Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik
Lehrstuhl für Softwaretechnik

Prof. Dr. Wolf Zimmermann
Annett Thüring

Ausgabe: 2008-01-17

Abgabe: 2008-01-24

Objektorientierte Programmierung

13. Übungsblatt

Hinweis: Bitte alle Programme kommentieren! Programme ohne Kommentar werden mit 0 Punkten bewertet!

Aufgabe 1 (Punkte: 5)

a) Welche der folgenden Aussagen sind wahr?

1. In einer abstrakten Klasse müssen alle Methoden als abstrakte Methoden definiert werden.
2. In einer Schnittstelle sind per Definition alle Methoden abstrakt.
3. Die Elemente einer Schnittstelle besitzen implizit die Zugriffsklasse public.
4. Wenn bei Elementen einer abstrakten Klasse der Modifizierer `public, privat` nicht angegeben wurde, so besitzen die Elemente implizit die Zugriffsklasse public.

b) Nennen Sie jeweils eine weitere Eigenschaft für eine Scnittstelle bzw. für eine abstrakte Klasse.

c) Schreiben Sie ein Hauptprogramm, was unter Verwendung der definierten Klassen und Schnittstellen den Text **„Never send a human to do a machine's job."** auf der Kolsone ausgibt.

```
public interface IWord
    {
        void Print();
    }
    public interface IWord2 : IWord
    {
        new void Print();
    }
    public abstract class Base
    {
        public Base() { Console.Write(this.GetString()); }
        static Base() { Console.Write("Never "); }
        public virtual void Print() { Console.Write("to "); }
        protected virtual string GetString() { return "hu "; }
        protected static string msg = "send ";
    }
    public class Derived : Base, IWord
    {
        static Derived() { Console.Write(Derived.msg); }
        public new virtual void Print() { Console.Write("do "); }
        protected override string GetString() { return "a "; }
    }
    public sealed class MoreDerived : Derived, IWord
```

```
{
    public override void Print() { Console.Write("mach"); }
    void IWord.Print() { Console.Write("a "); }
    protected override string GetString() { return "do "; }
}
public sealed class MoreDerived2 : Derived, IWord2
{
    static MoreDerived2() { Console.Write("ine"); }
    public new void Print() { Console.Write("job. "); }
    void IWord2.Print() { Console.Write("job."); }
    protected override string GetString() { return "'s "; }
}
public abstract class Unfinished : Base
{
    protected new void Print() { Console.Write("man "); }
    protected override string GetString() { return "human "; }
}
public class Finished : Unfinished
{
}
```

Aufgabe 2 (Punkte: 4)

Gegeben ist die Klasse `Viereck`.

```
class Viereck
{
    protected double a;       // Laenge der Seiten des Vierecks
    protected double b;
    protected double c;
    protected double d;

                              // Setzen der Seitenlaengen
    public void set_a (double sa) { a = sa; }
    public void set_b (double sb) { b = sb; }
    public void set_c (double sc) { c = sc; }
    public void set_d (double sd) { d = sd; }

                              // Ausgabe der Daten
    public virtual void print()
    {
        Console.WriteLine("Laenge der Seiten des Vierecks");
        Console.Write("Seite a = " + a + ", Seite b = " + b);
        Console.WriteLine(", Seite c = " + c + ", Seite d = " + d);
    }
}
```

a) Leiten Sie von dieser Klasse `Viereck` die Klasse `Rechteck` ab. Ergänzen Sie diese Klassen um eine Methode `flaecheninhalt`, welche bei Aufruf den Flächeninhalt berechnet und zurückgibt. Überschreiben Sie die Methode `print` der Basisklasse, so dass die zum Rechteck gehörigen Daten Höhe und Breite ausgegeben werden.

b) Leiten Sie weiterhin von der Klasse `Viereck` eine Klasse `Parallelogramm` ab. Fügen Sie dieser Klasse das Attribute `winkel` vom Typ float und eine zugehörige Methode `set_w`, über welche der Winkel aus dem Hauptprogramm gesetzt werden kann, hinzu. Ergänzen Sie auch diese Klassen um eine Methode `flaecheninhalt`. Verwenden Sie bei der Berechnung die Funktionen `Math.Sin()` (Achtung Parameter wird im Bogenmass erwartet!) und `Math.PI`. Überschreiben Sie ebenfalls die Methode `print` der Basisklasse, so dass die zum Parallelogramm gehörigen Daten, Länge der Seiten, Höhe und Winkel, ausgegeben werden.

c) Schreiben Sie weiterhin eine Klasse `Quadrat`, welche von der Klasse `Rechteck` abgeleitet ist. Implementieren Sie hier ebenfalls eine Methode `flaecheninhalt` und überschreiben Sie die Methode `print` entsprechend. Diese soll nur noch die Seitenlänge des Quadrates ausgeben.

Verwenden Sie zum Testen das gegebene Hauptprogramm.

```
static void Main(string[] args)
        {
            Rechteck r1;
            r1 = new Rechteck();
            r1.set_a(5);
            r1.set_b(6);
            r1.print();
            Console.WriteLine("Flaeche Rechteck = " + r1.flaecheninhalt() + "\n");

            Quadrat q1;
            q1 = new Quadrat();
            q1.set_a(9);
            q1.print();
            Console.WriteLine("Flaeche Quadrat = " + q1.flaecheninhalt() + "\n");

            Parallelogramm p1;
            p1 = new Parallelogramm();
            p1.set_a(4);
            p1.set_b(3);
            p1.set_w(45);        // Winkel im Gradmass
            p1.print();
            Console.WriteLine("Flaeche Parallelogramm = " + p1.flaecheninhalt() + "\n");
        }
```

Hinweis: Senden Sie nur Ihre drei Klassen `Rechteck`, `Parallelogramm` und `Quadrat` ein, nicht das Hauptprogramm, die Klasse `Viereck` den Namespace oder using-Direktiven. Einsendungen die sich nicht an dieses Konventionen halten, werden mit 0 Punkten bewertet.