HAUSARBEIT

Programmierung Grafischer Benutzeroberflächen

WINAPI-Programmierung in C/C++ und Java

Prof. Dr. D. Rosenthal
GUI

Leonel Nguimatsia Tsobguim
11142206

1. Einführung

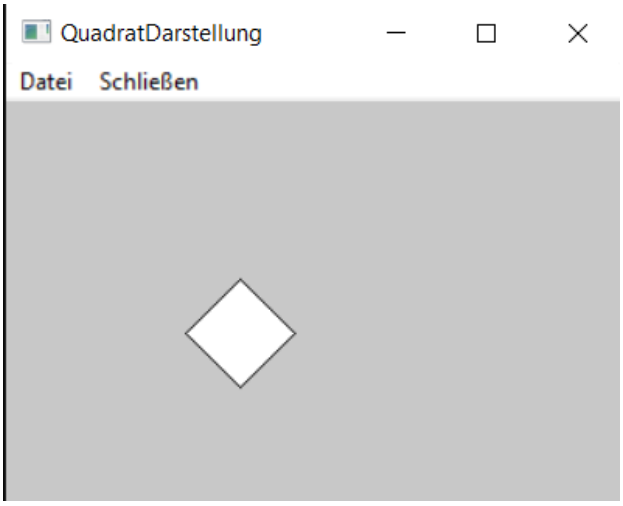
Grafische Benutzeroberfläche (Abk. **GUI** von englisch *graphical user interface*) bezeichnet eine Form von Benutzerschnittstelle eines Computers. Sie hat die Aufgabe, Anwendungssoftware auf einem Rechner mittels grafischer Symbole, Steuerelemente oder auch Widgets genannt, bedienbar zu machen. Dies geschieht bei Computern meistens mittels einer Maus als Steuergerät, mit der die grafischen Elemente bedient oder ausgewählt werden, bei Smartphones, Tablets und Kiosksystemen in der Regel durch Berührung eines Sensorbildschirms.

Die Gesamtgestaltung heutiger grafischer Oberflächen verwendet oftmals die sogenannte Schreibtischmetapher. Dieses Konzept wurde ab 1984 mit dem Macintosh von Apple populär, in den 1990er Jahren entwickelte es sich zum Industriestandard bei Personal Computern.

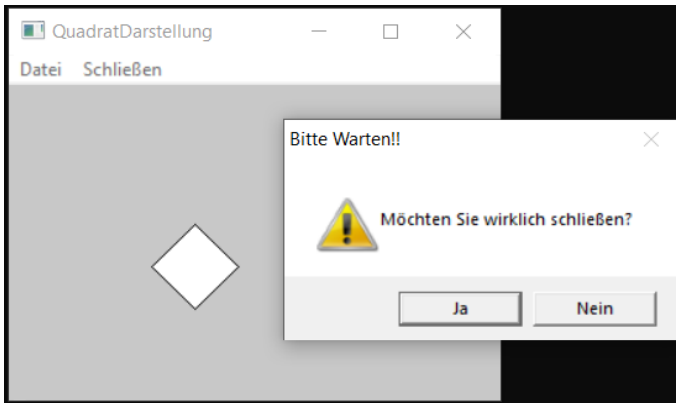
Dieses Projekt beschäftigt sich mit den Programmiergrundlagen grafischer Benutzeroberflächen speziell in Java (AWT oder Swing) und C/C++ (WINAPI-Programmierung). Dazu habe ich zwei minimalistischen Beispielprogramme implementiert.

2. WINAPI-Programmierung in C/C++

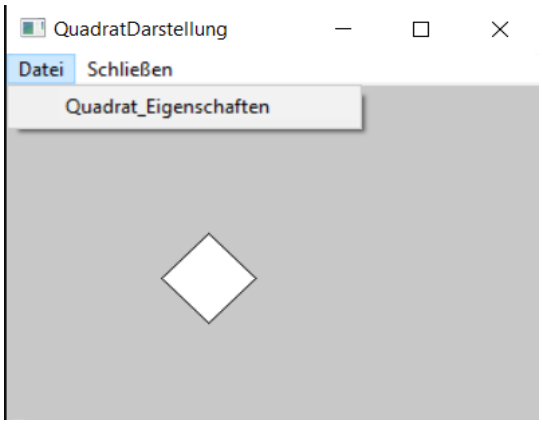
Hier geht es um die Darstellung eines Quadrates im Hauptfenster, von denen die Seitenlänge des Quadrats mit Hilfe eines Dialogfenster in Editfeld eingegeben wird. Dazu hat das Dialogfenster weiterhin anderen Eigenschaften, von denen ein Schiebregler (Scrollbar) zu realisieren ist und mit Hilfe der Drehung des Quadrates seinen Mittelpunkt ausgeführt wird.

