Übung 02

Responsibility Driven Design

http://www.hwk-chemnitz.de/fileadmin/user_upload/ausbildung/Pruefungen/Gesellenpruefungen/Fragezeichen.JPG

1. Responsibility Driven Design

Führen Sie eine detaillierte Analyse durch und finden Sie mit ihrer Hilfe alle **Klassen**, **Verantwortlichkeiten**, **Kollaborationen** und **Beziehungen**. Begründen Sie Ihre Entscheidung.

- **Klassen**
  - Finden von Nomen Phrasen
  - Kategorisierung & Verfeinerung
- **Verantwortlichkeiten**
  - Wissensverwaltung
  - Ausführbare Aktionen
  - Öffentliche Leistungen
- **Kollaborationen**
  - Kann die Klasse die Verantwortlichkeit selbstständig erfüllen?!
- **Beziehungen**
  - Is-part-of
  - Has-knowledge-of
  - Is-kind-of

1.1 RDD: Klassen

Die neue Bildbearbeitungssoftware intelliPhoto ist ein interaktives Tool zum Anzeigen und Bearbeiten von Bildern. Jedes Bild wird durch ein zweidimensionales Array von Bytes repräsentiert, wobei jeder Byte-Wert für einen Farbwert des Bildpunktes steht. Der Benutzer soll in der Lage sein die Bilddimensionen abzufragen. Es sollen zwei verschiedene Arten von Bildern repräsentiert werden können: "RasterImage" und "ShapedImage", wobei letzteres eine Spezialform vom "RasterImage" ist. Ein "ShapedImage" besitzt eine nicht-rechteckige Form (Polygon), wobei die Bytes im Array angeben, ob die jeweiligen Punkte transparent oder opak dargestellt werden sollen. Darüber hinaus soll die Software einfache Manipulationen von Bildern erlauben. So soll das Drehen, als auch das Vergrößern und Verkleinern von Bildern, das Setzen neuer Farbwerte im Bild und das Zusammenfügen zweier Bilder zu einem neuen Bild innerhalb von 0.2 Sekunden möglich sein.

1.1 RDD: Klassen

Die neue ~~Bildbearbeitungssoftware~~ intelliPhoto ist ein interaktives ~~Tool~~ zum Anzeigen und Bearbeiten von Bildern. Jedes Bild$_{\text{=Interface/Klasse}}$ wird durch ein ~~zweidimensionales Array~~$_{\text{=Impl.-Detail}}$ von Bytes repräsentiert, wobei jeder ~~Byte-Wert~~$_{\text{=Impl.-Detail}}$ für einen ~~Farbwert~~$_{\text{=nicht explizit modelliert}}$ des ~~Bildpunktes~~$_{\text{=nicht explizit modelliert}}$ steht. Der ~~Benutzer~~$_{\text{=nicht explizit modelliert}}$ soll in der Lage sein die Bilddimensionen$_{\text{=Attribut}}$ abzufragen. Es sollen zwei verschiedene Arten von Bildern repräsentiert werden können: "RasterImage“$_{\text{=Klasse}}$ und "ShapedImage“$_{\text{=Klasse,}}$ wobei letzteres eine Spezialform vom "RasterImage" ist. Ein "ShapedImage" besitzt eine nicht-rechteckige$_{\text{=Klasse}}$ Form$_{\text{=Klasse}}$ (Polygon$_{\text{=Klasse}}$), wobei die Bytes im Array angeben, ob die jeweiligen Punkte$_{\text{=Klasse}}$ transparent oder opak dargestellt werden sollen. Darüber hinaus soll die Software einfache ~~Manipulationen~~$_{\text{=Methoden}}$ von Bildern erlauben. So soll das Drehen, als auch das Vergrößern und Verkleinern von Bildern, das Setzen neuer Farbwerte im Bild und das Zusammenfügen zweier Bilder zu einem neuen Bild innerhalb von 0.2 Sekunden möglich sein.

1.2 RDD: Verantwortlichkeiten

Die neue Bildbearbeitungssoftware intelliPhoto ist ein interaktives Tool zum Anzeigen und Bearbeiten von Bildern. Jedes Bild wird durch ein zweidimensionales Array von Bytes repräsentiert, wobei jeder Byte-Wert für einen Farbwert des Bildpunktes steht. Der Benutzer soll in der Lage sein die Bilddimensionen abzufragen. Es sollen zwei verschiedene Arten von Bildern repräsentiert werden können: "RasterImage" und "ShapedImage", wobei letzteres eine Spezialform vom "RasterImage" ist. Ein "ShapedImage" besitzt eine nicht-rechteckige Form (Polygon), wobei die Bytes im Array angeben, ob die jeweiligen Punkte transparent oder opak dargestellt werden sollen. Darüber hinaus soll die Software einfache Manipulationen von Bildern erlauben. So soll das Drehen, als auch das Vergrößern und Verkleinern von Bildern, das Setzen neuer Farbwerte im Bild und das Zusammenfügen zweier Bilder zu einem neuen Bild innerhalb von 0.2 Sekunden möglich sein.

