Morphing Tool Probe - Lösungsbeispiele

Planung und Zusammenarbeit

Frage 1: Zur Planung und Zusammenarbeit während der Experten- und Softwarteile: Welche Planungspunkte haben gut funktioniert? Welche würden Sie das nächste Mal weglassen? Und welche würden Sie das nächste Mal hinzufügen?

Ihre Antwort umfasst drei Teile. Als Erstes **nennen** Sie die Planungspunkte. Als Zweites **illustrieren** Sie **wie** und **wo** Sie die Planungspunkte konkret umgesetzt haben oder umsetzen wollen. Als Drittes **begründen** Sie, welchen Vorteile aus den Planungspunkten hervor gehen. Für die Planungspunkte, welche Sie weglassen wollen, **begründen** Sie, warum Sie diese weglassen wollen.

2 Punkte: Drei unterschiedliche, vollständig beantwortete Planungspunkte.

1 Punkt: Zwei unterschiedliche, vollständig beantwortete Planungspunkte.

Lösung:

- Was: Gleiche Namen für die Punkt und Booleanvariablen wählen. Wie/Wo: In ImageContainer, MorphingContainer und MorphingGUI die gleichen Namen wählen. Vorteil: Einfacher Code zu lesen, einfacherer Zusammenbau der Expertenteile.
- Was: Expertenteil bestimmen, mit dem der Zusammenbau angefangen wird. Wie/Wo: In der Softwarerunde F, dieser Teil wird mit den anderen ergänzt. Vorteil: Bessere Kommunikation / Planung, alle wissen schon im Vorfeld, wer was wo anpassen muss.
- Was: Abmachen wie wann Kommuniziert wird. Wie/Wo: In der Softwarerunde F, z.B. via E-Mail. Vorteil: Es wird frühzeitig angefangen und nicht alles auf die letzte Minute verschoben.
- Was: Die Arbeit für den Zusammenbau aufteilen. Wie/Wo: In der Softwarerunde F arbeiten nicht alle gleichzeitig sondern gestaffelt. Vorteil: Weniger Aufwand für jeden Beteiligten und effizientes Vorankommen.
- Was: Refactoring. Wie/Wo: Expertenteile. Vorteil: Vereinfacht den Zusammenbau, da die Expertenteile für alle einfacher zu verstehen sind.
- Was: Nicht überlegt, wie bei Problemen vorgehen. Umsetzen: Vorher mögliche Problemquellen identifizieren und Massnahmen beschliessen, z.B. bei Problem mit dem Einsatz einer Methode, diese im Internet oder Buch nachschlagen oder sofort Frage ins Forum schreiben und sich bis zur Beantwortung mit anderen Dingen beschäftigen.
- Planungspunkte zum Weglassen: Individuelle Antworten möglich, alle drei Antwortteile müssen ersichtlich sein.

Frage 2: Worin liegt der Sinn und Nutzen einer detaillierten Softwareplanung? Formulieren Sie gemäss Ihrer persönlichen Erfahrung eine „ideale“ Planung.

Ihre Antwort umfasst zwei Teile. Als Erstes **nennen** Sie Gründe für den **Sinn und Nutzen** einer Softwareplanung. Als Zweites **entwerfen** Sie eine „**ideale**“ Planung basierend auf **Ihrer Erfahrung**, hierbei genügen **Stichworte** als Antwort.

2 Punkte: Drei verschiedene Gründe für den Sinn und Nutzen und **sechs verschiedene** Planungspunkte für die ideale Planung.

1 Punkt: Zwei verschiedene Gründe für den Sinn und Nutzen und **vier verschiedene** Planungspunkte für die ideale Planung.

Lösung:
- Sinn und Nutzen: Durch Aufgabenverteilung den Aufwand für jede Person verringern. Durch Koordination der Tätigkeiten Leerläufe vermeiden. Durch Planen der Kommunikation Missverständnisse vermeiden. Mögliche Fehlerquellen im Vorfeld erkennen und Massnahmen im Voraus beschliessen, das erspart Stress und Ärger. Bringt eine erste Auseinandersetzung mit dem Programm mit sich.
- Ideale Planung: Individuelle Antworten möglich.

