Übungsaufgabe 1: Gruppenprojekt Das Aquarium

Thema: GUI, Graphics, Vektoren & Multithreading

[cite$_s$*tart*]*DieseAufgabebautaufdemsimplenZeichenprogramm*[*cite* : 110]*auf*, *fgtabereinedynami*

A: Das Aquarium

Erstellen Sie eine Anwendung, die ein Aquarium simuliert.

- Es gibt einen `JFrame` mit einem `JPanel` (die Wasserfläche).
- Durch einen Mausklick (Left-Click) an eine beliebige Stelle soll ein neuer Fisch erzeugt werden.
- Ein Fisch ist grafisch einfach eine Ellipse oder ein Kreis (Farbe zufällig).

B: Die Logik (Threads)

- Jeder Fisch soll nicht nur als Daten-Objekt im Vektor existieren, sondern auch ein **eigener Thread** sein (oder von einem Timer gesteuert werden).
- Die Fische sollen sich horizontal von links nach rechts bewegen.
- Trifft ein Fisch auf den rechten Rand, soll er am linken Rand wieder erscheinen. [cite$_s$*tart*]
- **Wichtig:** Das Panel muss regelmäSSig `repaint()` aufrufen, damit die Bewegungen sichtbar werden[cite: 106].

Tipps zur Struktur

- Klasse `Fisch`: Speichert x, y, Farbe und Geschwindigkeit. Implementiert das Interface `Runnable`. [cite$_s$*tart*]
- Klasse `AquariumPanel`: Besitzt den `Vector<Fisch>`[cite: 128]. Die `paintComponent` Methode iteriert über den Vektor und zeichnet alle Fische an ihren *aktuellen* Positionen.

Übungsaufgabe 2: Fenster-Panik (FrameChaos)

Thema: JFrame, Threading & Randomization

[cite$_s$*tart*]*DieseAufgabeistdasGegenstckzumFrameMover*[*cite* : 144], *erfordertaberInteraktion.*

Implementieren Sie eine Klasse `PanicFrame`.

- Das Programm startet **einen** einzigen Frame (GröSSe 300x300).
- In diesem Frame befindet sich ein Button mit der Aufschrift Klick mich!.
- Der Button hat einen `MouseListener`.

Die Logik:

- Versucht der Nutzer, mit der Maus über den Button zu fahren (`mouseEntered`) oder ihn zu klicken, soll der **gesamte Frame** sofort an eine zufällige andere Position auf dem Bildschirm springen (`setLocation`). [cite$_start$]
- Gleichzeitig soll sich bei jedem Sprung die Hintergrundfarbe des Panels zufällig ändern[cite: 146].

Zusatzaufgabe (Threads):
Starten Sie im Hintergrund einen Thread, der die GröSSe des Fensters alle 500ms zufällig zwischen 100x100 und 400x400 verändert, um den Nutzer zusätzlich zu verwirren.

Übungsaufgabe 3: Rekursion & Fraktale

[cite$_s$*tart*]**Thema:***RekursionundGraphics*(*analogzudenAufgabenimSkript*[*cite* : 153])

A: Rekursives Zeichnen (Der Möchtegern-Baum)

Schreiben Sie eine Klasse `FractalPanel` (erbt von `JPanel`).
Implementieren Sie eine rekursive Methode:
`void drawBranch(Graphics g, int x, int y, int length, int angle)`

- Die Methode zeichnet eine Linie (den Ast) der Länge `length` vom Punkt (x, y) in Richtung `angle`.
- Am Ende des Astes ruft sich die Methode **zweimal** selbst auf:
  1. Einmal mit etwas kürzerer Länge und einem Winkel nach links.
  2. Einmal mit etwas kürzerer Länge und einem Winkel nach rechts.
- **Abbruchbedingung:** Wenn die Länge kleiner als 2 Pixel ist, stoppt die Rekursion.
- Starten Sie die Zeichnung in der `paintComponent` Methode am unteren Bildschirmrand.

B: Mathematische Rekursion

Implementieren Sie eine statische Methode:
`boolean isPalindrome(String s)`

- Die Methode soll **rekursiv** prüfen, ob ein String vorwärts wie rückwärts gleich ist (z.B. Lagerregal oder Rentner).
- *Tipp:* Vergleichen Sie das erste und letzte Zeichen. Wenn diese gleich sind, schneiden Sie sie ab und rufen die Methode mit dem Rest-String erneut auf.