|  |
| --- |
| Studieneinheit / Modul: Coding - Projektarbeit (Modulabschlussprüfung) |
| Studiengang: Elektro- und Informationstechnik (B. Eng.) (1648.01) |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Prüfung: | Coding - Projektarbeit (Modulabschlussprüfung) (CODI PAxx) | | Abgabedatum: | 14.07.2025\* | | Name: | **Tim Gassan** | | Matrikelnummer: | **910071405** | |

**Bitte berücksichtige die folgenden allgemeinen Hinweise!**

**Vorbemerkung:**

Die Projektarbeit ist bis zu dem oben genannten Termin einzureichen. Verspätete oder falsche eingereichte Lösungen können nicht gewertet werden.

Sende die vollständige Aufgabenstellung und Lösung über den *Online-Campus an der gleichen Stelle ein, an der du die Aufgabe angefordert haben.* **Die Projektarbeit muss zwingend im Format Microsoft Word (.docx) erstellt und eingereicht werden.** Du darfst bei der Projektarbeit aber auch eine **zusätzliche** PDF-Datei einreichen.

Sollten in der Aufgabenstellung weitere Dateien gefordert werden, darfst du das in der Aufgabenstellung genannte Format zusätzlich einreichen. Grundsätzlich sind aber nur die Formate PDF (.pdf), Microsoft Excel (.xls/.xlsx), MP3 (.mp3) oder Powerpoint (.ppt/.pptx) möglich. Bis maximal drei Dateien sind möglich.

Sollte es Abweichungen der Einreichform geben, findest du diese Hinweise in der Aufgabenstellung.

Bitte berücksichtige, dass bei Einreichung der Lösung zunächst alle Dateien ausgewählt werden müssen, bevor der Button „Lösung versenden“ betätigt wird.

Die Lösung darf nur eine maximale Größe von 8 MB haben.

Sollte es eine Vorgabe zum Umfang der Wortanzahl in Ihrer Aufgabenstellung geben, gilt für dich folgendes: Wortzahlüberschreitungen sind maximal in Höhe von 10% der vorgesehenen Wortzahl möglich. Wird die vorgegebene Wortzahlbegrenzung zuzüglich 10% Toleranzgrenze überschritten, führt dies zu einem pauschalen Punktabzug in Höhe der Prozentzahl der Wortzahlüberschreitung.

Bitte beachte die „Checkliste zur Selbsteinschätzung eines Forschungsprojekts auf ethische Unbedenklichkeit“. Sofern es sich bei der vorliegenden Arbeit um ein empirisches Forschungsvorhaben mit menschlichen Probandinnen und Probanden handelt, musst du die in der Checkliste genannten Anforderungen beachten. Die weiteren Hinweise hierzu findest du im Download Bereich auf dem Online Campus.

**Hinweis zur Bewertung:**

Die Prüfungs- oder Studienleistung ist bestanden, wenn sie mindestens mit ausreichend (4,0) oder besser bewertet wurde.

**Hinweise zur Bearbeitung:**

Solltest du für deine Lösung externe Quellen hinzuziehen oder fremdes Gedankengut verwenden, so ist dieses kenntlich zu machen. Die besonderen Hinweise zu dieser Projektarbeit findest du ggf. über der Aufgabenstellung

Bei Literaturempfehlungen kann nicht durchgängig gewährleistet werden, dass diese jederzeit kostenfrei über die von uns angebotenen Bibliotheken verfügbar sind. Im Regelfall kannst du die Literatur auch an anderen Stellen, etwa Online-Bibliotheken, beziehen. Grundsätzlich bist du frei in der Auswahl der für die Bearbeitung der Hausarbeit geeigneten Literatur, sodass du auch in den bereit gestellten Online-Bibliotheken alternative und ggf. kostenfreie Literatur heraussuchen können.

**Viel Erfolg!**

**Aufgabe:**

Erstellen Sie mit Java und Swing ein vollständig im Sinne der folgenden Aufgabenstellung lauffähiges Grafikprogramm mit grafischer Oberfläche. Das Programm soll über eine Menü- und Symbolleiste zur Auswahl von Funktionen und einen Bereich für das Erstellen von Grafik verfügen.

Es soll mindestens folgende Funktionen unterstützen:

* Zeichnen von Ellipsen, Linien und Rechtecken mit der Maus,
* Erstellen neuer Dateien,
* Laden von Dateien und
* Speichern von Grafiken.

