Übung 10: Design Patterns

Abgabetermin: 14.6.2018, 10:15

Name: _____ **Matrikelnummer**: _____

INF: ❑G1, Prähofer ❑G2, Weninger ❑G3, Prähofer ❑G4, Weninger ❑G5, Löberbauer ❑G6, Aumayr

WIN: ❑ G1, Khalil ❑ G2, Hummel ❑ G3, Khalil ❑ G4, Löberbauer

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 10	24	Java-Programm	Java-Programm	❑	

Übung 10: Alien Catcher

1. Im Spiel *Alien Catcher* tauchen an zufälligen Stellen Aliens mit verschiedenen Größen auf, welche im Laufe der Zeit kleiner werden. Spieler müssen auf die Aliens klicken bevor diese verschwinden („flüchten“), um sie zu fangen. Die Spieler beginnen mit drei Leben, flieht ein Alien wird ein Leben abgezogen. Für jedes gefangene Alien bekommen Spieler einen Punkt. Das Spiel endet, wenn alle Leben verbraucht sind. Abbildung 1 zeigt die Visualisierung einer Beispielimplementierung.

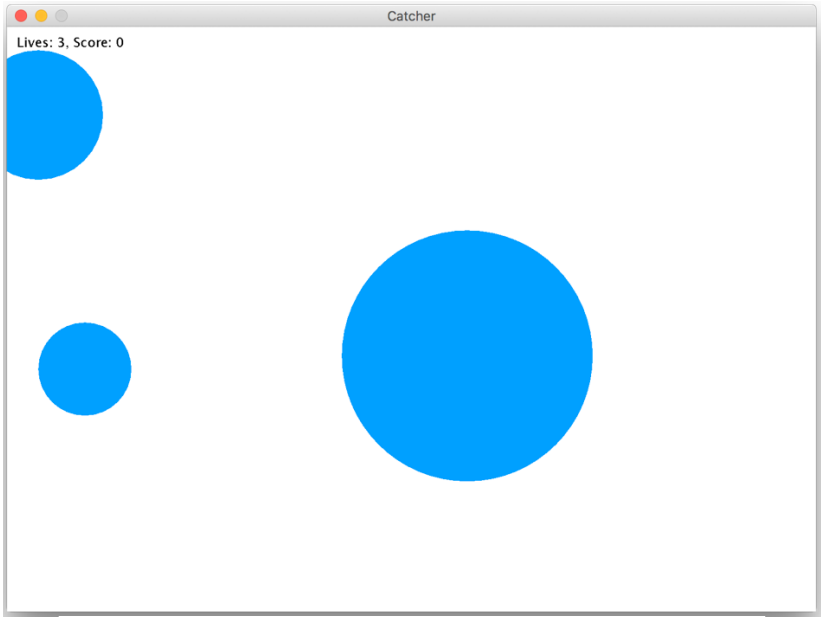

Abbildung 1) Alien Catcher mit drei Aliens.

Aufgabe 10.a) Spiel-Implementierung mit Blobs (Aliens als Kreise) (8 Punkte)

In der Klasse *Main* wird das Spielmodell (*GameModel*) und Spielfeld (*JCatcherField*) angelegt und einem *JFrame* angezeigt. Damit das Spiel animiert abläuft, wird ein Timer (*javax.swing.Timer*) genutzt, der periodisch alle 50ms das Modell aktualisiert und das Spielfeld neu zeichnet (Methode *update()*).

Implementieren Sie das Interface *Alien* in einer Klasse *Blob*, die ein Alien auf einen blau gefüllten Kreis abbildet. Ein Alien wird getroffen, wenn der Benutzer in den Kreis klickt. Bei jedem Update muss der Radius des Kreises um eins (1) verringert werden. Ist der Radius kleiner als zehn (10), dann gilt der Blob als geflohen.

Implementieren Sie die Klasse *GameModel*. Bei einem Update des Modells müssen alle Aliens „upgedated“ werden. Falls ein Alien geflohen ist oder gefangen wurde, muss es aus dem Modell entfernt werden. Sind nach dem Update weniger als drei (3) Aliens im Modell muss ein neuer Blob erzeugt werden. Die Position des Blobs soll zufällig im vorgegebenen Bereich ($x = 0$ bis *width* und $y = 0$ bis *height*) angelegt werden, der Radius soll zwischen 50 und 150 zufällig gewählt werden.