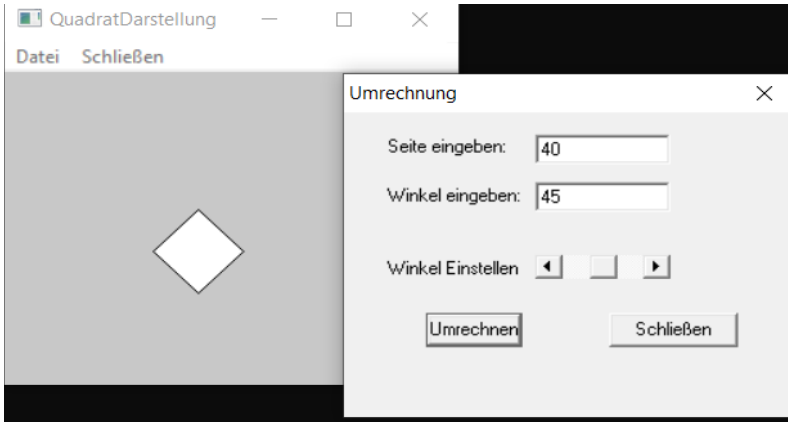


Beim Programmstarten hat man ein minimales Hauptfenster mit dem Titel **QuadratDarstellung** und einem Menu **Datei** und einem Menu Item **Schließen**. Dazu wird das Hauptfenster direkt mit einer Seitenlänge des Quadrates von 40 und einen Winkel von 45° initialisiert. Dies ist auf dem folgenden Bild zu sehen.



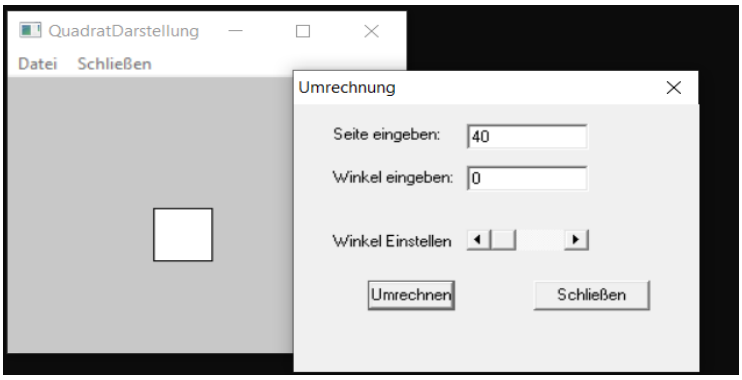
Wenn der Benutzer auf dem Menu Item **Schließen** klickt, kommt sofort eine MessageBox mit einer Nachricht, um zu fragen, ob der Benutzer wirklich das Programm schließen möchte. Bei Ja wird das Programm sofort geschlossen und bei Nein kann den Benutzer weitere Eingabe machen. Dies ist genauso dasselbe, wenn man auf dem **„X“** Zeichnen der Titelleiste bestätigt.



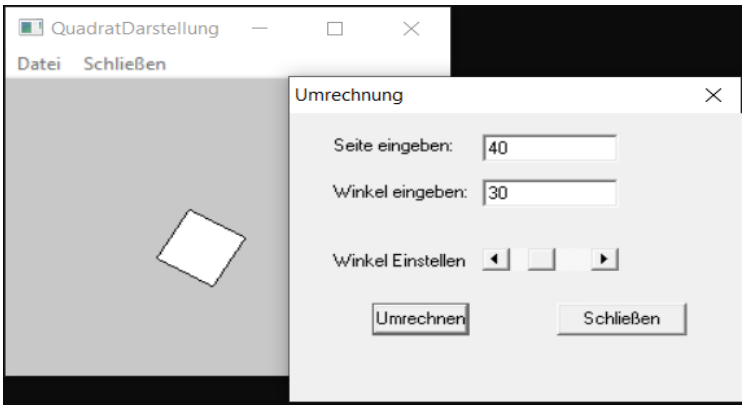


Hier folgt sich das Programm und das POPUP Menu Datei hat nur einen Menu Item **Quadrat_Eigenschaften**. Wenn der Benutzer darauf klickt, öffnet sich das Dialogfenster mit den Initialisierten Werten (Seitenlänge = 40 und Winkel = 45°) und den folgenden Eigenschaften des Quadrates:

