PTP - Aufgabe 1.4 und 1.5

Fortsetzung der Überarbeitung des `Draw`-Programms.

Teilaufgabe 1.4 - Refactoring von `Draw.java`:

Verbessern Sie die Code-Struktur, ohne die Funktionalität zu verändern, unter anderem durch:

1. Einführung von Hashtabellen für Farben und Formen (z.B. `color["Green"] => green`), um komplexe "if"-Anweisungen in `ColorItemListener` bzw. `ShapeManager` entfernen zu können.
2. Aufteilen einiger Funktionen in mehrere Teilfunktionen (insbesondere Konstruktor von DrawGUI).
3. Bessere Aufteilung der Funktionalität auf die Klassen (dabei können Klassen entfernt oder hinzugefügt werden) und Beseitigung von Redundanzen. Dabei sollte auch der UI-Delegate-Ansatz von Swing, der eine Abwandlung des MVC-Ansatzes darstellt (vgl. Partl oder den Wikipedia-Eintrag), verfolgt werden. Die Nutzung von "Observern" ist möglich, aber nicht zwingend.
4. Einführung von Command-Objekten (siehe Gamma, Helm, Johnson, Vlissides: "*Entwurfsmuster*"/"*Design Patterns*") sowie die Links zum Thema CommandPattern und Design patterns). Das heißt, dass die API-Funktionen bzw. die entsprechenden Mausaktionen nicht mehr unmittelbar ein Graphikobjekt zeichnen. Stattdessen wird für jedes Graphikobjekt zunächst eine Instanz einer entsprechenden Kommandoklasse erzeugt. Alle Kommandoklassen implementieren ein **Interface** `Drawable` mit der (einzigen!) abstrakten Methode `void draw(Graphics g)`, mit der das Graphikobjekt gezeichnet werden kann, ohne dass dessen Typ bekannt sein muss. Die Command-Objekte werden in einer Command-Queue gespeichert, um die Voraussetzung für die Implementation des *redraw* und des *undo* zu schaffen (siehe unten). `Draw` bleibt aber weiterhin die zentrale (Haupt-)Klasse und stellt die API bereit.

Teilaufgabe 1.5 - Modifikation und Erweiterung von `Draw.java`:

Das Ausgangsprogramm zeigt ein z.T. unschönes Verhalten, dass vermutlich auch in Ihrer Überarbeitung zu bemerken ist. So ist z.B. das Malen hinter den Menüs möglich, Teile der Zeichnung verschwinden bei Fensteroperationen etc. Korrigieren Sie solche Probleme!

Fügen Sie folgende neue Funktionalität hinzu:

1. Sofern nicht bereits in Teil 1.2 erledigt, bauen Sie einen Button `"Auto"` ein, der mittels der Methode `autoDraw()` die Testgrafik erzeugt und ...
2. ... ein *redraw*, wenn das Fenster neu gezeichnet werden muss, weil es z.B. verdeckt war oder sich seine Größe geändert hat (Überschreiben der `paintComponent()` Methode).
3. Menüpunkt zum Speichern des aktuellen Bildes (zumindest als BMP); Dateiname und Ablageort sollen mittels `"File chooser"` ausgewählt werden können.
4. Menü zum Setzen der Hintergrundfarbe analog zur Vordergrundfarbe (default="White", diese Farbe muss dann natürlich auch zur Auswahl stehen)
5. Mindestens drei weitere Formen (z.B. Dreiecke, gefüllte Formen, Bitmaps).

6. Implementation der *Undo*- und der *Redo*-Operation. Die Methoden fließen in die API ein und haben die Signaturen `public void undo()` bzw. `public void redo()`, damit sie auch in die Tests eingebaut werden können.
7. Über einen Menüpunkt `"Text speichern ..."` soll eine Textdarstellung der Zeichenkommandos (mit Parametern) in eine Datei ausgegeben werden. Diese Darstellung kann "flach" mit einer Zeile pro Kommando oder auch XML-artig strukturiert sein. Umgekehrt soll per Menüpunkt `"Datei lesen..."` die Erzeugung eines Bildes aus eingelesenen (Text-)Kommandos möglich sein. (Beides mittels `"File chooser"` wie unter Punkt 3.) Die zugehörigen API-Methoden haben die Signaturen `public void writeText(String name) throws TxtIOException` bzw. `public void readText(String name) throws TxtIOException .`
8. *Freiwillige* Zusatzaufgaben: weitere Farben, Menüpunkt zum Einlesen eines Bildes (z.B. als BMP), Zeichnen von rotierten Objekten, Splines ... *(mögliche Ausgleichspunkte...)*

Hinweise zu den Teilaufgaben:

- Prüfen Sie stets mit Hilfe der *Unit Tests*, dass die (vorherige) Funktionalität nicht verändert wurde. Neue Funktionalität muss in die Tests einfließen!
- Nutzen Sie geeignete Modifikatoren, um die Sichtbarkeit von Variablen, Methoden und Klassen sinnvoll zu regeln!
- Achten Sie auf "saubere" (sinnvolle) Fehlerbehandlung! Keine "Abstürze"! ("defensiver" Ansatz)
- Die Reaktion auf Fenstersteuerelemente nicht vergessen!
- Nach Hinzufügen von Funktionalität muss ggf. erneut "refactored" werden!

Abgabetermine:

Teilaufgabe 1.4	**Mittwoch 5.5.21 / Donnerstag, 6.5.21 (am Ende des Tages)**
Teilaufgabe 1.5	**Mittwoch 19.5.21 / Donnerstag, 20.5.21 (am Ende des Tages)**

Die Termine sind als jeweils Endtermine zu sehen. Eine frühere Abgabe ist durchaus erwünscht, um für den Projektteil mehr Zeit zu haben! In Absprache darf auch schon parallel mit Projekt(vor-)arbeiten begonnen werden.

Kontakt:

Andreas Heymann
Email: ptp@informatik.uni-hamburg.de