3. PRAKTIKUMSAUFGABE IM SOSE'2019

Vererbung, Polymorphie und dynamische Datenstrukturen

Aufgabenstellung:

Erweitern Sie das noch unvollständige Konsolenanwendungsprojekt „TextGraphics", so dass die vorgegebene Anwendung in `Program.cs` einwandfrei funktioniert und eine grafische und animierte Szene auf der Textkonsole dargestellt wird. Der Szenenaufbau, die Darstellung, Animation und Benutzereingabe sind in **`Program.Main()`** bereits vollständig vorimplementiert. Genauso ist auch die Klasse **`Image`** vorgegeben, die ein textbasiertes farbiges „Bild" verwalten und auf der Konsole ausgeben kann. Der Quellcode in den entsprechenden Dateien `Program.cs` und `Image.cs` bildet die Grundlage für Ihre eigene Entwicklung und darf nicht mehr verändert werden. Bei korrekter Implementierung sollte die fertige Anwendung folgende vordefinierte Funktionalität aufweisen:

- Darstellung einer textgrafischen Szene bestehend aus einem Hintergrund (hellblauer Himmel, grüner Boden), einem Baum (mit dunkelrotem Stamm und grünen Blättern), einer gelben Sonne, einer weißen Wolke und einer Person (mit dunkelblauen Beinen, rotem Oberkörper und einem grauen Kopf).
- Die Wolke bewegt sich langsam von links nach rechts.
- Die Person kann über die Pfeiltasten nach links und rechts bewegt werden.
- Die Anwendung beendet sich durch das Drücken der Escape-Taste.

Das Visual Studio 2017 Projekt „TextGraphics", inklusive der vorimplementierten Quellcodes, ist – zusammen mit dieser Aufgabenstellung – im Moodle-Kurs „ LV-SoSe2019 Programmieren 2 Praktikum" zu finden.

1. Vorgaben zur Aufgabenlösung – Geometrische Formen:

In der Quellcodedatei `Program.cs` wird zunächst eine Szenenbeschreibung aufgebaut. Dabei werden unterschiedliche geometrische Formen verwendet, die von Ihnen zu programmieren sind: Punkte (Klasse **`Point`**), horizontale und vertikale Linien (Klassen **`HorizontalLine`** und **`VerticalLine`**), Rechtecke (Klasse **`Rectangle`**) und ausgefüllte Rechtecke (Klasse **`FilledRectangle`**). Die Umsetzung all dieser Formen soll auf der vorgegeben Basisklasse **`Shape`** erfolgen, die in der Projektdatei `Shape.cs` teilweise vorimplementiert ist. Die Klasse **`Shape`** muss nur noch von Ihnen um eine (leere) Basismethode mit der Signatur **`Paint(int left, int top, Image image)`** erweitert werden. Die Methode soll in den jeweils abgleitenden Klassen implementiert werden und das Zeichnen der entsprechenden geometrischen Figur ab der 2D-Startposition (**`left`**, **`top`**) im Bild **`image`** umsetzen. Für das Zeichnen benötigen Sie aus der Klasse **`Image`** nur die vorimplementierte Methode **`SetColor(int x, int y, ConsoleColor color)`**, die an der Bildposition (**`x`**, **`y`**) einen „Pixel" der Farbe **`color`** setzt. Die Aufzählung (**`enum`**) **`System.ConsoleColor`** ist im .NET-Framework implementiert.

Bei der Implementierung der geometrischen Formen programmieren Sie die folgenden Konstruktoren und nutzen Sie dabei die in der Vorlesung kennengelernte „Verkettung von (Basis-)Konstruktoren“:

- **`Point(ConsoleColor color)`** – Konstruktion eines farbigen Punktes
- **`HorizontalLine(uint width, ConsoleColor color)`** – Horizontale Line mit gegebener Breite
- **`VerticalLine(uint height, ConsoleColor color)`** – Vertikale Line mit gegebener Höhe
- **`Rectangle(uint width, uint height, ConsoleColor color)`** – Rechteck mit geg. Ausmaßen
- **`FilledRectangle(uint width, uint height, ConsoleColor color)`** – Ausgefülltes Rechteck

Vermeiden Sie so weit wie möglich Codewiederholungen. Daher soll die Klasse **`FilledRectangle`** nicht direkt auf **`Shape`** basieren, sondern auf der Klasse **`Rectangle`**.

2. Vorgaben zur Aufgabenlösung – Szenenbeschreibung:

In der Quellcodedatei `Scene.cs` sollen Sie die Klasse **`Scene`** erweitern, so dass Sie eine zur Anwendung passende Szenenbeschreibung umsetzt. Eine Szenenbeschreibung enthält alle Szenenelemente, die auf dem Bildschirm gezeichnet werden sollen. Sie ist als eine dynamische (Listen-)Datenstruktur zu implementieren. Jedes einzelne Szenenelement (Klasse **`Scene.Element`**) soll – neben der Information zur Verkettung einzelner Elemente – die folgenden Daten enthalten:

- Ein Array an Zeichenkomponenten (Klasse **`Scene.Painting`** – Zur Beschreibung siehe unten)
- Zwei ganzzahlige Versätze in X- und Y-Richtung (**`XOffset`**, **`YOffset`**) für alle Zeichenkomponenten (Beide Versätze sollen bei der Konstruktion jeweils mit dem Standardwert (**`0`**) initialisiert bleiben)

Die Zeichenkomponenten-Klasse (**`Scene.Painting`**) soll wiederum die folgenden Daten enthalten:

- Zwei ganzzahlige Werte (**`Left`** und **`Top`**), die die 2D-Bildposition der geometrischen Form angeben
- Die zu zeichnende geometrische Form (Siehe Beschreibung oben im Abschnitt 1)

Die teilweise vorimplementierte Klasse **`Scene`** enthält bereits die fertige Methode **`Render(Image image)`**, die alle Szenenelemente und die jeweils enthaltenen Zeichenkomponenten durchläuft und mittels der oben beschriebenen **`Paint(…)`**-Methode in das übergebene Bild zeichnet. Die Methode **`Render(…)`** darf nicht mehr von Ihnen verändert werden und soll Ihnen als Vorgabe für die Umsetzung (inkl. Benennung) der noch folgenden restlichen Funktionalitäten der Klasse **`Scene`** dienen:

- Dynamische Verkettungsstruktur für die oben beschriebenen Szenenelemente
- Methode **`Add(…)`** zum Hinzufügen von einer oder mehreren der oben beschriebenen Zeichenkomponenten zur Szene, so wie in `Program.cs` vorgebeben (Tipp: Verwenden Sie das C#-Schlüsselwort **`params`**). Die Methode soll das entsprechend konstruierte neue Szenenelement als Rückgabe liefern. Beachten Sie, dass neue Szenenelemente jeweils am Ende der Verkettung eingefügt werden müssen, damit die richtige Zeichenreihenfolge erzielt wird.