Frage 3: Im Laufe dieses Projekts haben Sie sich zur Planung und Organisation viele Gedanken gemacht und Erfahrungen gesammelt. Uns ist es wichtig, dass dieses Wissen nicht irgendwo verstaubt, sondern dass Sie es auf für andere Projekte nutzen. Vergleichen Sie daher die Planung und Organisation dieses Projekts mit der Planung und Organisation Ihres Lernens auf die **schriftlichen** Maturaprüfungen. Wie stellen Sie sicher, dass Sie die Matura bestehen werden?

Ihre Antwort umfasst zwei Teile. Als Erstes **identifizieren** Sie Massnahmen der Planung und Organisation innerhalb dieses Projekts, welche Sie auch für Ihr Ziel „Matura bestehen" konkret **einsetzen** wollen. Als Zweites **schildern** Sie, wie Sie diese Massnahmen für Ihr Ziel „Matura bestehen" **umsetzen** wollen. Ihre Schilderungen zur Umsetzung beantworten die folgenden **drei** Punkte.

A) **Wie** setzen Sie die Massnahme **konkret** um?

B) **Wann** oder / und **wie oft** wenden Sie die Massnahme in etwa an?

C) **Wie viel Zeit** planen Sie für die Umsetzung der Massnahme ein?

2 Punkte: Fünf unterschiedliche, vollständige Massnahmen.

1 Punkte: Drei unterschiedliche, vollständige Massnahmen.

Lösung:
- Unterschiede: Hier das Projekt dauert wenige Wochen, bei Maturaprojekt viel längerer Zeithorizont. Hier Arbeit vorwiegend in Gruppen, bei Maturaprojekt vorwiegend allein. Hier Termine für Zwischenabgaben, bei Maturaprojekt sind diese selber zu setzten. Hier jeder Teilschritt wird kontrolliert, bei Maturaprojekt nicht.
- Gemeinsamkeit: Planen was wann gemacht wird. Einsatz alle paar Wochen, Monate. Umsetzung durch Sichten des Stoffs und schreiben einer To-Do Liste. Zeitdauer jeweils c.a. 15 Minuten.
- Gemeinsamkeit: Möglichkeit zu Fragen. Einsatz unregelmässig bei Unklarheiten während dem Lernen. Fragen persönlich oder per E-Mail an Kollegen Lehrer. Zeit, genügend Zeit dafür einplanen, frühzeitig anfangen.
- Gemeinsamkeit: Es ist nicht gut, alles auf den Letzten Moment zu verschieben. Aufgaben in Teile aufteilen. Sich für jeden Teil Deadlines setzten und Arbeitszeiten planen. Häufigkeit, in der Vorbereitung für jedes Thema. Zeitdauer jeweils c.a. 30 Minuten.
- Gemeinsamkeit: Es lohnt sich grosse Aufgaben in Teilaufgaben aufzuspalten. Aufgaben in Teile aufteilen. Sich für jeden Teil Deadlines setzten und Arbeitszeiten planen. Häufigkeit, in der Vorbereitung für jedes Thema. Zeitdauer jeweils c.a. 30 Minuten.

Umsetzung des Expertenteils

Expertenthema – GUI und Steuerung

Frage 4: Das Morphing Softwareprojekt orientiert sich am Model-View-Control Konzept, welches einen Spezialfall des Encapusation Prinzips darstellt. Die Programmiersprache Java fasst die View und Control Komponenten aus Effizienzgründen zusammen. Beurteilen Sie, mit welcher Konsequenz im Morphing Tool die Model Komponente von der View-Control Komponente getrennt wurde.

