# Kisten

Opdat een *kistenvuller* zijn werk zou kunnen doen, hoeven kisten slechts twee methoden te ondersteunen:

- Kijk of een bepaald bord in deze kist kan worden toegelaten

```
public boolean isAanvaardbaar (Bord bord) {
    ...
}
```

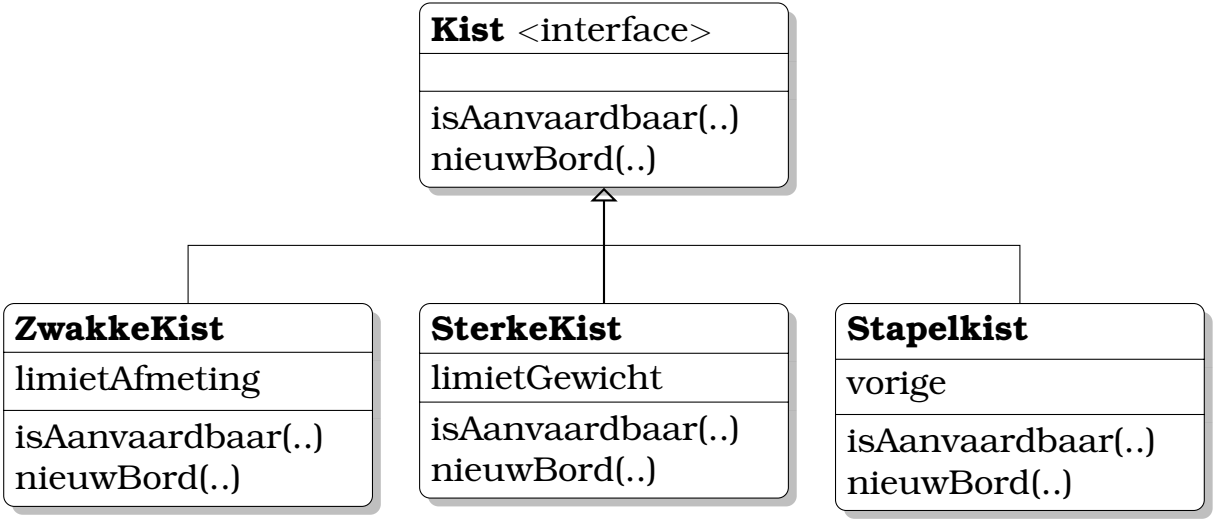
- Stop een (toegelaten) bord in de kist

```
public void nieuwBord (Bord bord) {
    ...
}
```

Dit vormt een soort *contract* waaraan een kist moet voldoen.

# Interfaces — begrip

In Java wordt een ‘contract’ vastgelegd in een *interface*.




Een interface lijkt goed op een gemeenschappelijke bovenklasse, maar dan zonder implementatie van de methoden (en met een lichtjes gewijzigde notatie).

# Interfaces — notatie

Een *interface* legt een contract vast waaraan een klasse moet voldoen.

```
public interface Kist {

    public boolean isAanvaardbaar (Bord bord);

    public void nieuwBord (Bord bord);
}
```

Aangeven dat een klasse een interface implementeert:

```
public class SterkeKist implements Kist {
  ...
}
```

Compiler controleert dat alle methoden van de interface wel degelijk een definitie hebben (zijn geïmplementeerd).

**Opgave** Download vereenvoudigde versie (v1) en schrijf klasse `SterkeKist` em methode `vulKisten` in `Kistenvuller`

# Even recapituleren. . .

**Overerving**

- Een klasse kan velden en methoden *erven* van een andere klasse, zodat je gemeenschappelijke code voor beide klassen maar één keer hoeft te schrijven.

**Polymorfisme**

- Je hoeft van een klasse enkel te weten aan welk 'contract' (= *interface*) ze voldoet om ze te kunnen gebruiken.
- Verschillende *implementaties* van dezelfde interface kunnen samen gebruikt worden: bijvoorbeeld in dezelfde array gestopt worden, in dezelfde lijst, in dezelfde variabele. . .

Polymorfisme is automatisch bij overerving, maar bestaat ook zonder.

```
List<Persoon> list = new ArrayList<Persoon> ();
```

`List` is geen klasse maar een *interface*. De klasse `ArrayList` is een *implementatie* van `List`.

# Opgave

Werk project *kisten* verder af.

- Implementeer de klasse `ZwakkeKist`
- Implementeer de klasse `Stapelkist`
- Test het programma

# Interfaces in de praktijk

Bij projecten die in verschillende delen worden gesplitst, worden interfaces gebruikt om te definiëren hoe de delen moeten interageren.

```
public interface Schaakklok {
    public void wisselSpeler ();

    public void herstart ();

    public void ontvangPuls ();

    public String geefTijd (int speler);
}
```

Naast de uiteindelijke implementatie wordt dan vaak een *nep*-implementatie gemaakt (een prototype) van dezelfde interface.

*Frameworks* (voor GUIs, webapplicaties, . . . ) eisen dat *plug-ins* aan een bepaalde interface voldoen.