

OOP - Prinzipien



Allgemeines - Prinzipien

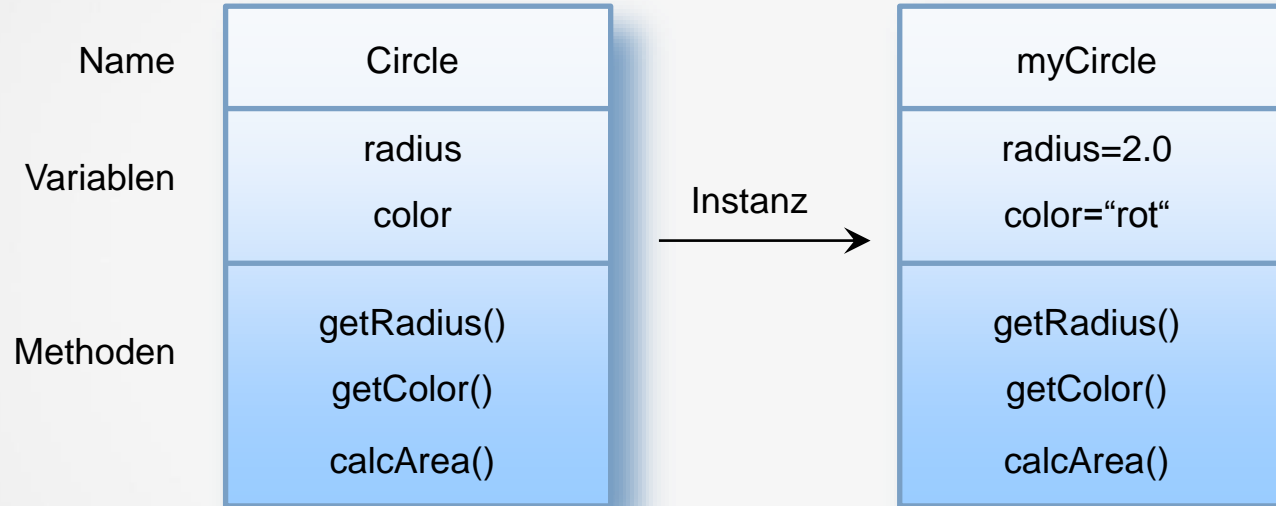
- ✓ Implementierungsdetails nur in der Klasse sichtbar und kann ohne Anpassung der übrigen Programmteile geändert werden
- ✓ Klar definierte Schnittstellen unterstützen arbeiten im Team
- ✓ Erleichtert Debugging, da Fehler besser in Programmteilen=Klassen lokalisierbar

Allgemeines – Kommunikation über Nachrichten



Die Klasse main sendet der Klasse Circle eine Nachricht mit der Methode calcArea der Klasse Circle

Beispielklasse: Circle



Klasse Circle – Datei Circle.java

```
public class Circle {  
    // Private variables  
    private double radius;  
    private String color;  
  
    public Circle(double r) {  
        radius = r;  
        color = "red";  
    }  
    .....  
}
```

Name der Klasse
= Dateiname

Private Variablen: Attribute

Konstruktor: Aufruf bei "new"
" im Hauptprogramm

Klasse Circle – Datei Circle.java

```
.....  
public double getRadius() {  
    return radius;  
}  
public void setRadius(double radius) {  
    this.radius = radius;  
}  
public String getColor() {  
    return color;  
}  
public double calcArea() {  
    return radius*radius*Math.PI;  
}  
}
```

Getter: Liest das private
Attribut von "außen"

Setter: Setzt das private
Attribut von "außen"

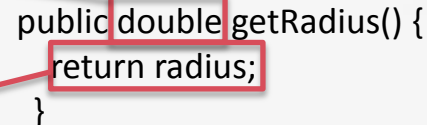
Öffentliche Methode, welche
die privaten Variable radius
verwendet

Beispielklasse: CCircle IV

Datentyp(double) des
Rückgabewertes

Gibt den Radius
zurück(Rückgabewert)

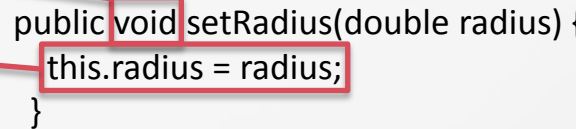
```
public double getRadius() {  
    return radius;  
}
```



Kein Rückgabewert(void)

Setzt den Radius mit dem Wert des
Parameters

```
public void setRadius(double radius) {  
    this.radius = radius;  
}
```



Hauptprogramm – Datei TestCircle.java

```
.....  
public class TestCircle {  
    public static void main(String[] args) {  
        Circle myCircle = new Circle(2.0);  
        System.out.println("Radius is " + myCircle.getRadius()  
            + " Color is " + myCircle.getColor()  
            + " Area is " + myCircle.calcArea());  
    }  
}
```

Erzeugt ein neues Object
myCircle der Klasse Circle

Aufruf der Methode calcArea
der Klasse Circle

Beispielklasse: CCircle V

Circle c1= new Circle();

Aufruf
bei "new"



```
public Circle() {  
    this.radius = 2;  
    this.color = "red";  
}
```

Circle c2= new Circle(2);

Aufruf
bei "new"



```
public Circle(double radius) {  
    this.radius = radius;  
    this.color = "red";  
}
```

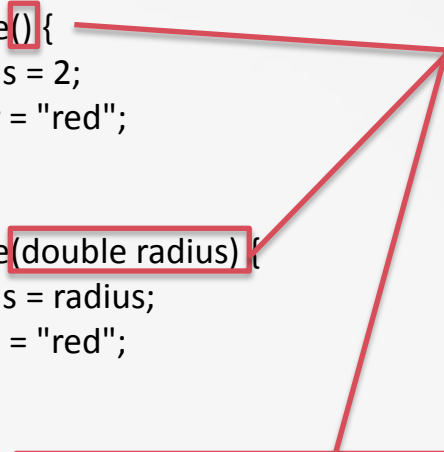
Circle c3= new Circle(3,"red");

Aufruf
bei "new"



```
public Circle(double radius, double color) {  
    this.radius = radius;  
    this.color = "red";  
}
```

„Überladen“ von
Methoden durch
unterschiedliche
Anzahl von
Parametern

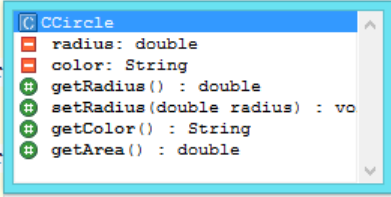


Beispielklasse: CCircle VI

```
public void setRadius(double radius) {  
    this.radius = radius;  
}
```

Verwenden des „this“-Schlüsselwortes um Objekte der Klasse bei Namensgleichheit zu referenzieren

```
public void setRadius(double radius) {  
    this.radius = radius;  
}  
public  
    return  
}  
public  
    return  
}
```



Vereinfachen von Konstruktoren

```
public CCircle()  
{  
    this( 0, "" );  
}
```

Aufruf des Konstruktors mit
zwei Parametern

```
public CCircle( CCircle CCircle )  
{  
    this( CCircle.radius, CCircle.color );  
}
```

Aufruf des Konstruktors mit
zwei Parametern zum Setzen
der Werte aus einer anderen Instanz

```
public CCircle( double radius, String color )  
{  
    this.radius = radius;  
    this.color = color;  
}
```

Konstruktor