Ergänzen Sie außerdem noch mindestens drei weitere Funktionen Ihrer Wahl – zum Beispiel eine Farbauswahl für die grafischen Elemente oder grafische Werkzeuge wie ein Radiergummi.

Stellen Sie den Nutzenden für das Laden und Speichern von Dateien entsprechende Dialoge zur Verfügung. Sie sollen nur Dateien vom Typ JPG anzeigen. Für die Dialoge können Sie die Klasse JFileChooser verwenden. Filter für die Dialoge erstellen Sie über die Klasse FileFilter.

Für den Bereich zum Erstellen von Grafiken können Sie eine Komponente vom Typ BufferedImage verwenden. Für die Zeichenoperationen beschaffen Sie sich den Grafikkontext vom Typ Graphics2D für die Komponente vom Typ BufferedImage. Zum Laden und Speichern von Grafiken können Sie die Klasse ImageIO verwenden. Zum Verarbeiten von Mausereignissen verwenden Sie die Klasse MouseAdapter.

**Bitte beachten Sie bei der Umsetzung folgende Punkte:**

Stellen Sie den Nutzenden Tastaturkommandos für das Erstellen neuer Dateien sowie für das Laden und Speichern von Dateien zur Verfügung. Auch die Bedienung des Menüs soll über die Tastatur möglich sein. Benutzen Sie dabei übliche Kürzel beziehungsweise Tastenkombinationen.

Verwenden Sie für die Symbolleiste übliche Symbole und versehen Sie die Symbole mit Tooltipps. Eine Sammlung mit Symbolen finden Sie zum Beispiel im Internet auf der Seite https://www.oracle.com/java/technologies/java-look-and-feel-graphics-repository.html.

Vor dem Anlegen einer neuen Datei soll eine Abfrage erscheinen, ob wirklich eine neue Datei angelegt werden soll.

Achten Sie auf die Betriebssicherheit des Programms – zum Beispiel beim Laden und Speichern von Dateien.

Beschreiben Sie ausführlich und fachlich Ihr Vorgehen beim Erstellen des Programms. Kommentieren Sie umfassend Ihren Code. Beachten Sie bei der Umsetzung Vorgaben wie Clean Code und führen Sie systematische Tests Ihres Programms durch.

**Reichen Sie 2 Dateien als Worddatei ein:**

1. Projektdokumentation (Vorgehensbeschreibung, Qualitätssicherungs- und Testdokumentation)
2. Vollständiger, kommentierter Java-Code ohne jegliche zusätzlichen Texte oder Elemente, der reine Code.

**Projektdokumentation:**

1. **Vorgehensbeschreibung**

Das Abschlussprojekt ist ein Zeichenprogramm nach obiger Aufgabenstellung. Um dieses realisieren zu können bestand der erste Schritt darin, sich einen groben Überblick über die bis dahin unbekannten Klassen für grafische Tools sowie das Laden und Speichern von Bildern zu verschaffen. Im gleichen Zeitrahmen wurde ein Gradle-Projekt in IntelliJ und ein zugehöriges Git-Repository erstellt.

Anschließend ging es an die Planung des Projekts um die Frage zu beantworten, welche Funktionen das Programm im finalen Zustand haben soll. Die detaillierte Auflistung ist in der Qualitätssicherungsdokumentation zu finden. Es wurde zu großen Teilen mit Klassen gearbeitet werden, von denen die möglichen Funktionen erst im Laufe des Programmierens entdeckt wurden. Daher ist im Laufe des Projekts immer wieder zu diesem Schritt zurückgekehrt worden, um das Programmziel an das neue Wissen anzupassen. Dabei wurde darauf geachtet, dass der Umfang des Projekts nicht mit immer neuen Funktionen zu groß wurde, sondern im ursprünglich geplanten Rahmen bleibt.

Im nächsten Schritt wurde die Struktur des Programms festgelegt. Dabei wurde sich für eine Aufteilung in drei Klassen entschieden. Die Klasse „PaintProgram“, welche die Main-Funktion enthält, die Klasse „Frame“, die das Fenster des Programms abbildet und die Klasse „PaintPanel“, die die Zeichenfläche darstellt. Zu diesem Zeitpunkt wurde auch festgelegt, dass das Programm in Englisch gecodet wird, mit der Ausnahme von sichtbarem Text im Programm – es ist für einen deutschen Anwender programmiert – und Kommentaren.

