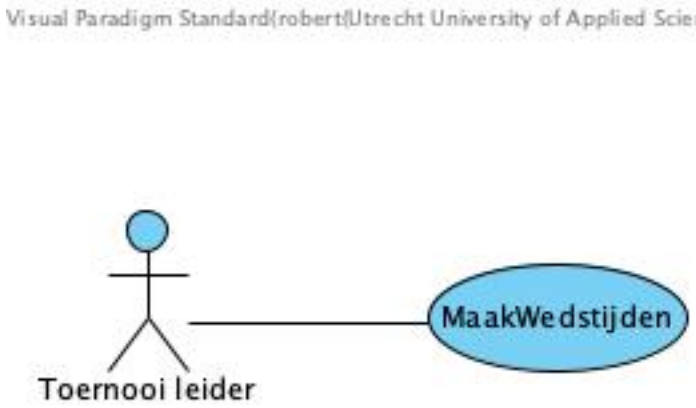
UNIVERSITY OF APPLIED SCIENCES UTRECHT

Demo UML en Domein model

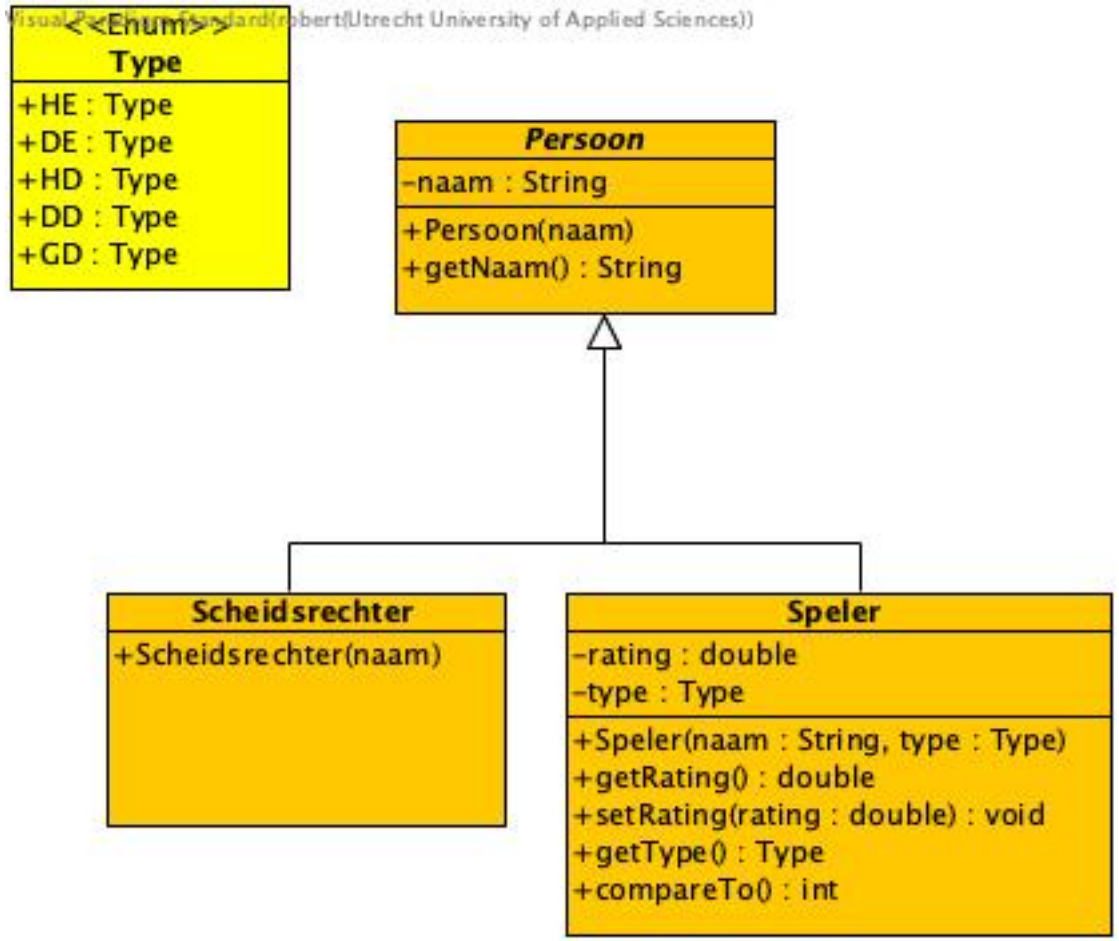
UNIVERSITY OF APPLIED SCIENCES UTRECHT

Use case

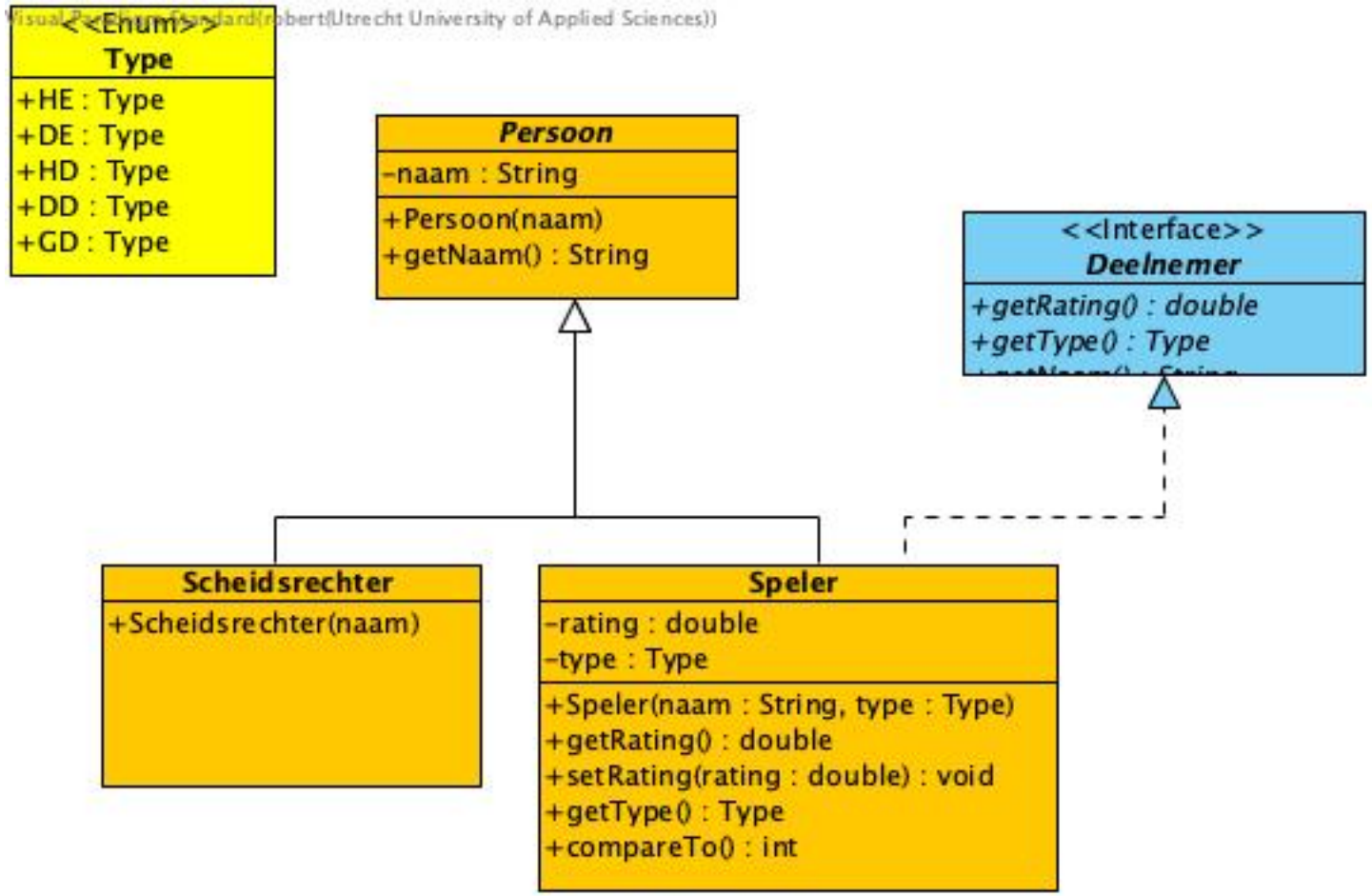


De toernooileider deelt de **wedstrijden** in in een afval **schema**. Om te zorgen dat de beste **deelnemers** (die met de **hoogste rating**)pas in de finale finale tegen elkaar spelen worden in deze in de eerse ronde ver van elkaar in het schema geplaatst. Deelnemers zijn kunnen van het type **enkel** of **dubbel** zijn, en ingedeeld in **sexe** van de speler of spelers. Iedere wedstrijd krijgt een **scheidsrechter**. In het overzicht zijn de **namen** van de **spelers** en de **scheidsrechter** te zien.

