Objektorientierte Sprachen

Sommersemester 2022 - Zweite Prüfung

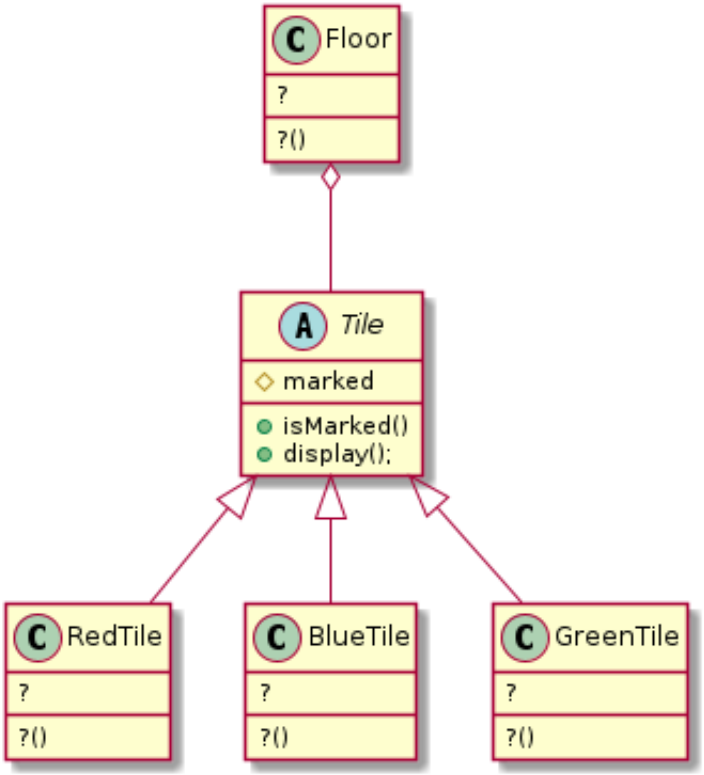
An- und Abgabe

Implementieren Sie ein Programm das einen Boden (Klasse Floor) von farbigen Kacheln (Klasse Tile) solange prüft und dabei Kacheln markiert und austauscht bis nur noch Kacheln einer bestimmten Farbe vorhanden sind.
Sie haben dazu eine Grundstruktur vorgegeben, die bereits eine main-Funktion enthält, sowie die Header- und CPP-Dateien für die abstrakte Tile-Klasse bereitstellt. Erweitern Sie nun diese Grundstruktur, indem Sie die abstrakte Klasse ausbauen, drei Klassen implementieren, die von ihr erben und die Floor-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 19:20 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 Floor-Objekt angelegt, dabei wird das Objekt mit einem zweidimensionalen Feld von Pointern auf Tile-Objekte initialisiert. **Implementieren Sie den Konstruktor der Klasse Floor.**
- Im Konstruktor der Floor-Klasse soll der Boden zufällig mit BlueTile- und GreenTile-Objekten befüllt werden, die beiden Klassen, als auch die RedTile Klasse, sollen von der abstrakten Klasse Tile erben. **Definieren Sie die rein virtuelle getColor- und mark-Funktion in der Klasse Tile, führen Sie dann die Vererbung an die drei Kindklassen durch und implementieren Sie dann deren Konstruktoren und getColor- und mark-Funktionen.**
- Danach werden solange zufällige Kacheln des zweidimensionalen Feldes markiert und getauscht indem die mark- und getColor-Funktion der Kacheln durch die mark- und replace-Funktion des Bodes aufgerufen wird, bis der Boden nur mehr aus roten Kacheln besteht. **Implementieren Sie die mark- und replace-Funktion der Floor-Klasse und zur Prüfung, ob der Boden bereits nur noch aus roten Kacheln besteht, die isRed-Funktion.**
- **Implementieren Sie auch die print-Funktion**, zur Ausgabe der Bodens. Passen Sie die display-Funktion der Klasse Tile an.
- Während der Ausführung der mark- und replace-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 Floor.**

Aufgaben

main-Funktion

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

Abstrakte Klasse Tile

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

- **getColor():** Die Funktion soll ein einzelnes Zeichen zurückgeben.
- **mark():** Die Funktion soll den marked-Wert setzen.