Der nachfolgende Teil hat sich mit dem eigentlichen Programmieren beschäftigt. An dieser Stelle soll eine kurze Zusammenfassung der implementierten Funktionen folgen, konkrete Hinweise zum Code stehen in den Kommentaren. Es wurde das JFrame und das darauf abzubildende JPanel erstellt und die Menü- und die Symbolleiste zunächst mit Platzhaltern erstellt. Anschließend wurden die großen Funktionen wie die Zeichenwerkzeuge, die Farbauswahl und die Strichdickenauswahl hinzugefügt. Es wurden dann die Funktionen zum Erstellen neuer Zeichenflächen erstellt, die mit Sicherheitsabfrage entweder direkt ein leeres Blatt in gleicher Größe oder über Benutzerabfrage ein leeres Blatt mit neuer Größe erstellen. Es kamen die Funktionen zum Speichern- und Laden der Dateien hinzu, die standardmäßig im Projektordner zu finden sind. Die Funktionen haben ihre Hotkeys und Tooltips bekommen und es wurden Scrollbars bei zu großen Zeichenflächen hinzugefügt.

Im nächsten großen Schritt wurden die Feinheiten bearbeitet. Dazu gehören zum Beispiel das Layout der Symbolbar sowie die Icons der Buttons, die bis dahin noch Platzhalter waren. Das Markieren der Buttons, wenn sie angewählt sind ist dazugekommen und als Abschluss eine eigentlich große Funktion, die bis zum Ende nicht sicher in der finalen Version enthalten war, weil nicht sicher war, wie aufwendig sie sein würde. Letztendlich hat sich die Implementation der Vorschaubilder beim Ziehen von Linien, Ellipsen und Rechtecken als machbar erwiesen. Währen dieses gesamten Prozesses wurde darauf geachtet, nach möglichst kleinen, vollständigen Änderungen einen neuen Commit und Push auf GitHub auszuführen, um bei Problemen möglichst sicher und flexibel zu sein.

Im letzten großen Schritt stand das Testen des Programms auf dem Plan. Zunächst wurde sich bewusst dazu entschieden, auf das Schreiben von automatisierten Tests während der Programmierung zu verzichten, da durch neue Erkenntnisse über die grafischen Klassen Programmteile immer wieder Gefahr liefen, überarbeitet zu werden. Das hätte bei automatisierten Tests zu Mehrarbeit geführt und gleichzeitig ist dieses Projekt vom Umfang her vergleichsweise klein, so dass das Testen ans Ende verschoben werden konnte. Im zweiten Punkt gibt es nur eine Methode im Projekt, die mit den bekannten Mitteln automatisiert getestet werden kann. Für das automatisierte Testen des grafischen Teils dieses Programms wären extra Werkzeuge wie z. B. QF-Test notwendig. Auch an dieser Stelle wurde aus Zeitgründen darauf verzichtet. Stattdessen wurden die weiter unten dokumentierten Testfälle manuell durchgeführt. Für ein größeres Projekt sei angemerkt, dass dieses Vorgehen nicht zu empfehlen ist. Wenn das Programm öffentlich gemacht werden sollte, würden sich an dieser Stelle weitere Anwendungstests mit mehreren Testpersonen empfehlen, um Feedback zu erhalten und das Programm abzurunden.

Die beim Testen gefundenen Fehler wurden behoben und bei Bedarf, neue Testfälle hinzugefügt, um ein möglichst vollständiges Bild vom Zustand des Programms zu enthalten.

Der Abschluss bestand darin, diese Dokumentation zu verfassen und die Qualitätssicherungs- und Testdokumentation zusammenzufassen.

1. **Qualitätssicherungsdokumentation**

Um die Qualität dieses Programms sicherzustellen sollen an der Stellen auf folgende Punkte eingegangen werden: die Entwicklungsrichtlinien, die Qualitätskriterien, die Versionskontrolle und das Testen.