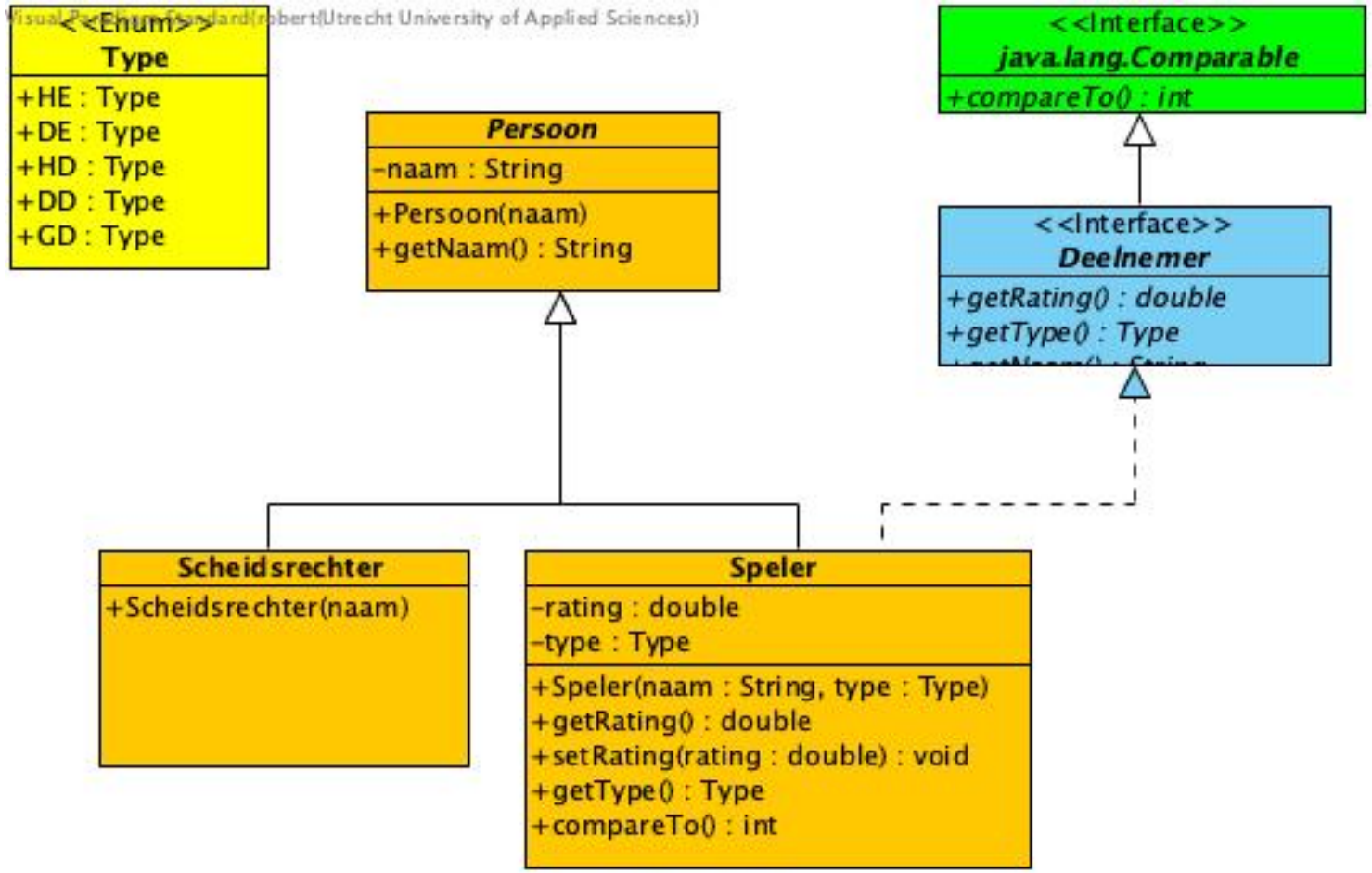
UNIVERSITY OF APPLIED SCIENCES UTRECHT



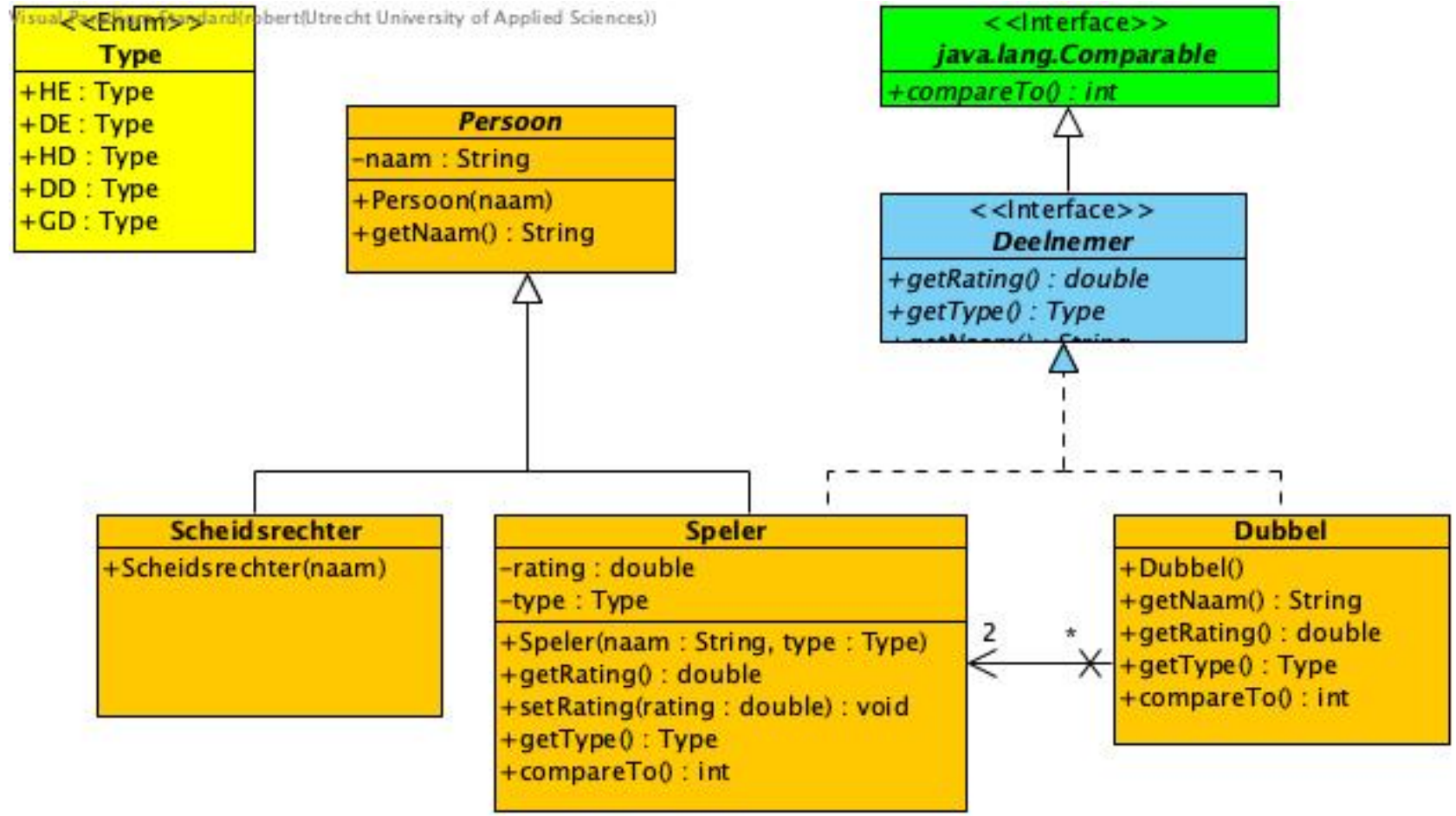
UNIVERSITY OF APPLIED SCIENCES UTRECHT



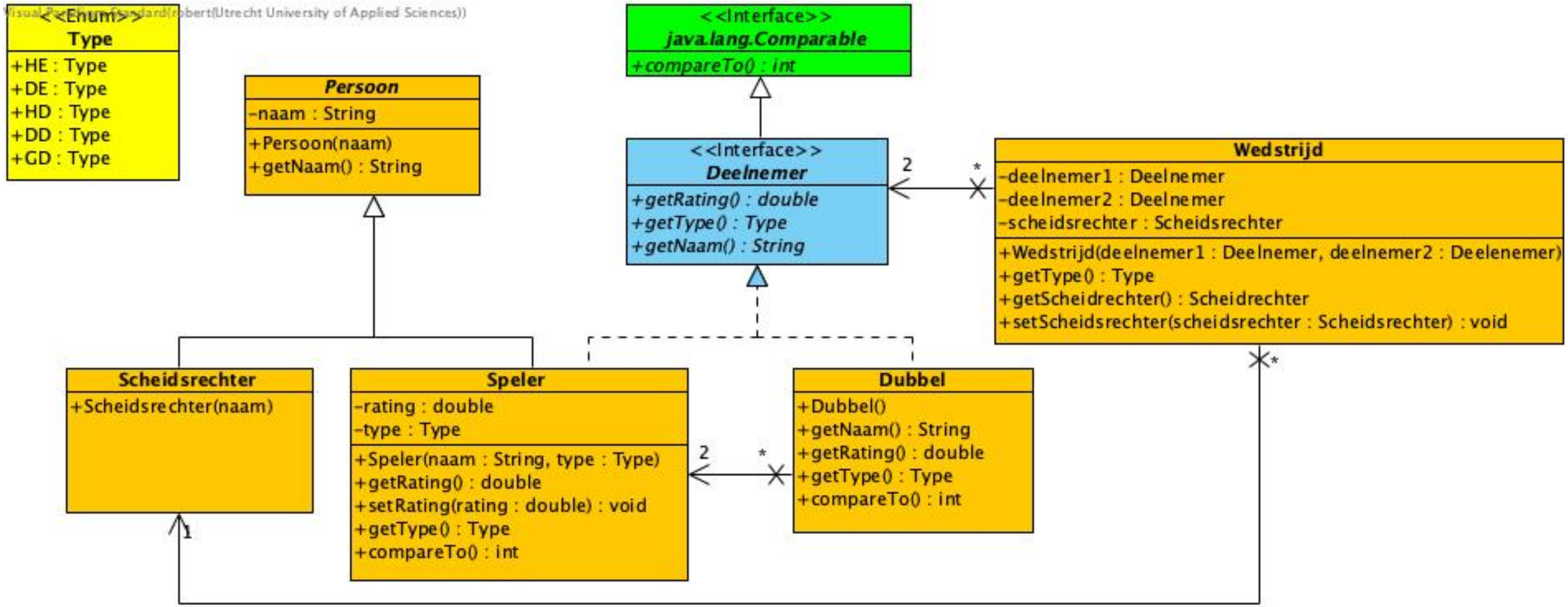
UNIVERSITY OF APPLIED SCIENCES UTRECHT



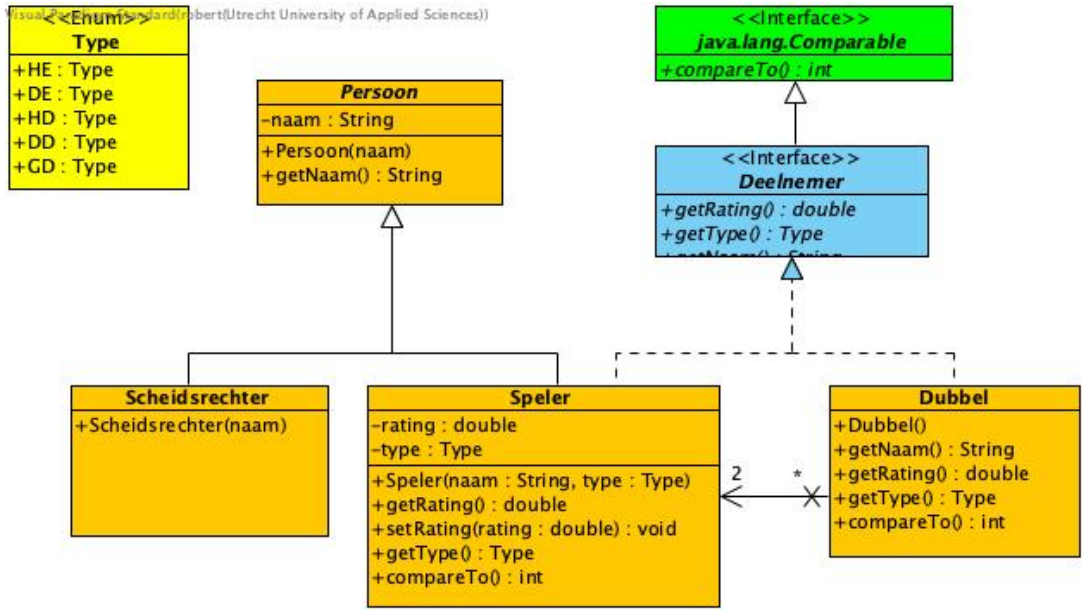
UNIVERSITY OF APPLIED SCIENCES UTRECHT



