Software Engineering II

MandalaBuddy, aktueller Stand (26.11.2023)

Gruppe C

Bauhaus-Universität Weimar

26. November 2023

Aktueller Stand des MandalaBuddys

Grundlegende Programmstruktur:
Unser Programm besteht grundsätzlich aus den Klassen Mandala, Composite, Kreis, Rechteck und der Main(App).

Main:

In der Main wird Hauptsächlich die Oberfläche erstellt. Diese besteht dabei vorrangig aus Labeln, Button und einem Canvas.
Es gibt zwei „Seiten “. Eine Startseite und eine Genratorseite. Die Startseite dient ausschließlich als Startbildschirm und bei Betätigung des Startbuttons wird die Generatorseite aufgerufen.
Auf dieser befinden sich die beiden Label Form und Segmente, sowie die dazugehörige ChoiceBoxen mit entsprechenden Auswahlelementen.
Es gibt zudem noch zwei Button, einen um ein Mandala zu generieren und einen zweiten um ein generiertes Mandala zu speichern:

- Button-Generieren:
  In diesem Button wird zu Beginn ein weißer Hintergrund auf das Canvas gelegt, anschließend ein Mandala generiert, welches auf das Canvas gelegt wird.

- Button-Speichern:
  Dieser Button speichert das Bild des Canvas. Das Problem an dieser Stelle ist, das man das Bild nicht einfach so speichern kann. Zum Speichern nutzen wir an dieser Stelle ein sog. BufferImage. Dabei wird das Canvas mit einem GraphicalContext 'bemalt' und das BufferImage mit einem Graphics2D. Dabei kommt der GraphicalContext aus der JavaFX Bibliothek und Graphics2D aus der AWT Bibliothek. So kommt es, dass zum 'bemalen' der beiden verschiedene Funktionen und Objekte genutzt werden müssen, so werden Farben in JavaFX mit z.B. mit Color.Black festgelegt und in der AWT Bibliothek müssen sie expliziz als solche gekennzeichnet werden mit java.awt.Color.BLACK, es müssen auch zum zeichnen selbst unterschiedliche Funktionen verwendet werden.

Mandala:

Ein Mandala besitzt ein Composite als Attribut. In diesem Composite können verschiedene Formen oder andere Composites, die zusammen das Mandala darstellen, gespeichert werden.

Mit der generate Methode des Mandalas wird ein neues Mandala generiert und auf der Oberfläche gezeichnet. Dafür wird ein neues Composite erstellt, welches dann die Aufgabe des zufälligen generierens übernimmt. Schließlich wird die print Funktion des Composites aufgerufen, um alle Elemente des Mandalas zu zeichnen.

Composite:

Das Composite hat einen Vektor, in dem alle Kinder des Composites gespeichert werden.
Des weiteren hat das Composite einen Durchmesser und einen Mittelpunkt (angegeben in x- und

y-Koordinate) um Composites verschiedener Größen und Anordnung auf dem Canvas darstellen zu können.

Die wichtigsten Methoden des Composites sind die generate Methoden.
Mit der generate Methode können die Formen eines zufälligen Mandalas erstellt werden.
Dafür wird zuerst entschieden, ob das Mandala nur aus dem Mandalakörper bestehen soll oder zusätzlich eine Bordüre haben soll. Des weitern wird bei manchen Mandalakörpern zusätzlich zufällig entschieden, ob sie noch ein Zentrum haben sollen.
Es gibt 5 verschiedene Techniken einen Mandalakörper zu erstellen, 4 verschiedene Bordürenarten und 2 unterschiedliche Zentren. Welche der Techniken genutzt wird wird über die random_body, random_border, random_center Methoden zufällig entschieden.

Des weiteren hat das Composite eine print und eine save Methode. Damit können alle Elemente des Composites auf dem Canvas und auf dem zu speichernden Bild gezeichnet werden.

Kreis und Quadrat:

Ein Kreis und ein Quadrat besitzen als Attribute einen Mittelpunkt (angegeben in x- und y-Koordinate), einen Radius, der beim Quadrat den Abstnad zwischen Mittelpunkt und Ecke beschreibt, eine Farbe (einmal AWT und einmal JavaFX) und einen Wahrheitswert, der beschreibt, ob das Objekt transparent ist oder nicht.