Passen Sie die Funktion display so an das statt X die entsprechende Farbe ausgegeben wird.

Klasse RedTile

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

- **Konstruktor:** Der marked-Wert des Objekts soll auf false gesetzt werden.
- **mark():** Die Funktion setzt marked auf false.
- **getColor():** Die Funktion soll das Zeichen 'R' zurückgeben.

Klasse BlueTile

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

- **Konstruktor:** Der marked-Wert des Objekts soll auf false oder true gesetzt werden mit einer 50:50 Wahrscheinlichkeit.
- **mark():** Die Funktion setzt marked auf true.
- **getColor():** Die Funktion soll das Zeichen 'B' zurückgeben.

Klasse GreenTile

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

- **Konstruktor:** Der marked-Wert des Objekts soll auf true gesetzt werden.
- **mark():** Die Funktion setzt marked auf true.
- **getColor():** Die Funktion soll das Zeichen 'G' zurückgeben.

Klasse Floor

Definieren Sie die Klasse Floor aufgeteilt in separate Header- und CPP-Dateien. Die Klasse soll ein zweidimensionales Feld enthalten die aus Tile-Pointern besteht, damit dann über Polymorphismus auf RedTile-, BlueTile- und GreenTile Objekte zugegriffen werden kann. Implementieren Sie folgende Funktionen für die Klasse:

- **Konstruktor:** Der Konstruktor erstellt einen 3x3 Feld und befüllt dieses dann mit blauen und grünen Kacheln, dabei soll mit einer 50:50 Wahrscheinlichkeit BlueTile- und GreenTile-Objekte erstellt werden.
- **Destruktor:** Der Destruktor gibt den Speicher, der im Konstruktor für das Feld beansprucht wurde wieder frei.
- **print():** Gibt den Boden in einem 3x3 Raster aus indem von jeder Kachel die display-Funktion der Klasse Tile aufgerufen wird.
- **mark(int x, int y):** Ist einer der Parameter ungültig (zu groß oder zu klein) wird eine invalid_argument-Exception geworfen.
  Es wird die mark-Funktion der Kachel aufgerufen, die sich an den Koordinaten x/y des Bodens befindet.
- **replace(int x, int y):** Ist einer der Parameter ungültig (zu groß oder zu klein) wird eine invalid_argument-Exception geworfen.
  Falls sich an den Koordinaten x/y des Bodens eine grüne Kachel befindet wird sie durch eine neue blaue Kachel ersetzt.
  Andernfalls, falls sich an den Koordinaten x/y des Bodens eine blaue Kachel befindet und diese markiert ist wird sie durch eine neue rote Kachel ersetzt.
  Nutzen Sie die Funktionen getColor und isMarked der von Tile vererbten Klassen.
- **isRed():** Die Funktion soll prüfen ob der Boden nur noch aus RedTile-Objekte besteht und dementsprechend true retournieren, wenn das der Fall ist, ansonsten false.

Benotungsaspekte

Aspekt	Bewertung
main - Fehlerbehandlung mit Exceptions	10%
Tile - getColor und mark	7%
RedTile - Header-/CPP-Dateien	1%
RedTile - Vererbung+Konstruktor	4%
RedTile - getColor und mark	3%
BlueTile - Header-/CPP-Dateien	1%
BlueTile - Vererbung+Konstruktor	4%
BlueTile - getColor und mark	3%
GreenTile - Header-/CPP-Dateien	1%
GreenTile - Vererbung+Konstruktor	4%
GreenTile - getColor und mark	3%
Floor - Header-/CPP-Dateien	3%
Floor - Konstruktor	10%
Floor - Destruktor	8.5%
Floor - print	7.5%
Floor - mark	2%
Floor - mark-Exception	3.5%
Floor - replace	6%
Floor - replace-Exception	3.5%
Floor - replace-Speicherverwaltung	7.5%
Floor - isRed	7.5%
Gesamt	100%

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