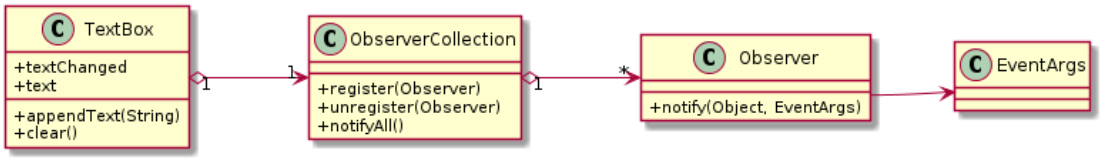
Übungsblatt 3

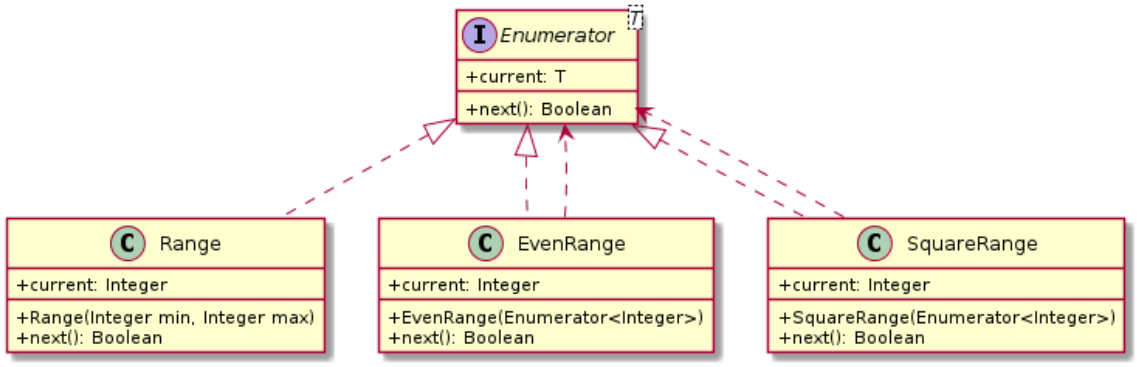
Aufgabe 8 Ein Eventsystem erstellen

Bauen Sie eine abstrakte Klasse *ObserverCollection*, an der sich Beobachter an- und abmelden können. Die Klasse benötigt daher Methoden wie *register(Observer)*, *unregister(Observer)* und *notifyAll()*.
Beobachter müssen von einer Klasse *Observer* erben. Alle Beobachter müssen eine Methode *notify(Object, EventArgs)* implementieren, wobei das erste Argument die Instanz des Senders darstellt. Der zweite Übergabeparameter ist eine Instanz von der Klasse *EventArgs* und wird dazu benötigt um eine beliebige Anzahl möglicher Werte zu übertragen.
Erstellen Sie abschließend eine Klasse *TextBox*, die eine Eigenschaft *text* und mehrere Methoden wie z.B. *appendText*, *clear*, ... hat. Wenn der Text geändert wird, soll die Methode *notify* aus der *ObserverCollection*-Instanz aufgerufen werden, die im Feld *textChanged* hinzugefügt worden sind.



Aufgabe 9 Zahlenketten

Erstellen Sie eine Klasse *Range* die das Iterator Pattern implementiert. Anschließend sollen noch weitere Klassen implementiert werden, die Instanzen von Range ausnutzen um diese weiter zu filtern.
Folgendes Klassendiagramm soll implementiert werden:



Das Verhalten des Range-Iterators soll so implementiert werden, dass bei Konstruktion einer Instanz mit den Argumenten 0 und 10 die Zahlen 0, 1, ..., 9, 10 iteriert werden.
Konstruiert man eine Instanz von *EvenRange* über diese Instanz von *Range*, so würde man die Zahlen 0, 2, ..., 8, 10 erhalten. Die *SquareRange* Klasse hingegen würde Zahlen 0, 1, 4, 9, ..., 81, 100 liefern.

Aufgabe 10 4-3-3 und 4-4-2

In einem Fußballspiel wird das Verhalten der Spieler vom Computer über das *Strategy Pattern* bestimmt. Erstellen Sie eine Klasse *FormationStrategy*, die zwei konkrete Realisierungen in Form von *Formation442Strategy* und *Formation433Strategy* hat.
Jede Realisierung muss folgende Methoden besitzen:

- *placeDefense()*,
- *placeMidfield* und
- *placeStrikers*

Außerdem braucht jede Strategie eine Menge Informationen, die über zwei Parameter übergeben werden:

- *SoccerMatch* für das aktuelle Spiel und
- *Team* zum Zugriff auf das eigene Team.

Wie würde ein passender objektorientierter Entwurf in Form eines Klassendiagramms für ein solches Spiel aussehen?

! Wichtig

Alle Diagramme können über Anwendungen (z.B. PlantUML, yUML, Visual Studio, ...) oder auch per Hand gezeichnet werden.