- Zwei **Editbox** mit den Beschriftungen, wo man die Seitenlänge und Winkel des Quadrates eingeben kann.
- Ein **Scrollbar** schon beschriften für die Einstellung des Winkels
- Zwei Buttons **Umrechnen und Schließen.**
  - Mit **Umrechnen bestätigt** wird die Eigenschaften des Quadrates umgerechnet.
  - Mit **Schließen** wird das Dialogfenster geschlossen.



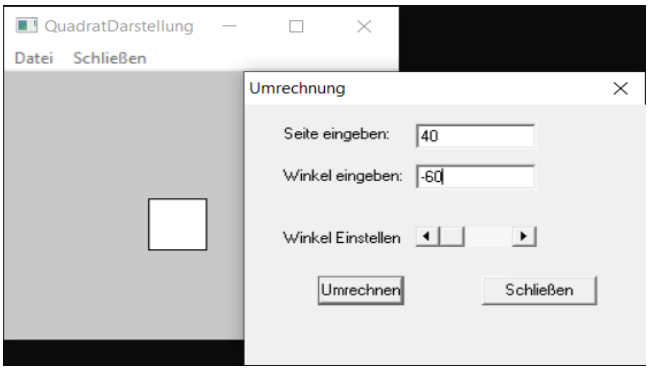
Erste Beispiel:
Ein Quadrat mit einer Seitenlänge von 40 und einen Drehwinkel von 0°

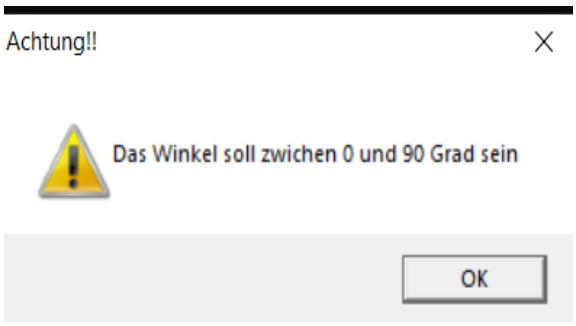


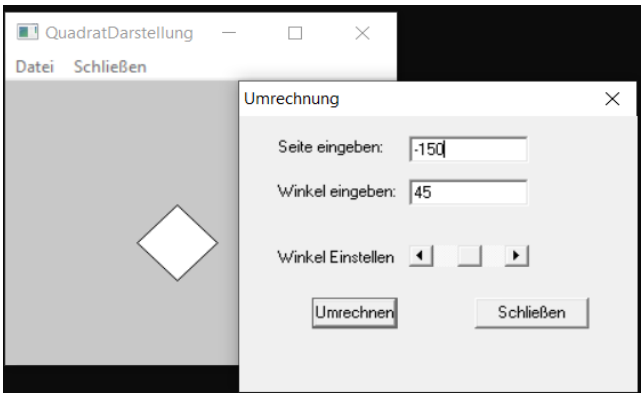
Zweite Beispiel:
Ein Quadrat mit einer Seitenlänge von 40 und einen Drehwinkel von 30°

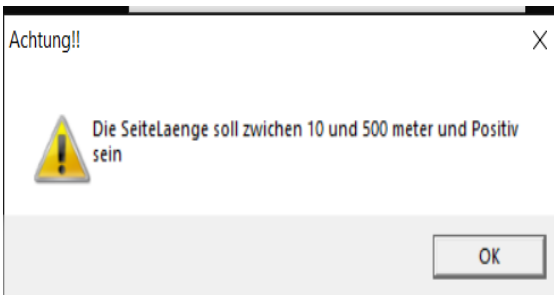
Dritte Beispiel:
Ein Quadrat mit einer Seitenlänge von 40 und einen Drehwinkel von 60°

Was es hier am wichtigsten steht, ist die Fehlermeldung bei falscher Eingabe des **Winkels** beziehungsweise die **Seitenlänge** des Quadrates, wird durch eine MessageBox angezeigt. Damit die Programm-code sinnvoll ist, habe ich mich für eine bestimmte Intervall zur Seitenlänge des Quadrates entschieden. Dies ist zwischen 10 und 500, damit die Drehung und die Eigenschaften des Quadrates besser aussehen. Diese lässt sich durch die folgenden Bilder sehen:





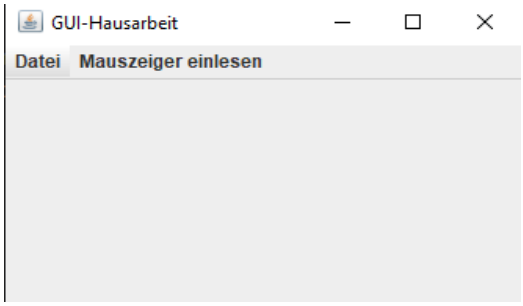




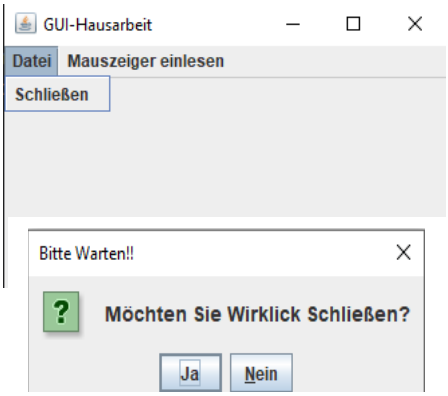
3. WINAPI-Programmierung in Java Swing

In diesem zweiten Teil der Programmieraufgabe geht es um die Erstellung ein WINAPI-Programm in Java, von dessen Mauszeigers und Titels des Hauptfensters einer Windowsapplikation dynamisch geändert werden sollen.

.

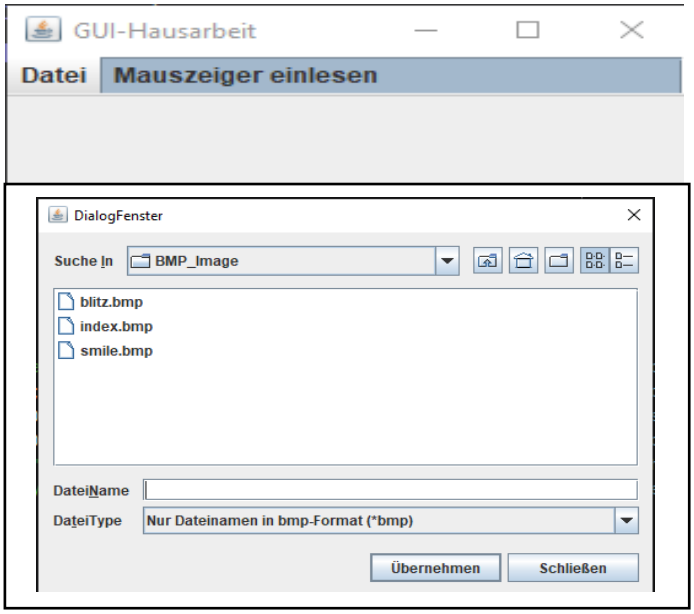


Das Programm start mit einem minimalen Hauptfenster, von dessen **GUI-Hausarbeit** den Titelleiste ist. Dazu besitzt das Hauptfenster ein Menu Datei und ein Menu Item **Mauszeiger einlesen**.



Hier hat der Menu **Datei** ein Menu Item **Schließen** und wenn der Benutzer auf dem Menu Item **Schließen** klickt, kommt sofort eine MessageBox mit einer Nachricht, um zu fragen, ob der Benutzer wirklich das Programm schließen möchte. Bei Ja wird das Programm sofort geschlossen und bei Nein kann den Benutzer weitere Eingabe machen. Dies ist genauso dasselbe, wenn man auf dem **„X“** Zeichnen der Titelleiste bestätigt.

Hier beginnen wir mit dem Kern des Programms selbst. Eine sehr häufige Verwendungsart für ein Dialogfenster, von denen Speicherung und Öffnen **usw.** der Dateien möglich ist, heißt ein Objekt der Klasse **JFilechooser**. Ich habe das Objekt dieser Klasse verwendet, um ein einfaches Dialogfenster auf dem Menu Item **Mauszeiger einlesen** zu integrieren. Dies sieht wie folgt aus:

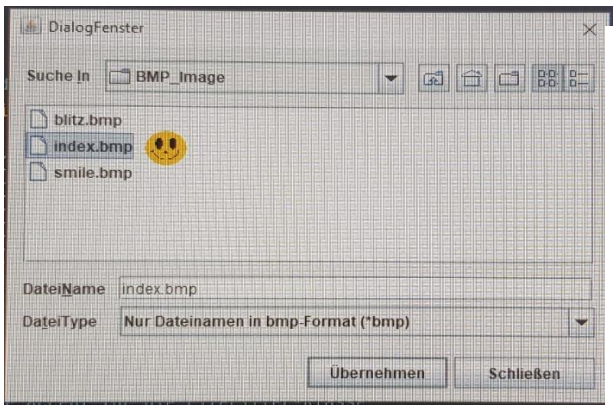


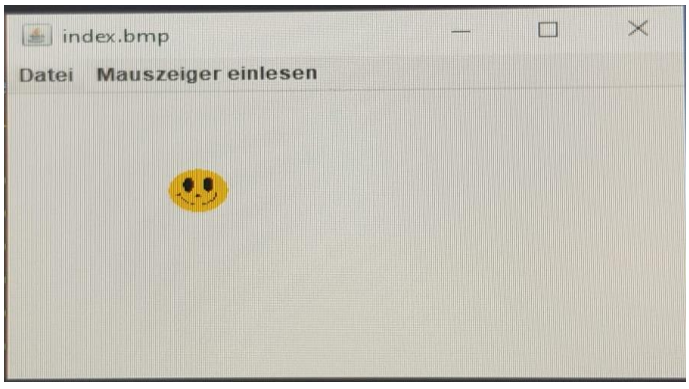
Wenn der Benutzer darauf klickt, öffnet sich das Dialogfenster mit den aktuellen Werten und den folgenden Eigenschaften des Dialogfensters:

- Im Verzeichnis **BMP_Image** (Das aktuelle Verzeichnis) gibt es drei bmp-Bilder.
- Die Label **Dateiname** mit dem **Editbox** erwartet das eingelesene Bild.
- Die Label **DateiType** mit einer ausgewählten Liste von Dateien gibt nur die verfügbaren Dateien. Auf diesem Fall hat man nur die bmp-format zur Verfügung.
- Zwei Buttons **Übernehmen** zur Anpassung der Mauszeiger und **Schließen**, um Das Dialogfenster genau zu schließen.

<u>Beispiel mit Index.bmp-Bild:</u>

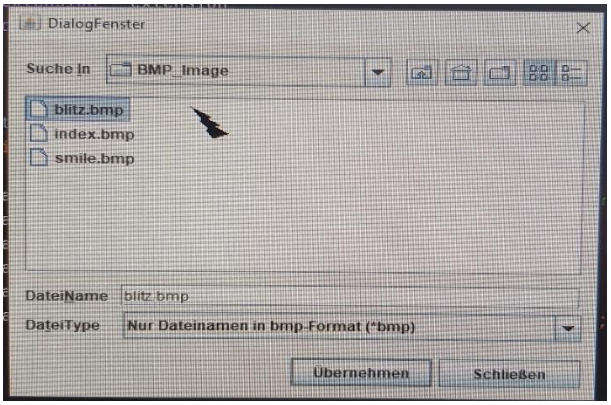
Wenn man auf dem Bild **Index.bmp** klickt, ist das Bild im Editbox **Dateiname** eingelesen und gleichzeitig wird den Mauszeiger mit dem entsprechenden Bild angepasst.
Durch einen Klick auf dem Button **Übernehmen** wird der neue Mauszeiger auch im Hauptfenster verwendet und der Dateiname für den Mauszeiger wird gleichzeitig in der Titelleiste des Hauptfensters angezeigt.

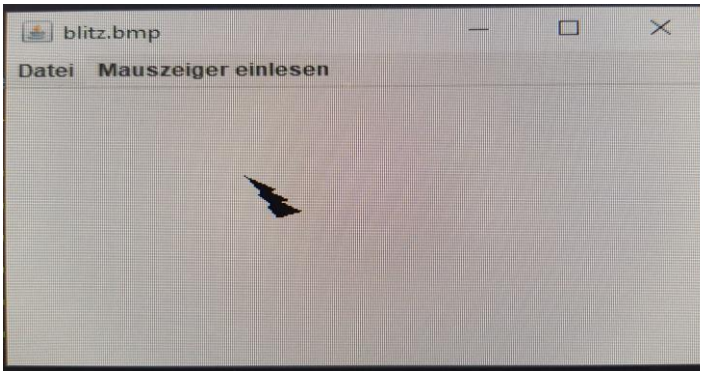




<u>Beispiel mit Blitz.bmp-Bild:</u>

Wenn man auf dem Bild **Blitz.bmp** klickt, ist das Bild im Editbox **Dateiname** eingelesen und gleichzeitig wird den Mauszeiger mit dem entsprechenden Bild angepasst.
Durch einen Klick auf dem Button **Übernehmen** wird der neue Mauszeiger auch im Hauptfenster verwendet und der Dateiname für den Mauszeiger wird gleichzeitig in der Titelleiste des Hauptfensters angezeigt.

