Objektorientierte Sprachen

Sommersemester 2021 - Erste Prüfung

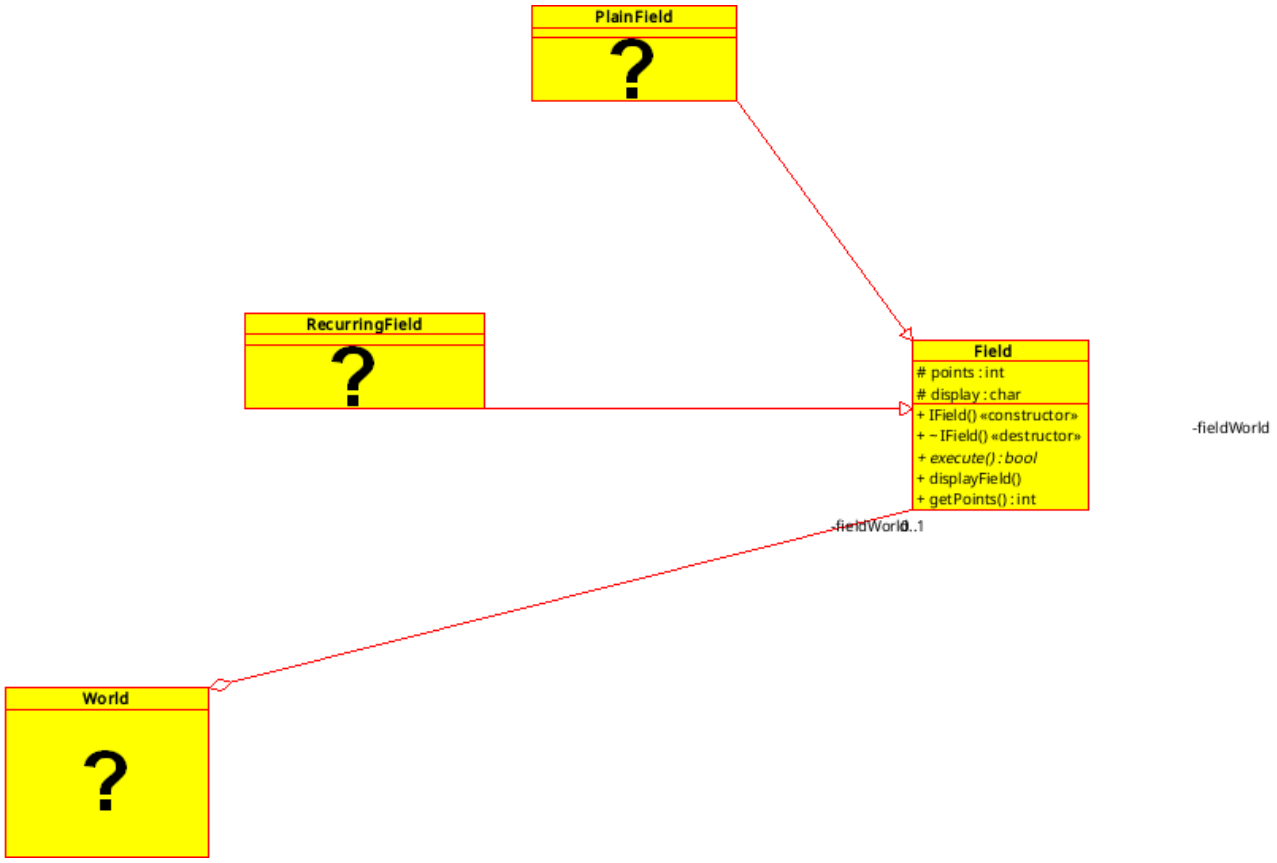
An- und Abgabe

Implementieren Sie ein Programm das ein zweidimensionales Feld von Objekten solange anspricht und dabei Punkte sammelt, bis keine Objekte mit Punkten mehr vorhanden sind. Sie haben dazu eine Grundstruktur vorgegeben, die bereits eine main-Funktion enthält, sowie die Header- und CPP-Dateien für die abstrakte Field-Klasse bereitstellt. Erweitern Sie nun diese Grundstruktur, indem Sie die abstrakte Klasse ausbauen, zwei Klassen implementieren, die von ihr erben und die Welt-Klasse komplett definieren und implementieren.

Laden Sie Ihre Abgabe in einem ZIP-Archiv verpackt auf Moodle hoch, fehlerhaft oder unvollständig hochgeladene Abgaben können nicht nachgereicht werden.
Die Abgabe ist bis 12:50 möglich.

Die Grundstruktur ist im Code durch Kommentare erklärt und auf der nächsten Seite finden Sie ein Klassendiagramm sowie eine Zusammenfassung des Programmablaufs.

