
- Roep `ontvangPuls()` 2 keer op
- Vraag opnieuw de tijd op voor spelers 1 en 2
- wisselSpeler van speler en stuur één puls
- Vraag opnieuw de tijd op voor spelers 1 en 2

Doe nu hetzelfde opnieuw met het terminalvenster open.

Herstart virtuele machine en herhaal dit nu vanuit het evaluatievak:

```
new Schaakklok()    // en geef dit object de naam 'klok'
klok.herstart();    // opgelet: kommapunt
klok.geefTijd(1)    // GEEN kommapunt
klok.geefTijd(2)
klok.ontvangPuls();
klok.wisselSpeler();
```

De klasse `Cijferpaar`

- Maak een cijferpaar-object aan.
- Stel de waarde van dit cijferpaar in op 11 m.b.v. `zetWaarde`.
- Roep `verminderMet1` een aantal keer op (in GUI én in evaluatievenster)
- Bekijk de 'waarde' en de 'tijd'. Wat is het verschil?
- Doe hetzelfde met een tweede cijferpaar-object.
- Beïnvloeden beide objecten elkaar?

Inspecteren

- Inspecteer een schaakklokobject. Wat is de inhoud van het `speler`-veld?
- Roep `wisselSpeler` op voor dit object
- Wat is nu de inhoud van het `speler`-veld?

En nu: zelf programmeren!

- Maak een nieuw project aan in BlueJ
- Maak daar de klasse `Cijferpaar` aan
- Veeg het meeste uit van wat er daar reeds staat
- Volg de Java spiekbrief en de 'implementaties' op blz. 13–15.

(**Opgelet!** We gebruiken een andere volgorde dan in de nota's.)

De klasse Cijferpaar


```
public class Cijferpaar {

   private int teller;

   // constructor
   public Cijferpaar () { ... }

   // procedures en functies
   public void verminderMet1() {
       ...
   }
   public void zetWaarde(int w) { ... }

   public int geefWaarde() { ... }
   public String geefTijd() { ... }
}
```

De klasse Cijferpaar — procedures

```
public class Cijferpaar {

   private int teller;

   public void verminderMet1() {
       teller = teller - 1;
   }

   public void zetWaarde (int w) {
       teller = w;
   }
   ...
}
```

De klasse Cijferpaar — functies

```
public class Cijferpaar {
    ...

    public int geefWaarde() {
        return teller;
    }

    public String geefTijd() {
        if (waarde > 9) {
            return "" + teller; // trucje
        } else {
            return "0" + teller;
        }
    }

}
```

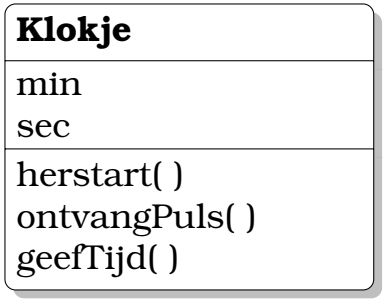
De klasse Cijferpaar — constructor

```
public class Cijferpaar {

   private int teller;

   public Cijferpaar () {
       teller = 0;
   }
   ...
}
```

De klasse Klokje




```
public class Klokje {

    private Cijferpaar min;
    private Cijferpaar sec;

    public Klokje () {
        min = new Cijferpaar ();
        sec = new Cijferpaar ();
    }

    public void herstart() {
        min.zetWaarde(6);
        sec.zetWaarde(0);
    }
    ...
}
```

De klasse Klokje — vervolg

```
public String geefTijd() {
    return min.geefTijd() + ":" + sec.geefTijd();
}

public void ontvangPuls() {
    if (sec.geefWaarde() == 0) {
        sec.zetWaarde(59);
        min.verminderMet1 ();
    } else {
        sec.verminderMet1 ();
    }
}
```

De klasse Schaakklok


```
public class Schaakklok {

    private int speler; //1: wit, 2: zwart

    private Klokje wit;
    private Klokje zwart;

    public Schaakklok() {
        wit = new Klokje ();
        zwart = new Klokje ();
        speler = 1;
    }

    ...
}
```

De klasse Schaakklok — alternatief

In plaats van de spelers aan te duiden met een gehele waarde `speler` kan je ook een logische waarde `isWit` gebruiken, met waarden `true` of `false`.

```
public class Schaakklok {

    private boolean isWit; // true of false
    ...
    public String geefTijd (boolean w) {
        ...
    }
}
```

Opmerking Ook `Cijferpaar.geefWaarde` vervangen door `Cijferpaar.isNul`

Constructoren met parameters

- Laat toe om van in het begin onderscheid te maken tussen seconden- en minutencijferparen
- Hebben elk een andere `herstartWaarde` die wordt meegegeven bij aanmaken

```
min = new Cijferpaar (6);
sec = new Cijferpaar (0);
```

- Constructor:

```
private int herstartWaarde;
...
public Cijferpaar (int hw) {
    herstartWaarde = hw;
}
```

- `Cijferpaar` kan nu een eigen `herstart` hebben.

```
public void herstart() {
    teller = herstartWaarde;
}
```

- En `zetWaarde` kan volledig overbodig gemaakt worden.

Notatieconventies

```
public Cijferpaar (int herstartWaarde) {
    this.herstartWaarde = herstartWaarde;
}
```