In der print Methode wir das Objekt mit den in den Attributen festgelegten Eigenschaften auf den GraphicContext des Canvas gezeichnet. Dabei müssen für die Zeichenfunktionen die x- und y-Koordianten der linken oberen Ecke der BoundingBox des Objektes und der Durchmesser des Kreises bzw. die Kantenlänge des Quadrates berechnet werden.
Die save Methode funktioniert identisch, nur wird hier auf dem GraphicContext für das zu speichernde Bild gezeichnet.

Möglichkeiten der Erweiterung:

- weitere Formen zum Zeichnen hinzufügen
- Möglichkeit einer farblichen Veränderung hinzufügen
- weitere Muster der Mandalaerstellung hinzufügen

Main:
In der Main befindet sich die grundsätzliche Oberfläche des Programms. Die Hintergründe der Oberfläche befinden sich im Ordner Ressourcen und der Pfad ist entsprechend angelegt.
Des weiteren besteht unser Oberfläche aus Labels, Buttons und ChoiceBoxen.
Es gibt zwei relevante Buttons.

Speichern-Button: Hier wird der erstellte GraphicalContext in einen PixelWriter kopiert, von welchem ein Snapshot erstellt wird, welcher widerum in einen File namens 'Mandala made by generator' kopiert und so im Dateiformat .png gespeichert wird.

Generate-Button: Innerhalb dieses Buttons werden die Nutzerauswahlen der ChiceBoxen ausgelesen und es wird ein Mandala erstellt, an welches diese Daten übergeben werden. Zum Schlusss wird an dieser Stelle der GraphicalContext in welchem das Mandala gezeichnet wurde an das Canvas übergeben und so graphisch dargestellt.

Es gibt zudem 3 ChoiceBoxen sowie 3 zugehörige Label(first item, second item, third item)
In diesen wird durch den Nutzer eine Auswahl getroffen, wie seine Mandalas erstellt werden sollen.
Wir haben bis jetzt 2 DummyChoiceBoxen erstellt (Farbe, Formen). Diese sind noch nicht bei der Mandalaerstellung implementiert.
Momentan wird ausschlieslich Sections, auch tatsächlich bei der Mandalaerstellung verwendet.

Der Kern stellt ein leeres Canvas dar.

Mandala:
Diese Klasse besitzt einen Konstruktor, welcher bei der Erstellung aufgerufen wird. Diesem muss ein GraphicsContext, die Sections, die Farbe sowie die Form aus welcher das Mandala aufgebaut werden soll, übergeben werden muss. Ein Mandala hat einmal die Attribute Farbe, Form, Sections und besitzt ein Komposit (siehe SE1)

Das Mandala besitzt eine print()-Funktion mit der die Print-Funktion des eigenen Composites aufgerufen wird.

Composite:
Jedes Komposit besitzt einen Vektor mit Elementen. Das können Kreise, Rechtecke und weitere Composites sein.
Die Klasse besitzt eine Printfunktion, welche die Printfunktionen der im Vektor gespeicherten Objekte aufgerufen wird.

Kreis:
Die Kreis-Klasse besitzt die Attribute Radius, Farbe und Mittelpunkt. Sie besitzt auch eine Print-Funktion, welcher erneut der GraphicalContext übergeben werden muss. Bei Aufruf wird in den GraphicalContext mit der Funktion StrokeOval ein kreis mit den gegebenen Variablen in den GraphicalContext gezeichnet.
Das Problem beim zeichnen des Kreises ist, das Kreise auf Basis von Boxen erstellt werden. Die StrokeOvalFunktion benötigt einen Eckpunktder Box in welchem der Kreis gezeichnet werden soll, und die Länge und Höhe der Box. Über ein paar einfache Modifikationen ist es durch die

Modifizierte Eingabe, aber möglich den mit dem Mittelpunkt der Referenz zu arbeiten. (das findet ausschließlich in der StrokeOvalFunktion statt und erfordert keine weiteren Funktionen)

Rechteck:
Das Rechteck besitzt noch keine nennenswerten Attribute und Funktionen.