UNIVERSITY OF APPLIED SCIENCES UTRECHT

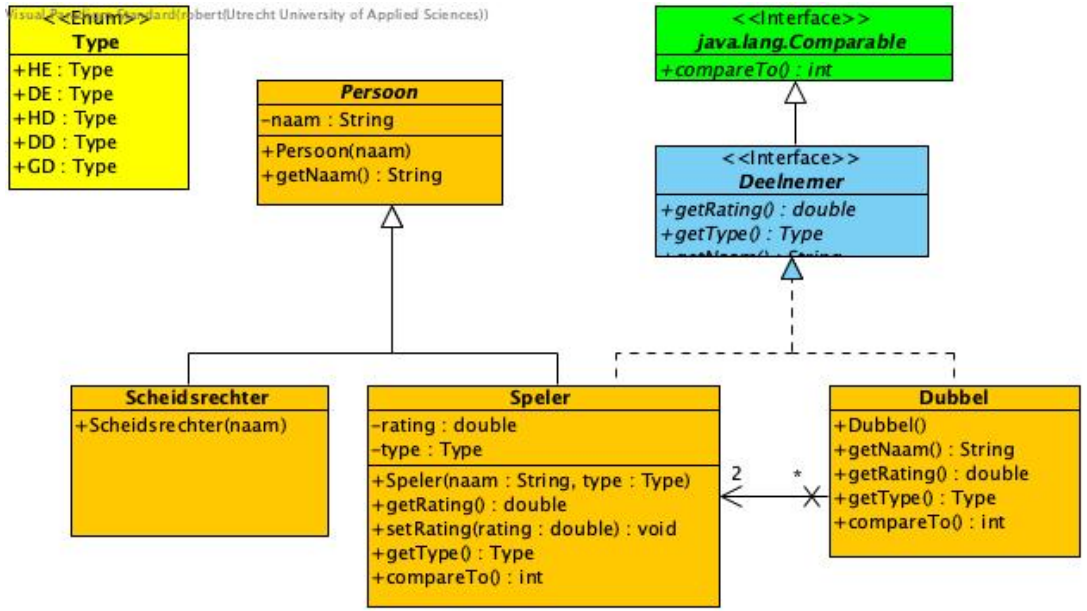


UNIVERSITY OF APPLIED SCIENCES UTRECHT



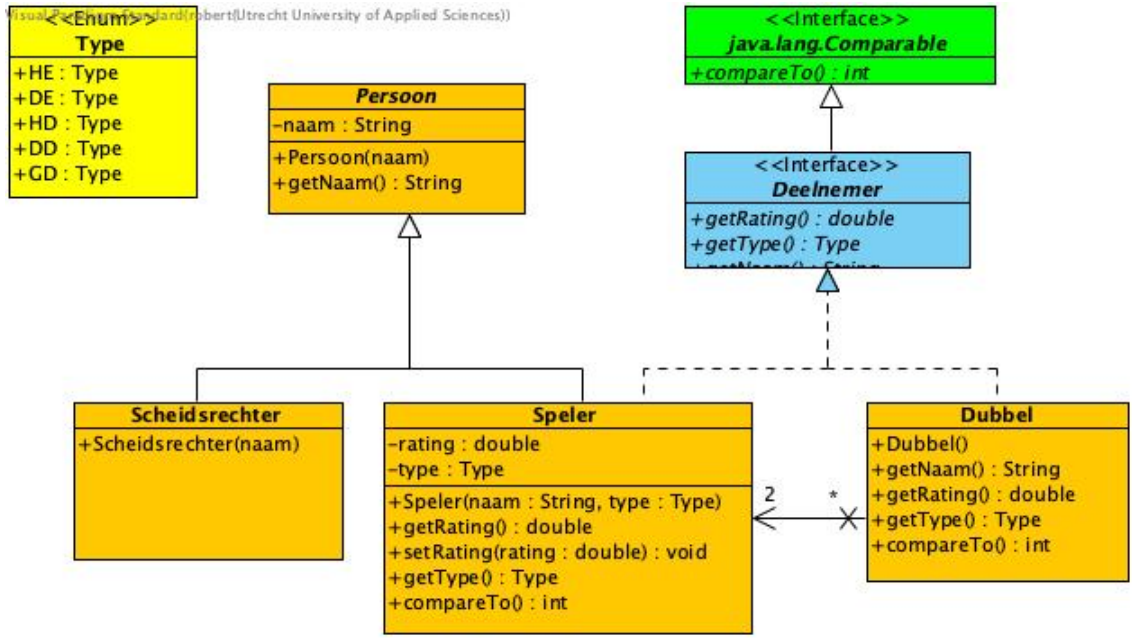

```java
public interface Deelnemer<T> extends Comparable<T> {

    /**
     * geeft de naam van de deelnemer.
     * @return de naam van de deelnemer.
     */
    String getNaam();

    /**
     * Geeft get type van de deelnemer.
     *
     * @return het type van de deelnemer.
     */
    Type getType();

    /**
     * Geeft de rating vsan de deelnemer.
     * @return de rating.
     */
    double getRating();

}
```

UNIVERSITY OF APPLIED SCIENCES UTRECHT




```java
public class Speler extends Persoon implements Deelnemer<Speler>
{

    private double rating;
    private Type type;

    public Speler(String naam,Type type) {
        super(naam);
        this.type = type;
    }

    public Type getType() {
       return type;
    }

    public double getRating() {
        return rating;
    }

    public void setRating(double rating) {
        this.rating = rating;
    }

    public int compareTo(Speler that) {
        if ( this.getRating() < that.getRating()) return -1;
        if ( this.getRating() > that.getRating()) return 1;
        return 0;
    }
}
```

