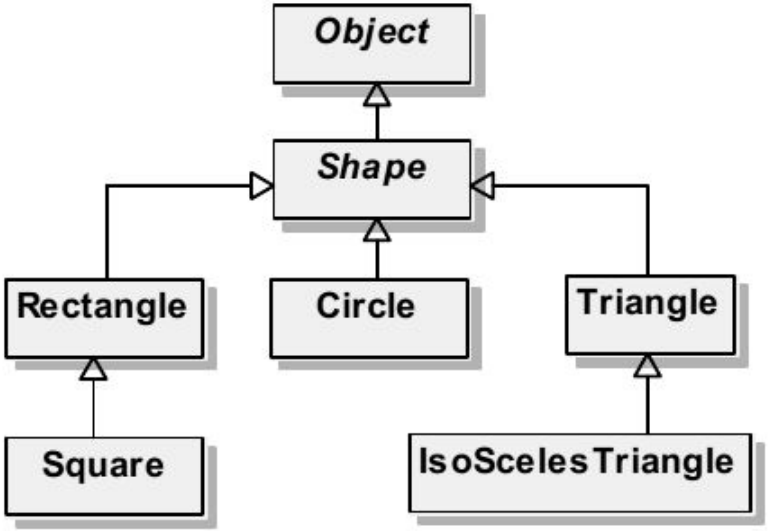
De superklasse Object

Mother of all objects

Klasse-hiërarchie



Iedere klasse is in Java rechtstreeks of onrechtstreeks een subklasse van de **object klasse**.

De operator instanceof

- Met de **operator** instanceof kan men nagaan of een object een instantie is van één of andere (super)klasse of interface.

➔ Op voorwaarde dat het object zich in dezelfde hiërarchische tak bevindt als de klasse of interface

- instanceof geeft een boolean terug.

```
Rectangle r = new Rectangle();
System.out.println(r instanceof Object);    // true
System.out.println(r instanceof Shape);     // true
System.out.println(r instanceof Rectangle); // true
System.out.println(r instanceof Square);    // false
System.out.println(r instanceof Circle);    // Compilation error
```

Methoden van de Object-klasse

Returntype	Methode	Omschrijving
String	toString()	Geeft een string terug waarin het object beschreven wordt
boolean	equals(Object o)	Geeft aan of twee objecten gelijk zijn. Geeft **true** als de referentievariabelen gelijk zijn
int	hashcode()	
Class	getClass()	Geeft een object terug dat de concrete klasse voorstelt
void	finalize()	Garbage collection

toString( )

- Elk object erft deze methode over van de klasse object.
- → De implementatie is cryptisch.
- → Bedoeling dat je de methode overschrijft met een eigen implementatie.
  - Betekenisvol
  - IDE'S hebben doorgaans een Wizard die de toString( ) methode genereert.

Show me the goods, show me the code!

```
public class Rectangle extends Shape{
   ...

   @Override
   public String toString() {
      return String.format(
       "Rectangle with width %d, height %d at position (%d,%d)",
        width, height, getX(), getY());
   }
   ...
}
```

equals()

- Met deze methode kan je nagaan of twee objecten gelijk zijn.
- Methode van Object gebruikt de identity operator (==)
  - ➔ Werkt voor primitieve datatypes.
  - ➔ **Niet** meer voor **objecten**.
- Methode overschrijven met eigen implementatie zodat er nagegaan wordt of twee objecten gelijke inhoud hebben.

equals( )

Standaard implementatie in Object

➔
```
public boolean equals(Object o){
        return this == o;
}
```

Waaraan moet een eigen implementatie voldoen.

- ➔ Reflexief
- ➔ Symmetrisch
- ➔ Transitief
- ➔ Consistent
- ➔ x.equals(null) moet steeds false teruggeven

Voorbeeldige code.

```
public boolean equals(Object o){
        if((o != null)&&
                (o.getClass()== this.getClass()) &&
                (((Rectangle)o).getX()== getX()) &&
                (((Rectangle)o).getY()== getY()) &&
                 (((Rectangle)o).height== height) &&
                 (((Rectangle)o).width== width))
                 return true;
                 else return false;
}
```

hashCode()

HashCode voorwaarden

- ➔ Als je equals vervangt MOET je hashcode( ) vervangen
  - ◆ Waarom? voor later...later…
- ➔ Consistent
  - ◆ Als er niets veranderd is aan de gegevens die equals( ) methoden bepalen
- ➔ Objecten die volgens de equals( ) methode gelijk zijn moeten gelijke hashcodes hebben.

hashCode( )

- Objecten die volgens de equals( ) methode gelijk zijn moeten gelijke hashcodes hebben.
  - ➔ Unequal objects moeten verschillende hashcodes hebben.
  - ➔ Objecten met gelijke hashcodes moeten equal zijn.

Fout!

hashCode( )

- In een klasse hiërarchie is het aangewezen om de hashcode van de super klasse mee in rekening te brengen.
- Gebruik Eclipse om de hashCode( ) en equals te genereren.

```
public class Rectangle extends Shape{
   ...
   public int hashCode() {
      return getX()*7 + getY()*13 + height*17 + width*23;
   }
   ...
}
```

finalize()

- Speciale methode in Java.
- De garbage collector roept deze methode aan alvorens het object definitief op te ruimen.
- Je kan hier acties ondernemen die zullen uitgevoerd worden bij het opruimen van je object.

Overerving vs. associaties

- Doel : code reuse
  - ➔ kan door overerving
    - Code geschreven in de superklasse moeten we niet meer herschrijven in de subklasse.
  - ➔ kan door associaties
    - Je gebruikt de functionaliteit van een ander object.
- **IS A - Relationship**
- **HAS A - Relationship**