Ihre Antwort umfasst vier Teile. Als Erstes **identifizieren** Sie welche Klasse(n) hauptsächlich für View-Control zuständig sind und welche hauptsächlich für das Model. Eine strikte Trennung wurde in diesem Projekt jedoch nicht umgesetzt. Als Zweites **geben** Sie an, mit welchen programmtechnischen Elementen die Trennung in den Klassen „**unterstützt“** wird. Als Drittes beschreiben Sie wo die Trennung „**verletzt**“ wird. Als Viertes schildern Sie der Idee nach, was man im Source Code verändern müsste, um dennoch eine strikte Trennung zu erhalten. Welche Nachteile würde dieser Schritt mit sich bringen?

Maximal 4 Punkte: Pro vollständig beantworteten Antwortteil bekommen Sie je 1 Punkt.

Lösung:

- Klassen: MorphingGUI für View-Control, ImageContainer und MorphingContainer für das Model.
- Unterstützende Elemente: Alle Events werden in der MorphingGUI Klasse abgefragt und mit Methoden verknüpft. Die Instanzvariablen der Modelklassen werden nur über Methoden (Getter- und Setter) vom GUI abgefragt und gesetzt, nie direkt. Die Anweisungen an den Nutzer werden in den Model Klassen erzeugt, jedoch im GUI angezeigt. Im GUI werden keine Bildberechnungen durchgeführt. Für die Punktverschiebung wird vom GUI nur die aktuelle Mausposition weitergegeben, der nächste Punkte wird im Model identifiziert.
- Element welches die Trennung verletzt: Die Model Klassen zeigen die Bilder über „paintComponent()“ selber an (Model und View vereint).
- Um diese Trennung strikt durchzusetzen müsste man die paintComponent() Methode in die Klasse MorphingGUI integrieren. Nachteil: Man müsste ständig die vollständigen Bilder inklusive Punkte zwischen MorphingGUI und ImageContainer sowie MorphingContainer austauschen, das würde die Software langsamer und den Source Code komplizierter machen.

Expertenthema – Bilder erzeugen und darstellen

Frage 4: In dieser Frage geht es um die Handhabung von Objekten aus dem Bereich Bild und Grafik und um häufige Fehler im Umgang mit solchen Objekten. Betrachten Sie dazu den untenstehenden Programmausschnitt, welcher keinen direkten Zusammenhang mit dem Projekt hat. Der Einfachheit halber wurden die Zeilen für diese Aufgabe durchnummeriert.

In den Zeilen 1 und 2 werden zwei Color Objekte erzeugt, welche die Farben Rot und Blau tragen. In den Zeilen 3 und 4 werden zwei leere BufferedImage Objekte mit der Höhe und Breite 10 Punkte erzeugt. „Leer“ bedeutet, dass alle Punkte die Farbe Schwarz tragen. In den Zeilen 5 und 6 werden in beiden BufferdImage Objekten die Punkte an der Koordinate (3,3) blau bzw. rot eingefärbt. Dabei wird über die Methode „public int getRGB()“ der RGB Farbcode

der aktuellen Farbe aus dem Color Objekt als Integer Wert ausgelesen. Die Methode „public void setRGB(int x, int y, int rgbCode)“ färbt im BufferedImage Objekt den Punkt an der Koordinate (x,y) mit dem Farbcode „rgbCode“ ein.

```
1  Color farbeRot = Color.RED;
2  Color farbeBlau = Color.BLUE;
3  BufferedImage bildA = new BufferedImage(10, 10 , BufferedImage.TYPE_INT_RGB);
4  BufferedImage bildB = new BufferedImage(10, 10 , BufferedImage.TYPE_INT_RGB);
5  bildA.setRGB(3, 3, farbeBlau.getRGB());
6  bildB.setRGB(3, 3, farbeRot.getRGB());
7  bildA = bildB;
8  bildA.setRGB(7, 7, farbeBlau.getRGB());
9  bildB.setRGB(7, 7, farbeRot.getRGB());
```

Worin liegt das Problem in den Zeilen 7, 8 und 9? Wie lässt es sich beheben?