2.1 Entwicklungsrichtlinien

In diesem Projekt sind folgende Entwicklungsrichtlinien zu Beginn definiert worden, die dann im Laufe des Projekts eingehalten werden sollten:

* Variablennamen werden beschreibend in Englisch gewählt.
* Es werden beschreibende Konstanten für Strings (z. B. Actioncommands) verwendet, wenn diese mehr als zwei- bis dreimal verwendet werden, um Tippfehler zu vermeiden.
* Das Programm soll so geschrieben werden, dass es möglichst einfach wartbar und erweiterbar ist. Dazu gehört, dass sich wiederholende Abläufe wenn möglich in einer Methode zusammengefasst werden und das Codeteile, die zu den gleichen Objekten gehören (z. B. das Setzen verschiedener Eigenschaften eines Buttons) auch örtlich beieinander stehen.
* Das Programm wird detailliert auf Deutsch auskommentiert
  1. Qualitätskriterien

Hier sollen die konkreten Anforderungen an das Zeichenprogramm dargelegt werden, um sowohl dem Programmierer ein klares Ziel zu geben, sowie das fertige Produkt daran messen zu können. An dieser Stelle kann davon ausgegangen werden, dass diese Kriterien erfüllt wurden, wenn keine Anmerkungen in Klammern hinter diesen stehen.

* Die Oberfläche besteht aus einer Menü- und Symbolleiste sowie einer grafischen Fläche zum Zeichnen.
* Die Menüleiste enthält übliche Reiter und kann mit der Tastatur bedient werden
* In der Menüleiste sind die Menüs Datei, ~~Bearbeiten, Ansicht~~ und Werkzeuge zu finden. (Das Menü „Bearbeiten“ war für die Funktion „Rückgängig machen“ gedacht und das Menü „Ansicht“ war ursprünglich für das Anpassen der Zeichenfläche gedacht und nicht mehr benötigt)
* Im Menü Werkzeuge sind die Zeichentools zusätzlich zur Symbolleiste zu finden.
* Im Menü Datei kann ein neues Zeichenblatt erstellt, ein Bild geladen, das aktuelle Bild gespeichert und das Programm beendet werden. Bis auf das Beenden des Programms sind diese Funktionen mit Hotkey ausführbar.
* Beim Erstellen eines neuen Zeichenblattes, beim Laden einer Datei und beim Beenden soll der Benutzer gefragt werden, ob er diese Aktion wirklich durchführen möchte.
* Beim Erstellen eines neuen Zeichenblattes soll der Benutzer die Option haben, die Größe des Zeichenblattes zu ändern
* ~~Im Menü Bearbeiten soll die Funktion „Rückgängig machen“ sein, mit der die letzte Aktion auf der Zeichenfläche zurückgenommen werden kann.~~ (Funktion gestrichen, da die Implementierung zu aufwändig gewesen wäre.)
* Die Symbolleiste enthält die Werkzeuge zum Zeichnen, die mit Icons und Tooltips zeigen, wofür sie gedacht sind.
* Die Werkzeuge sind über nachvollziehbare Hotkeys auswählbar.
* Die Zeichenwerkzeuge sind: ein Pinsel zum freien Zeichnen, Werkzeuge zum Zeichnen von Linien, Ellipsen und Rechtecken und ein Radierer zum freien Radieren. Die Werkzeuge für die Figuren sollen dabei beim Ziehen mit gedrückter Maustaste eine Vorschau anzeigen und in alle vier Richtungen gezogen werden können.
* Zum Arbeiten auf der Zeichenfläche soll die linke Maustaste verwendet werden.
* Wenn beim Ziehen mit einem der Figurenwerkzeuge die Zeichenfläche verlassen wird, soll die Figur sich trotzdem entsprechend auf der Zeichenfläche weiterverändern und gezeichnet werden, wenn die Maustaste losgelassen wird.
* Für alle Werkzeuge soll die Strichdicke mittels Textfeld und die Farbe über entsprechende Buttons eingestellt werden können. Diese Einstellung soll beim Wechsel des Werkzeugs erhalten bleiben. Ausnahme davon ist der Radierer. Bei ihm soll nichts passieren, wenn eine Farbe ausgewählt wird. Die Größe des Radierers bleibt einstellbar.
* Die Buttons sollen dem Benutzer anzeigen, welches Werkzeug bzw. welche Farbe ausgewählt ist.
  1. Versionskontrolle

Während des Programmierens wurde regelmäßig das Repository aktualisiert und auch auf GitHub gepusht. Es wurde darauf geachtet, dass alle benötigten Dateien für das Programm im Repository sind, alle nicht benötigten Dateien z. B. der Ordner „savedPictures“ als Standardordner für das Laden und Speichern von Bildern aber nicht enthalten sind.