1.2 RDD: Verantwortlichkeiten

Die neue Bildbearbeitungssoftware intelliPhoto ist ein interaktives Tool zum Anzeigen$_{=Methode}$ und ~~Bearbeiten~~ von Bildern. Jedes Bild wird durch ein zweidimensionales Array von Bytes repräsentiert, wobei jeder Byte-Wert für einen Farbwert des Bildpunktes steht. Der Benutzer soll in der Lage sein die Bilddimensionen abzufragen. Es sollen zwei verschiedene Arten von Bildern repräsentiert werden können: "RasterImage" und "ShapedImage", wobei letzteres eine Spezialform vom "RasterImage" ist. Ein "ShapedImage" besitzt eine nicht-rechteckige Form (Polygon), wobei die Bytes im Array angeben, ob die jeweiligen Punkte transparent oder opak dargestellt werden sollen. Darüber hinaus soll die Software einfache Manipulationen von Bildern erlauben. So soll das Drehen$_{=Methode}$, als auch das Vergrößern$_{=Methode}$ und Verkleinern$_{=Methode}$ von Bildern, das Setzen neuer Farbwerte$_{=Methode}$ im Bild und das Zusammenfügen zweier Bilder$_{=Methode}$ zu einem neuen Bild innerhalb von 0.2 Sekunden möglich sein.

1.3 RDD: Zusammenfassung

Klassen

- Bild/Image
- RasterImage
- ShapedImage
- Form / Shape
  - Rechteck
  - Polygon
- Punkt / Point

Verantwortlich-keiten

- zeichnen
- vergrößern
- verkleinern
- rotieren
- einfärben
- zusammenfügen

Beziehungen

- RasterImage is kind of Image
- ShapeImage is kind of RasterImage
- Dimension is part of Image
- Rectangle is kind of Shape
- Polygon is kind of Shape
- Manipulations has knowledge of Image

Kollaborationen

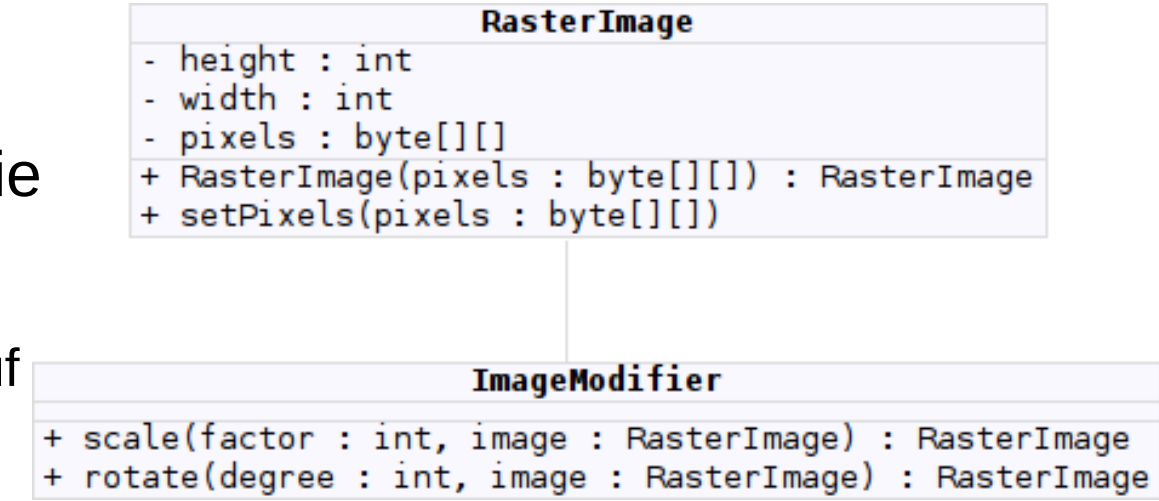
- Image kann seine Verantwortlichkeiten selbst erfüllen
  - Sehr gute Kapselung der Daten möglich!
- Weitere Verantwortlich-keiten sind nicht zu berücksichtigen

1.4 RDD: Klarstellung

- Klasse BildManipulation (Tools) vorgeschlagen
  - SkalierungsTool
  - DrehTool
  - VereinigungsTool
  - ...
- ABER BildManipulation müsste Zugriff auf die interne „private“ Repräsentation von Image haben!
  - ➢ jede andere Klasse könnte demnach auch darauf zugreifen
    - ➢ Hohe Koppelung
    - ➢ Bricht Konzept der Kapselung / Information Hiding

➢ Image weiß selbst am Besten, wie es damit umgehen soll

➢ Lösung: Vertrag (Interface) schaffen



2. Kapselung

Was versteht man unter dem Begriff *Kapselung* im Kontext der (Objektorientierten) Programmierung? Welche Vorteile bringt dieses Konzept mit sich?

- Details der Implementierung bleiben nach außen verborgen (interne Datenstruktur, Typen, ...) → Black-Box-Modell
- Programmierer muss nur die vom Objekt unterstützten Methoden und deren Parameter kennen
- Implementierung der Operation, die durch eine Methode ausgeführt wird, ist für den Benutzer nicht relevant → Änderung dieser Operation beeinträchtigt nicht die Zusammenarbeit der Klassen
- Das Objekt ist für die Ausführung der in der Methode entsprechenden Operationen verantwortlich
- Leichtere Testbarkeit, da nur öffentliche Schnittstellen getestet werden müssen
- Übersichtlicher, klarer strukturiert, stabiler

http://www.hwk-chemnitz.de/fileadmin/user_upload/ausbildung/Pruefungen/Gesellenpruefungen/Fragezeichen.JPG