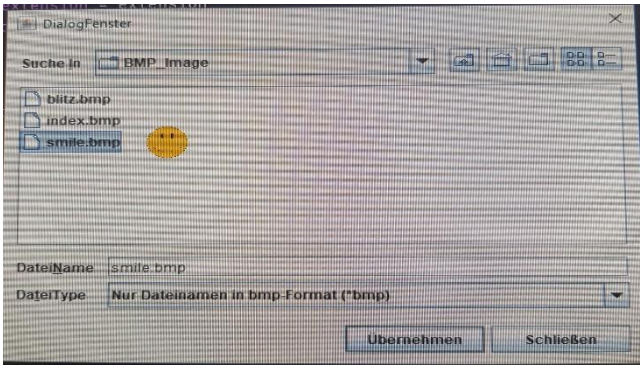


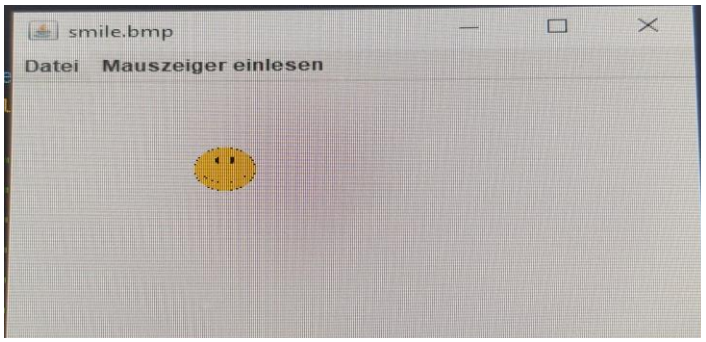


Beispiel mit smile.bmp-Bild:

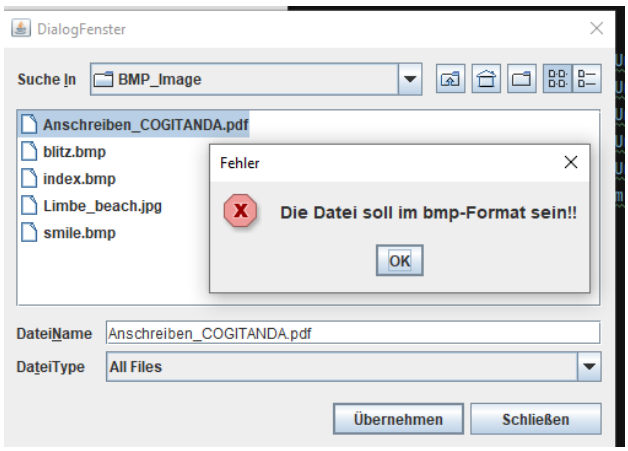
Wenn man auf dem Bild **smile.bmp** klickt, ist das Bild im Editbox **Dateiname** eingelesen und gleichzeitig wird den Mauszeiger mit dem entsprechenden Bild angepasst.

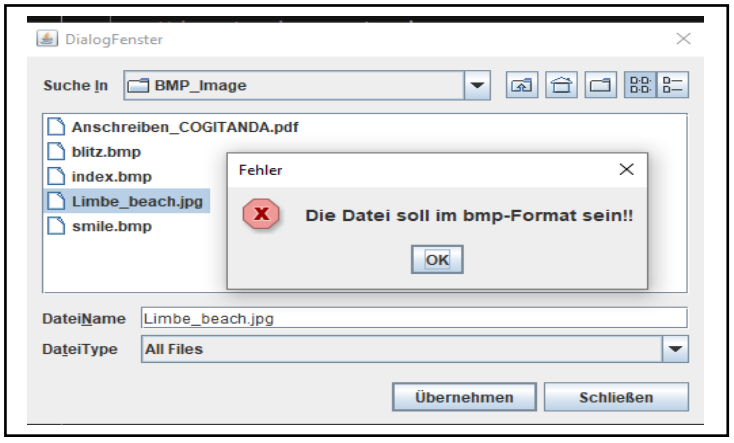
Durch einen Klick auf dem Button **Übernehmen** wird der neue Mauszeiger auch im Hauptfenster verwendet und der Dateiname für den Mauszeiger wird gleichzeitig in der Titelleiste des Hauptfensters angezeigt.