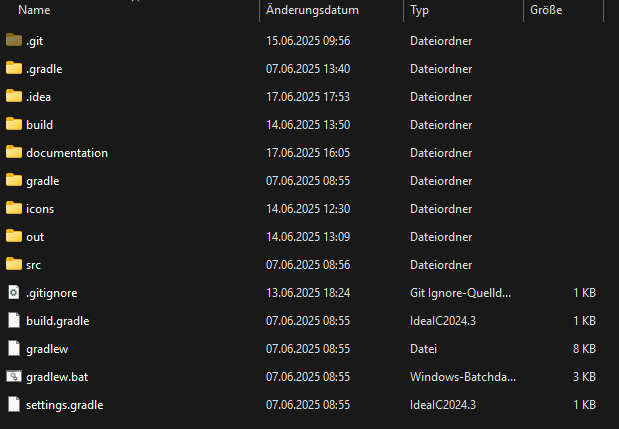


Abbildung 1 Projektordner mit allen Dateien und Ordnern

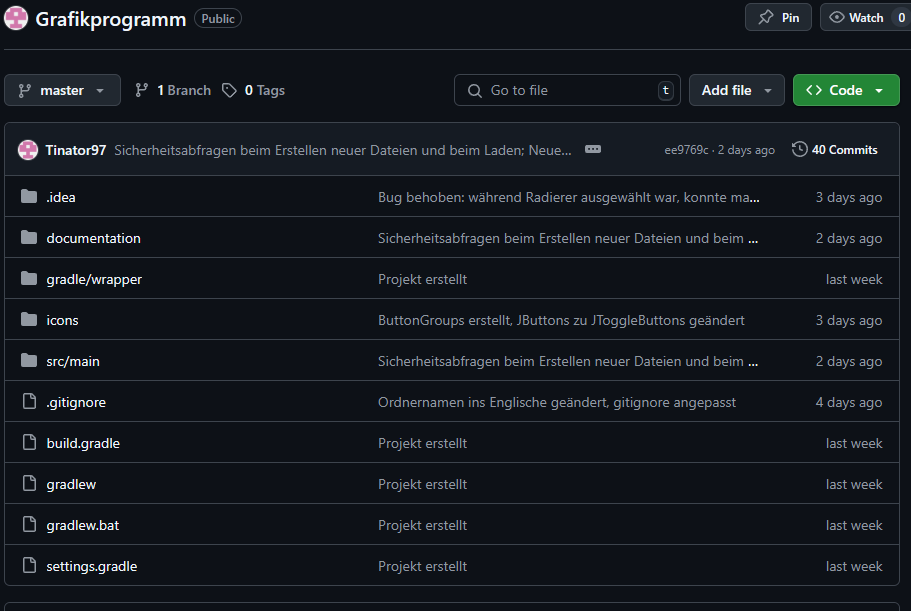


Abbildung 2 Auf GitHub hochgeladene Dateien und Ordner

Im Folgenden ist die ist die Push-Historie vom Beginn des Projekts bis zur Fertigstellung zu sehen.

…

* 1. Testen

Das Testen nimmt einen wichtigen Platz ein, um sicherzustellen, dass das Programm allen Anforderungen genügt. Wie oben bereits beschrieben, wurde das Testen in diesem Projekt vornehmlich manuell und erst nach Abschluss des eigentlichen Programmierens vorgenommen.

Die Methode PaintPanel.checkOrientation ist die einzige Methode – abgesehen von Gettern – die einen anderen Rückgabewert als void hat und damit mit unseren Mitteln automatisiert getestet werden kann. Das ist im entsprechenden Ordner „src/test“ zu finden. Für alle anderen Methoden ist grafisches Testen nötig, das mit speziellen Programmen auch automatisiert möglich wäre, in diesem Projekt aber manuell erledigt wird.

Für alle Tests wurden entsprechende Grenzwerte und besondere Testfälle für Blackbox-Tests ausgewählt und weitere Testfälle für Whitebox-Tests hinzugefügt. Damit sollte jede Methode vollumfänglich getestet sein. Die genauen Tests mit ihren konkreten Testfällen sind im nächsten Abschnitt dokumentiert.

1. **Testdokumentation**