Bereich: Klassen (2)

Polynom 2. Grades	**Musterlösung**
Package: `de.dhbwka.java.exercise.classes`	**Klasse:** `Polynomial`

```java
package de.dhbwka.java.exercise.classes;

/**
 * @author DHBW lecturer
 * @version 1.0
 *
 * Part of lectures on 'Programming in Java'.
 * Baden-Wuerttemberg Cooperative State University.
 *
 * (C) 2015 by W. Geiger, T. Schlachter, C. Schmitt, W. Süß
 */
public class Polynomial {

	/* polynomial 2nd degree like ax^2 + bx + c */
	double a, b, c;

	public Polynomial() {
		this(0.0, 0.0, 0.0);
	}

	public Polynomial(double a, double b, double c) {
		super();
		this.a = a;
		this.b = b;
		this.c = c;
	}

    public double f(double x) {
        return a*x*x + b*x + c;
    }

    public Polynomial sub(Polynomial p) {
        return new Polynomial(a-p.a,b-p.b,c-p.c);
    }

    public Polynomial add(Polynomial p) {
        return new Polynomial(a+p.a,b+p.b,c+p.c);
    }

    public Polynomial scale(double factor) {
        return new Polynomial(factor*a,factor*b,factor*c);
    }
```

```
    /** get zeros using the 'midnight formula'
    /*  x1/2 = ( -b +/- sqrt(b*b-4*a*c) ) / 2a */
    public double[] getZeros() {
        if (a == 0) {
            if (b == 0) { // no zeros
                return new double[0];
            } else { // 1 zero
                return new double[]{-c / b};
            }
        } else if (b * b < 4 * a * c) // no zeros
            return new double[0];
        else if (b*b==4*a*c)  // 1 zero
            return new double[] { -b/(2*a) };
        else  // 2 zeros
            return new double[] {
                    (-b+Math.sqrt(b*b-4*a*c))/(2*a),
                    (-b-Math.sqrt(b*b-4*a*c))/(2*a)
            };
    }

      @Override
      public String toString() {
            return a+"x^2 " + (b>=0 ? "+":"") + b+"x "
                        + (c>=0 ? "+":"") + c;
      }

    public static void main(String[] args) {
      Polynomial pol1 = new Polynomial(2,0,0);
      System.out.println("P1: " + pol1);
      Polynomial pol2 = new Polynomial(0,-4,1);
      System.out.println("P2: " + pol2);
      Polynomial pol3 = pol1.add(pol2);
      System.out.println("P3 = P1 + P2: " + pol3);
        pol3 = pol3.scale(2.0);
      System.out.println("P3 = 2.0 * P3: " + pol3);
        double[] zeros = pol3.getZeros();
        System.out.println("Nullstellen von P3 (" + pol3 + "): ");
        for(double zero : zeros)
            System.out.print(zero + " ");
    }

}
```

Bereich: Klassen (2)