Was es hier am wichtigsten steht, ist die Fehlermeldung bei falschem Einlesen des **Dateinamens**. Nur **bmp-Bilder** können hier wie schont oben erwähnt eingelesen werden und bei falsche **bmp-Format** des Bildes, wird durch eine MessageBox angezeigt, dass Das entsprechende Bild nicht eingelesen werden kann. Damit die Programm-code bei der Fehlermeldung sinnvoll ist, habe ich mich für einen nicht passende **bmp-Format (ZB. .JPG und .PDF**) entschieden. Die Fehlermeldung lässt sich durch die folgenden Bilder sehen:





4. C/C++ vs. Java

C/C++ und Java sind die beliebten und weit verbreitete Sprachen auf der Welt. Die Herkunft von C stammt aus Assemblersprache, während C++ auf C-Sprache basiert und Java aus C und C++ stammt. Dazu verwenden C/C++ nur den Compiler, um die Programme-code zu übersetzen, wo Java eine interpretierte Sprache (Compiler + Interpreter) benutzt. Nach der Übersetzung wird die Programme-c in C/C++ direkt ausgeführt, was es anders in Java ist, wo nach der Übersetzung die Programme-code von JVM (Java Virtual Maschine) ausgeführt wird.

Ein besonderes Fall zwischen die beide Programmiersprache C/C++ und Java wäre die Plattformabhängig. C/C++ ist Plattformabhängig. Das heißt die Quelle-code wird nicht überall übertragbar, was in Java anders wäre, wo die Quelle-code überall übertragbar (Plattformunabhängig). Außerdem unterstützt C/C++ nicht nur das Zeigerkonzept, sondern auch die Vereinigungs- und Strukturdatentypen und verwendet Präprozessordirektiven (wie #include, #define usw.). Im Gegensatz zu Java, wo das Zeigerkonzept aus Sicherheitsgründen nicht von Java unterstützt und auch keine Präprozessordirektiven verwendet werden, sondern Pakete.

Im Kontext der WINAPI-Programmierung in C++ spricht man über standardisierte Funktionssätze, die es einer Anwendungssoftware ermöglichen, die Funktionalitäten der verschiedenen Betriebssysteme der Windows-Familie zu nutzen. Win32 ist das am häufigsten verwendete Feature, und sogenannte Windows-Programme sind normalerweise Programme, die Win32 verwenden. Dazu man kann nicht direkt bei der Windows-Programmen abfragen und ausgeben, sondern wir sollten einen Teil des Betriebssystems verwenden, um direkte Eingabeabfragen (Tastatur, Maus) und Ausgabe (Bildschirm, Drucker) zu machen. Im Gegensatz zu Java gibt es zur Oberflächenprogrammierung bereits seit Java-Versionen die Swing-Bibliothek. Diese Bibliothek besteht aus verschiedenen Klassen und Interfaces mit denen man unterschiedlichste GUI-Komponenten Wie Schaltflächen, Menüs, Tabellen usw. und Vieles mehr anlegen, anordnen und mit Logik verknüpfen kann.

Jedoch gibt es in WINAPI-Programme in C++ als auch in Java zahlreichen ähnlichen Elementen, die zur Erstellung und zur Gestaltung von Windows-Applikation oder Java dienen wie zum Beispiel: Icon, Menüs, Dialogfenster, Cursor usw. Aber Der Verwendungsart diesen Elementen ist abhängig vom Typ der WINAPI, dass man erstellt möchte. Dazu benötigen Windows-Programm in C++ mindestens Zwei Funktionen zum Einstiegspunkt der Anwendung: **WinMain** als Hauptfensterfunktion und die **Callback-Funktion** des Hauptfensters, was in Java anders ist, wo die **Fensterklasse** (**JFrame**) und eigene **Listener** (**Ereignisse und Aktion**) verwendet werden, um die Elemente des Fensters zu steuern. In Java gibt es aus Vielen unterschiedliche Fensterklasse wie zum Beispiel: **JAppelt, JDialog, JInternalFrame, JWindow** usw.

