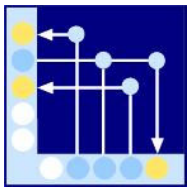
Hochschule Aalen
Fakultät Elektronik und Informatik
Studienbereich Informatik



Objektorientierte Programmierung

Vorlesung im Wintersemester 2023/2024
Prof. Dr. habil. Christian Heinlein

8. Übungsblatt (8. – 20. Dezember 2023)

Aufgabe 8: Schnittstellen

Implementieren Sie folgende Schnittstellen und Klassen (vgl. UML-Diagramm am Ende)!

Schnittstellen

- Geometrische Objekte (Schnittstelle `Figure`) besitzen Breite, Höhe und Fläche (abstrakte Methoden `width`, `height` und `area`).
- Rechtecke (`Rectangle`) und Ellipsen (`Ellipse`) sind geometrische Objekte, deren Fläche aus ihrer Breite und Höhe berechnet werden kann (vorimplementierte Methode `area`).
- Rechtecke besitzen zusätzlich eine Diagonale, die ebenfalls aus Breite und Höhe berechnet werden kann (vorimplementierte Methode `diag`).
- Regelmäßige geometrische Objekte (`RegularFigure`) sind geometrische Objekte, deren Größe mit ihrer Breite und Höhe übereinstimmt (vorimplementierte Methode `size`).
- Quadrate (`Square`) und Kreise (`Circle`) sind regelmäßige Rechtecke bzw. Ellipsen.
- Die Diagonale eines Quadrats kann aus seiner Größe effizienter berechnet werden als die Diagonale eines Rechtecks (Überschreibung der Methode `diag`).
- Kreise besitzen zusätzlich einen Durchmesser, der mit der Größe übereinstimmt, und einen Radius, der halb so groß wie der Durchmesser ist (vorimplementierte Methoden `diam` und `rad`).

Abstrakte Hilfsklassen

- Die abstrakten Klassen `FigureData` und `RegularFigureData` definieren die zur Repräsentation allgemeiner bzw. regelmäßiger geometrischer Objekte benötigten Datenstrukturen sowie zugehörige unterklassenöffentliche Konstruktoren zu ihrer Initialisierung.
- `RegularFigureData` besitzt hierfür eine private Objektvariable `size`, die zur Speicherung der Größe eines regelmäßigen geometrischen Objekts verwendet wird und implementiert damit die Methoden `width` und `height`.
- `FigureData` erbt von `RegularFigureData` die Variable `size` und verwendet sie zur Speicherung der Breite eines allgemeinen geometrischen Objekts. Zusätzlich besitzt die Klasse eine private Objektvariable `height` zur Speicherung der Höhe, d. h. die Methode `height` wird entsprechend überschrieben.
- `RegularFigureData` implementiert die Schnittstelle `RegularFigure` bewusst nicht, weil sonst auch `FigureData` und seine unten beschriebenen Unterklassen `RectangleImpl` und `EllipseImpl` indirekte Untertypen von `RegularFigure` wären, was logisch falsch wäre. `FigureData` implementiert dann die Schnittstelle

`Figure` konsequenterweise auch nicht, obwohl das keine logischen Fehler verursachen würde.
Aufgrund dieser fehlenden `implements`-Beziehungen müssten die beiden Klassen „technisch“ nicht abstrakt sein. „Logisch“ sind sie jedoch abstrakt, weil man von ihnen direkt keine Objekte erzeugen soll.

Konkrete Implementierungsklassen

- Die Klassen `RectangleImpl`, `EllipseImpl`, `SquareImpl` und `CircleImpl` erben von `FigureData` bzw. `RegularFigureData` und implementieren die korrespondierenden Schnittstellen, wofür keine weiteren Methodenimplementierungen erforderlich sind.
- Sie besitzen jeweils einen öffentlichen Konstruktor, der als Parameter Breite und Höhe eines Rechtecks oder einer Ellipse bzw. die Größe eines Quadrats oder Kreises erhält.

UML-Diagramm