Klassendiagramm



Übersicht

- Zu Beginn des Programms wird ein neues World-Objekt angelegt, dabei wird das Objekt mit einem zweidimensionalen Array von Pointern auf Field-Objekte initialisiert. **Implementieren Sie den Konstruktor der Klasse World.**
- Im Konstruktor der World-Klasse soll die Spielwelt zufällig entweder mit RecurringField- oder PlainField-Objekten befüllt werden, beide Klassen sollen von der abstrakten Klasse Field erben. **Definieren Sie die rein virtuelle execute-Funktion in der Klasse Field, führen Sie dann die Vererbung an die beiden Kindklassen durch und implementieren Sie dann deren Konstruktoren und execute-Funktionen.**
- Danach werden solange zufällige Felder des zweidimensionalen Arrays angesprochen indem die execute-Funktion der Felder durch die executeField-Funktion der Welt aufgerufen wird, bis die Welt keine Felder mehr enthält, die Punkte haben. **Implementieren Sie die executeField-Funktion der Welt und zur Prüfung, ob die Welt bereits leer ist die isEmpty-Funktion.**
- **Implementieren Sie auch die printWorld-Funktion**, zur Ausgabe der Spielwelt.
- Während der Ausführung der executeField-Funktion kann es zu Exceptions kommen, **implementieren Sie dafür eine entsprechende Fehlerbehandlung in der main-Funktion**.
- Zum Schluss soll natürlich auch jeglicher Speicher korrekt freigegeben werden, **implementieren Sie dazu den Destruktor der Klasse World.**

Aufgaben

main-Funktion

- Implementieren Sie eine Fehlerbehandlung durch Exceptions, für den Aufruf der executeField-Funktion, fangen Sie dabei sowohl eine invalid_argument-Exception als auch allgemeine Objekte ab.

Abstrakte Klasse Field

Definieren Sie folgende **rein virtuelle** Funktion für die abstrakte Klasse:

- **execute():** Die Funktion soll einen booleschen Wert zurückgeben.

Klasse RecurringField

Definieren Sie die Klasse RecurringField, die von Field erben soll, aufgeteilt in separate Header- und CPP-Dateien. Implementieren Sie folgende Funktionen für die Klasse:

- **Konstruktor:** Der points-Wert des Objekts soll zufällig auf 1, 2 oder 3 gesetzt werden und der display-char auf O.
- **execute():** Die Funktion soll den points-Wert um 1 senken und dann true zurückgeben.

Klasse PlainField

Definieren Sie die Klasse PlainField, die von Field erben soll, aufgeteilt in separate Header- und CPP-Dateien. Implementieren Sie folgende Funktionen für die Klasse:

- **Konstruktor:** Der points-Wert des Objekts soll auf 0 gesetzt werden und der display-char auf _.
- **execute():** Die Funktion gibt false zurück.

Klasse World

Definieren Sie die Klasse World aufgeteilt in separate Header- und CPP-Dateien. Die Klasse soll eine zweidimensionale Spielwelt enthalten die aus Field-Pointern besteht, damit dann über Polymorphismus auf RecurringField- und PlainField Objekte zugegriffen werden kann. Implementieren Sie folgende Funktionen für die Klasse:

- **Konstruktor:** Der Konstruktor erstellt eine 3x3 Spielwelt und belegt dabei mit einer 50:50 Wahrscheinlichkeit die einzelnen Felder mit RecurringField- und PlainField-Objekten.
- **Destruktor:** Der Destruktor gibt den Speicher, der im Konstruktor für die Spielwelt beansprucht wurde wieder frei.
- **printWorld():** Gibt die Spielwelt in einem 3x3 Raster aus indem für jedes Feld die displayField-Funktion der Klasse Field aufgerufen wird.

- **executeField(int x, int y):** Es wird versucht die execute-Funktion des Felds auszuführen, das an den Koordinaten x/y in der Spielwelt gespeichert ist. Ist einer der Parameter ungültig (zu groß oder zu klein) wird eine invalid_argument-Exception geworfen.
  Wurde die execute-Funktion aufgerufen und hat sie true retourniert **(was nur bei einem RecurringField-Objekt der Fall sein sollte)** wird geprüft ob das Feld an dieser Stelle noch einen Punktewert größer 0 besitzt.
  Ist das der Fall wird der Punktewert des Feld-Objekts auf ein neues RecurringField-Objekt übertragen, das anstelle eines zufälligen PlainField-Objekts in der Spielwelt platziert wird.
  Dann wird das Feld-Objekt durch ein PlainField-Objekt ersetzt.
- **isEmpty():** Die Funktion soll prüfen ob sich noch RecurringField-Objekte in der Spielwelt befinden und dementsprechend false retournieren, wenn das der Fall ist, ansonsten true.

Benotungsaspekte

Aspekt	Bewertung
main - Fehlerbehandlung mit Exceptions	10%
Field - execute	5%
RecurringField - Header-/CPP-Dateien	3%
RecurringField - Vererbung+Konstruktor	7.5%
RecurringField - execute	2.5%
PlainField - Header-/CPP-Dateien	3%
PlainField - Vererbung+Konstruktor	7.5%
PlainField - execute	2.5%
World - Header-/CPP-Dateien	3%
World - Konstruktor	10%
World - Destruktor	8.5%
World - printWorld	7.5%
World - executeField	7.5%
World - executeField-Exception	7.5%
World - executeField-Speicherverwaltung	7.5%
World - isEmpty	7.5%
Gesamt	100%

Compiler-Warnungen, Speicherfehler und schlechter Stil können zu Punkteabzügen führen.

Benotung

$\geq 88\%$	Sehr gut
$\geq 75\%$ und $< 88\%$	Gut
$\geq 63\%$ und $< 75\%$	Befriedigend
$\geq 50\%$ und $< 63\%$	Genügend
$< 50\%$	Nicht genügend