Java stellt neben **AWT** und **Swing** allerdings auch weitere Frameworks wie JavaFX oder SWT zur Verfügung, um grafische Benutzeroberflächen zu programmieren, welche einem viel Zeit und Arbeit ersparen können. Im Folgenden sollen jedoch speziell die Unterschiede zwischen AWT und Swing sowie die Gründe, wieso ich mich für Swing entschieden habe, herausgearbeitet werden.

5. AWT vs. Swing

AWT (Abstract Window Toolkit) ist der Ursprung der graphische Benutzeroberflächen und beschränkt sich auf GUI Komponenten, die von allen OS (Operating System) darstellbar sind. Die AWT enthält Komponenten und Container, die plattformabhängig sind und bietet weniger Komponenten als Swing. Die AWT-Komponente wird nicht nur zur Graphischer Darstellung des Objekts auf dem Bildschirm und Interaktionsmöglichkeiten mit dem Benutzer verwendet, sondern auch besitzt auch typische Methoden zum Setzen und Abfragen von Eigenschaften der Darstellung bspw. Methode für die Größe, Position, Farbe usw. Dazu wird die Systemreaktion bei Benutzeraktionen zur Definition von Listener festgelegt.
Außerdem ist der Container (AWT) eine Unterklasse von Komponente (AWT). Das heißt der AWT-Container sind Komponenten, die die andere Komponente enthalten können und die Schachtelungen sind über mehrere Stufen möglich. Weiterhin gibt es auch typische Methode der AWT-Container zum Hinzufügen, Entfernen und Abfragen von Komponenten. AWT ist Heavyweight (Schwergewicht), da es die Ressourcen des Betriebssystems nutzt und in Bezug auf die Leistung ist Java AWT langsamer als Swing.

Swing hingegen wird hauptsächlich als Java Foundation Classes (JFC) bezeichnet und ist unabhängig von Plattformspezifischen Einschränkungen. Er hat eine bessere Trennung von Funktionalität und Daten sowie Ihrer Darstellung, aber auch Weiterverwendung einiger AWT-Klassen. Zudem ist der Swing größtenteils lightweight(leichtgewichtig), da er kein Betriebssystemobjekt zur Verarbeitung benötigt. Die Swing-Komponenten sind auf der Oberseite von AWT aufgebaut und ist schneller als das AWT. Deswegen realisiert der Swing-Konzept, die nicht in AWT enthalten sind.

Meine Meinung nach AWT oder Swing Komponente zur grafische Benutzeroberfäche ist, dass die Swing-Komponente mehrere Komponenten als AWT enthält. Was die Implementierung meiner Aufgabe angeht, wurden die Swing-Komponenten am meisten verwendet. Aber wichtig sind die Pakete der beiden Java-Standard, bei der Programmierung der grafischen Benutzeroberfläche zu importieren: **Import java.awt. *** und **Import javax.swing.***

Quellen:
Beispielprogramme und Vorlesung-Skript (PI2 und GUI) aus Ilias.
https://www.javatpoint.com/awt-and-swing-in-java
https://www.javatpoint.com/c-vs-cpp-vs-java
https://www.java-tutorial.org/jfilechooser.html
https://stackoverflow.com/questions/29086300/how-to-change-jfilechooser-label-look-in-to-save-in-not-title
https://www.codejava.net/java-se/swing/add-file-filter-for-jfilechooser-dialog
https://docs.oracle.com/en/java/javase/13/docs/api/java.desktop/javax/swing/JFileChooser.html
https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setscrollinfo
https://www.codeproject.com/Articles/1889/Using-SetWorldTransform-to-Rotate-Basic-Shapes-by
https://docs.microsoft.com/de-de/windows/win32/gdi/rotation
https://docs.microsoft.com/de-de/windows/win32/learnwin32/learn-to-program-for-windows
https://www.youtube.com/watch?v=8GCvZs55mEM&list=PLWzp0Bbyy_3i750dsUj7yq4JrPOIUR_NK

Ich bestätige hiermit, dass ich**, Leonel Nguimatsia Tsobguim**, diese Hausarbeit zusammen mit den abgegebenen Programmen allein und nur mithilfe der angegebenen Quellen für das Modul **„Programmierung Grafischer Benutzeroberflächen“** an der TH Köln angefertigt habe.