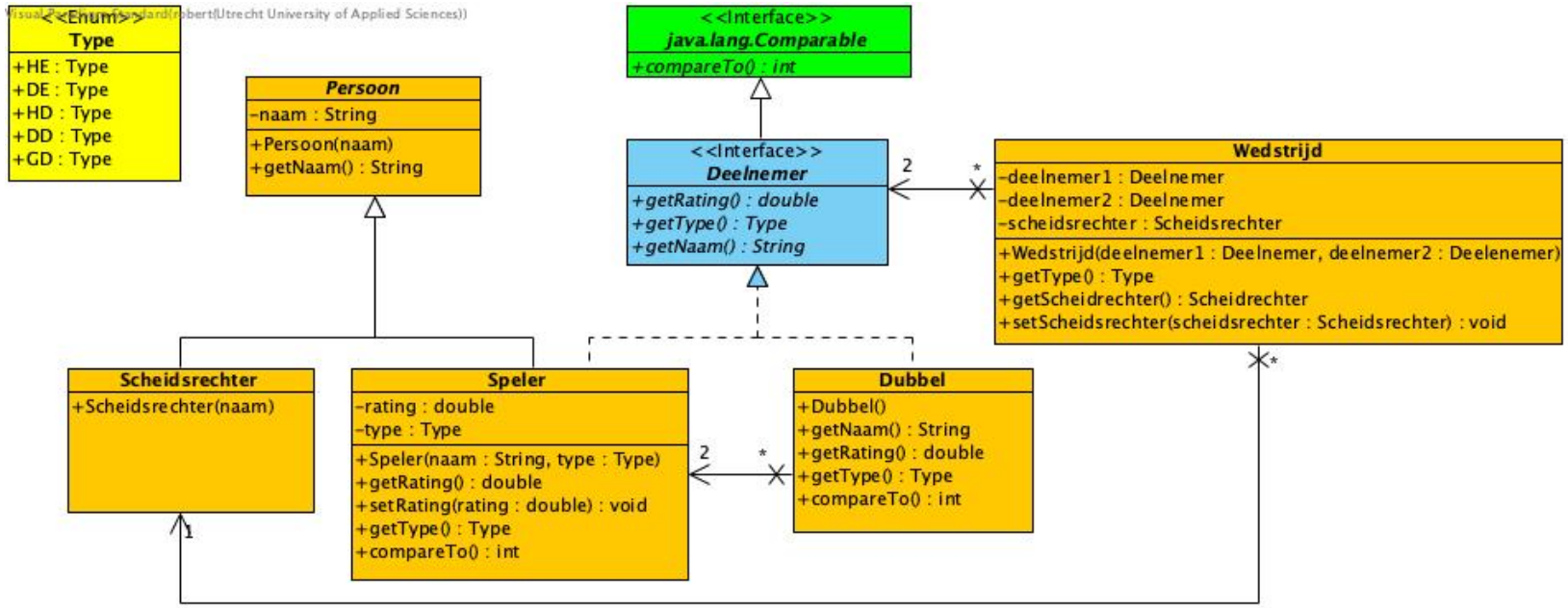
UNIVERSITY OF APPLIED SCIENCES UTRECHT




```java
public class Dubbel implements Deelnemer<Dubbel> {

    private final Speler speler1;
    private final Speler speler2;

    public Dubbel(Speler speler1, Speler speler2) {
        this.speler1 = speler1;
        this.speler2 = speler2;
    }

    public String getNaam() {
        return speler1.getNaam() + "/" + speler2.getNaam();
    }

    public Type getType() {
        if (speler1.getType().equals(HE) &&
            speler2.getType().equals(HE)) return HD;
        if (speler1.getType().equals(DE) &&
            speler2.getType().equals(DE)) return DD;
        return GD;
    }

    public double getRating() {
        return (speler1.getRating() + speler2.getRating())/2;
    }

    public int compareTo(Dubbel that) {
        if ( this.getRating() < that.getRating()) return -1;
        if ( this.getRating() > that.getRating()) return 1;
        return 0;
    }
}
```

UNIVERSITY OF APPLIED SCIENCES UTRECHT

```
public class Wedstrijd {

    private final Deelnemer deelnemer1;
    private final Deelnemer deelnemer2;
    private Scheidsrechter scheidsrechter;

    public Wedstrijd(Deelnemer deelnemer1, Deelnemer deelnemer2) throws ToernooiException {
        if ( ! deelnemer1.getType().equals(deelnemer2.getType())) {
            throw new ToernooiException("Type deelnemer1 is niet gelijk ytype deenemer2");
        }
        this.deelnemer1 = deelnemer1;
        this.deelnemer2 = deelnemer2;
    }

    public Deelnemer getDeelnemer1() {
        return deelnemer1;
    }

    public Deelnemer getDeelnemer2() {
        return deelnemer2;
    }

    public Type getType() {
        return deelnemer1.getType();
    }

    …
    …
```



UNIVERSITY OF APPLIED SCIENCES UTRECHT

```
public List<Deelnemer> sorteerVoorType(final List<Deelnemer> deelnemers, final Type type) {

    return deelnemers.stream()                                          // Stream van deelnemers
            .filter(deelnemer -> deelnemer.getType().equals(type)) // Filter op type
            .sorted()                                                   // sorteer mbv compareTo
            .collect(Collectors.toList());                              //   collect als List
}
```