Aufgabe 10.b) Factory für Blobs (4 Punkte)

Implementieren Sie das Interface *AlienFactory* für Blobs in einer Klasse *BlobFactory*. Die *create*-Methode hat Parameter für die Maximalwerte der Position (*maxX* und *maxY*) als Minimalwerte gilt weiterhin null (*0*). Schieben Sie den Code, zum Erzeugen von Blobs, den Sie in Aufgabe 10.a entwickelt haben, in die *create*-Mehtode und verwenden Sie im *GameModel* die *BlobFactory* anstelle der direkten Erzeugung.

Aufgabe 10.c) BlobFactory über Singleton anbieten (3 Punkte)

Erstellen Sie ein Enum *FactorySingleton* (implementiert *AlienFactory*) mit dem Feld *ALIEN*, um auf das Singleton-Objekt zuzugreifen. Benutzen Sie intern eine *BlobFactory*, um die Blobs zu erzeugen. Das Singleton soll über ein Enum realisiert werden, um sicherzustellen, dass das Singleton-Objekt über die Laufzeit unveränderlich ist.

Aufgabe 10.d) Dekorator für Aliens (6 Punkte)

Implementieren Sie eine Klasse *AlienDecorator* als Basis für Dekoratoren. Diese Klasse muss alle Aufrufe an das dekorierte Objekt weiterleiten. Implementieren Sie damit einen konkreten Dekorator *Shield*, der die Methode *hit(int x, int y)* wie folgt überschreibt: falls der Klick getroffen hätte (*isHit(int x, int y)*), speichert der Dekorator die Position des Treffers und erst ab drei Treffern ruft er die *hit*-Methode des dekorierten Aliens auf. Außerdem implementiert der Dekorator die *paint*-Mehtode, um nach dem Aufruf der *super.paint*-Mehtode die Treffer mit kleinen schwarzen Kreisen markiert (wenn die Treffer noch innerhalb des Aliens sind, was sich ja durch die Größenänderung ändern kann, siehe Abbildung 2). *FactorySingleton* soll nun die *Blob*-Objekte, welche von der internen *BlobFactory* erzeugt werden, mit dem *Shield*-Decorator versehen.

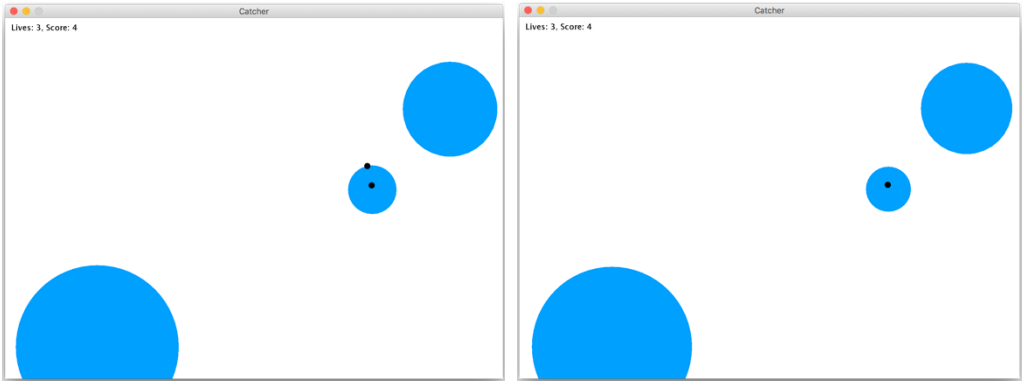

Abbildung 2) Das kleine mittlere Alien hat zwei Treffer (links), davon ist, nachdem das Alien geschrumpft ist nur mehr einer sichtbar (rechts).

Aufgabe 10.e) JavaDoc (3 Punkte)

Versehen Sie alle Klassen und Methoden mit JavaDoc-Kommentaren.