Ihre Antwort umfasst vier Teile. Als Erstes **beschreiben** Sie, welche „Bilder“ die Objekte „bildA“ und „bildB“ nach Ablauf der Zeilen 7, 8 und 9 darstellen. Als Zweites **benennen** Sie, das offensichtliche **Problem** in den Zeilen 7, 8 und 9. Als Drittes **identifizieren** Sie **eine** Stelle im Source Code Ihres Expertenteils, wo dieses Problem gelöst wurde. Als Viertes **formulieren** Sie, was passiert wäre, wenn das Problem im Source Code **nicht** behoben worden wäre.

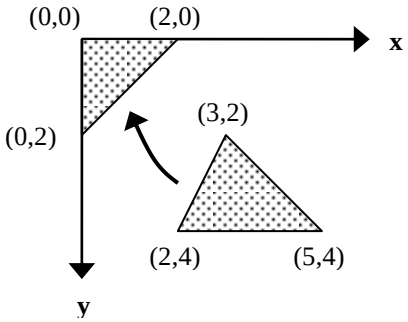
Maximal 4 Punkte: Pro vollständig beantworteten Antwortteil bekommen Sie je 1 Punkt.

Lösung:

- Resultat: Nach den Zeilen 7, 8 und 9 zeigen beide Referenzvariablen bildA und bildB auf dasselbe BufferedImage Objekt im Speicher, d.h. auch auf dasselbe Bild. Dieses zeigt zwei rote Punkte an den Koordinaten (3,3) und (7,7).
- Problem: Die blauen Punkte gehen unter. Der Grund liegt in der Zuordnung des BufferedImage Objekts bildB zur Referenzvariable bildA. Dabei geht das BufferedImage Objekt verloren, auf welches die Referenzvariable bildA ursprünglich verwiesen hat. Ändert man das bildB, so ändert man das bildA.
- Lösung: Damit dies nicht passiert, müssen die Objekte bei einer Zuordnung geklont werden. Beispiel: Klasse MorphingContainer, Methode kopiereBilderZurWeiterenBearbeitung(…), die BufferedImages werden zur weiteren Bearbeitung im MorphingContainer geklont. Ansonsten würden die Bilder bei der Bearbeitung auch in den ImageContainern ändern.

Expertenthema – Affine Transformation

Frage 4: In Ihrem Expertenteil mussten Sie eine Transformationsaufgabe lösen, um zwei Dreiecke deckungsgleich zu machen. In dieser Aufgabe geht es darum, in mehreren einzelnen affinen Transformationen das Dreieck [(3,2), (5,4), (2,4)] in das Dreieck [(0,0)(2,0)(0,2)] überzuführen. Dabei ist es egal, welcher Punkt des originalen Dreiecks auf den Punkt (0,0) zu liegen kommt. Die Situation ist in der nebenstehenden Abbildung verdeutlicht.



Tipp: Verwenden Sie auch die Koordinaten des Zieldreiecks für Ihre Berechnungen. In der Wahl der einzelnen Transformationsschritte gibt es eine Abfolge, welche es erlaubt, alle Parameter aller Transformationsschritte ohne grössere Schwierigkeiten im Kopf auszurechnen. Sie können jedoch eine eigene Abfolge wählen.

Ihre Lösung umfasst zwei Teile. Als Erstes **Skizzen** der einzelnen Lösungsschritte. Fertigen Sie für jeden Transformationsschritt eine **neue** Skizze an. Als Zweites **rechnen** Sie alle **Parameter** der einzelnen Transformationsschritte aus.

Maximal 4 Punkte: Wobei maximal 2 Punkte für den ersten Teil vergeben werden und maximal 2 Punkte für den zweiten Teil.

Lösung:

1. Dreieck [(3,2), (5,4), (2,4)] → translate(-2,-4) → Dreieck [(1,-2), (3,0), (0,0)]
2. Dreieck [(1,-2), (3,0), (0,0)] → rotate(90) → Dreieck [(0,3), (2,1), (0,0)]
3. Dreieck [(0,3), (2,1), (0,0)] → shear(0,-1/2) → Dreieck [(0,3), (2,0), (0,0)]
4. Dreieck [(0,3), (2,0), (0,0)] → scale(0,2/3) → Dreieck [(0,2), (2,0), (0,0)]