Komplexe Zahlen	**Musterlösung**
Package: `de.dhbwka.java.exercise.classes`	**Klasse:** `Complex`

```java
package de.dhbwka.java.exercise.classes;

/**
 * @author DHBW lecturer
 * @version 1.0
 *
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2015 by W. Geiger, T. Schlachter, C. Schmitt, W. Süß
 */
public class Complex {

    private double real = 0.0;
    private double imag = 0.0;

    public Complex() {
             this(0.0, 0.0);
       }

    public Complex(double real, double imag) {
       super();
         this.real = real;
         this.imag = imag;
    }

    public double getReal() {
        return real;
    }

    public double getImag() {
        return imag;
    }

    /** Complex addition
     *  @param comp the complex number to add
     *  @return a new complex as result of addition */
    public Complex add(Complex comp) {
        return new Complex(this.real+comp.real, this.imag+comp.imag);
    }

    /** Complex subtraction
     *  @param comp the complex number to subtract
     *  @return a new complex as result of subtraction */
    public Complex sub(Complex comp) {
        return new Complex(this.real-comp.real, this.imag-comp.imag);
    }
```

```java
/** Complex multiplication
 *  @param comp the complex number to multiply
 *  @return a new complex as result of multiplication */
public Complex mult(Complex comp) {
    return new Complex(this.real*comp.real-this.imag*comp.imag,
         this.real*comp.imag+this.imag*comp.real);
}

/** Complex division
 *  @param comp the complex number to divide
 *  @return a new complex as result of division */
public Complex div(Complex comp) {
    double denom = comp.real*comp.real + comp.imag*comp.imag;
    return new Complex((this.real*comp.real+this.imag*comp.imag)/denom,
         (this.imag*comp.real-this.real*comp.imag)/denom);
}

/** Complex as String */
public String toString() {
    return "Complex " + real + " " + imag + "i";
}

/** Magnitude of a complex number */
public double getMagnitude() {
    return Math.sqrt(real*real+imag*imag);
}

/** true if magnitude if comp is bigger */
public boolean isLess(Complex comp) {
    return (this.getMagnitude()<comp.getMagnitude());
}

/** Sort an array of complex numbers with bubblesort */
public static void sortComplexArray(Complex[] comps) {
    int n = comps.length;
    for(int i=1; i<n; i++)
        for(int j=n-1; j>=i; j--)
         // falls x[j-1] größer als x[j]
            if (comps[j-1].getMagnitude() >
                comps[j].getMagnitude()) {
                Complex tmp = comps[j-1];  // swap
                comps[j-1]=comps[j];
                comps[j]=tmp;
            }
}
```

```
    public static void main(String[] argv) {
        Complex c1 = new Complex(1.0,2.0);
        Complex c2 = new Complex(2.0,1.0);
        System.out.println("C1:     " + c1);
        System.out.println("C2:     " + c2);
        System.out.println("C1+C2:  " + c1.add(c2));
        System.out.println("C1-C2:  " + c1.sub(c2));
        System.out.println("C1*C2:  " + c1.mult(c2));
        System.out.println("C1/C2:  " + c1.div(c2));
        System.out.println("C1<C2?: " + c1.isLess(c2));
        // create an array of 10 random complex numbers
        Complex[] comps = new Complex[10];
        for(int i=0; i<comps.length; i++)
             comps[i]=new Complex(Math.random()*10,Math.random()*10);
        System.out.println("Unsortiert:");
        for(Complex c : comps)
             System.out.printf("%5.3f + %5.3fi  Betrag: %5.3f\n",
                   c.getReal(),c.getImag(),c.getMagnitude());
        // sort array
        sortComplexArray(comps);
        System.out.println("Sortiert:");
        for(Complex c : comps)
             System.out.printf("%5.3f + %5.3fi  Betrag: %5.3f\n",
                   c.getReal(),c.getImag(),c.getMagnitude());
    }

}
```

Bereich: Klassen (2)

Polynome und Horner-Schema	**Musterlösung**
Package: `de.dhbwka.java.exercise.classes`	**Klasse:** `Horner`

```java
package de.dhbwka.java.exercise.classes;

/**
 * @author DHBW lecturer
 * @version 1.0
 *
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2015 by W. Geiger, T. Schlachter, C. Schmitt, W. Süß
 */
public class Horner {

    // coefficients coeff[0] for x^0, coeff[1] for x^1, etc.
    double[] coeff;

    Horner() {
        this(new double[0]);
    }

    Horner(double[] coeff) {
        this.coeff = coeff;
    }

    /** calculate value of Polynomial for a given x
     *  using Horner scheme
     *  @param x
     *  @return value of Polynomial for a given x
     */
    public double getValue(double x) {
        if (coeff.length>0) {
            double sum = coeff[coeff.length-1];
            for (int i = coeff.length-2; i >= 0; i--)
                sum = sum * x + coeff[i];
            return sum;
        } else
            return 0;
    }

    @Override
    public String toString() {
        StringBuffer s = new StringBuffer("");
        for (int i = coeff.length-1; i >= 0; i--) {
            s.append(coeff[i]+"*x^"+i+" ");
            if (i>0 && coeff[i-1] >= 0)
                s.append("+");
        }
        return s.toString();
    }
```

```
    public static void main(String[] args) {
        double[] k = {1,-2,3,4.5,8,-10,3,4,2,-3,0.5};
        Horner p = new Horner(k);
        double x = 1.5;
        System.out.println("Polynomial f: " + p.toString());
        System.out.println("f(" + x + ") = " + p.getValue(x));
    }

}
```