Übungsblatt: Interfaces, Erweiterungen, Rekursion, Überladen

**Aufgabe 1: HangMan-Spiel 40 Punkte**  
Schreiben Sie eine Klasse, die alle nötigen Methoden für das HangMan-Spiel zur Verfügung stellt. Als Vorgabe dient das Interface *HangManGame*, welches Sie auf der Vorlesungshomepage herunterladen können. Importieren Sie es in Ihr Projekt und implementieren Sie es in Ihrer Klasse. Schauen Sie sich das Interface an und überlegen Sie welche Attribute nötig sind, um den Zustand des Spiels zu beschreiben. Einige Hinweise und Tipps finden Sie in den Quellcode Kommentaren des Interfaces. *HangManGame* stellt auch zwei Konstanten zur Verfügung, die Sie in Ihrer Klasse verwenden sollen. Die möglichen Begriffe, die im Spiel vorkommen können, sollen aus einer Datei (*Dictionary*) eingelesen werden. Beim Starten des Spiels wird dann zufällig ein neuer Begriff ausgewählt. Zusätzlich soll das Abfragen, ob das Spiel gewonnen oder verloren wurde, möglich sein. Ist einer der beiden Fälle eingetreten so ist das Spiel beendet und sollte auch keine weiteren Rateversuche entgegen nehmen. Wenn Sie sich exakt an das Interface halten und Ihre Spiele-Logik funktioniert, können Sie es am Ende über eine Swing-Benutzeroberfläche spielen.

1. Implementieren Sie einen Konstruktor, der als Parameter den Pfad zum *Dictionary* bekommt.
2. Implementieren Sie alle vom Interface vorgegebenen Methoden.
3. Laden Sie *hangManGUI.jar* von der Vorlesungshomepage herunter und binden Sie es wie die *MakeItSimple* in Ihr Projekt ein. Erzeugen Sie in Ihrer main-Methode eine GUI-Instanz, in dem Sie den Konstruktor *HangManMainWindow(HangManGame game)* aufrufen und eine Instanz Ihrer geschriebenen Klasse übergeben.

**Hinweise:**

* Sie dürfen eigene Methoden definieren, die dann aber eine eingeschränkte Sichtbarkeit haben sollen.
* Für die Generierung einer Zufallszahl darf die Java Funktion *Math.random()* benutzt werden, welche einen *double* Wert 0.0 <= x <= 1.0 liefert.

**Korrekturhinweise:**

* Wie wird ein Begriff *zufällig* ausgewählt? Berechnung einer Integer Zufallszahl? Wie genau? Oder anders…?
* Wie wird auf *das Dictionary* zugegriffen? Wird es im Hauptspeicher gehalten?
* Welche Attribute wurden definiert? Sinnvoll?
* Ist die Methode *addChar(char c)* case insensitiv? Wie wurde es gelöst?
* Die positiven Aufrufe der Methoden *won()* und *lost()* müssen sich gegenseitig ausschließen, d.h. es dürfen nie beide Methoden *TRUE* liefern.
* Kann man noch weitere Buchstaben raten, obwohl das Spiel bereits gewonnen oder verloren wurde?
* Wird bei einem verlorenen Spiel der wirkliche Suchstring, also ohne versteckte Buchstaben, geliefert?
* Wurden die Konstanten sinnvoll eingesetzt?

**Aufgabe 2: HangMan-Erweiterung 10 Punkte**Das Spiel soll so erweitert werden, dass der gesuchte Begriff auch vom Benutzer eingegeben werden kann und nicht immer nur aus dem *Dictionary* gelesen wird. Durch diese Erweiterung kann man das Spiel dann auch mit zwei Leuten spielen. Für die Fachlogik heißt das, wir benötigen eine weitere Möglichkeit, das Spiel zu starten. Auch hier gibt es als Vorgabe ein Interface *HangManGameExt* welches von *HangManGame* erbt und dieses um lediglich eine weitere Methode erweitert.

1. Schreiben Sie eine Klasse, die das erweiterte Interface HangManGameExt implementiert. Auch Sie soll wieder einen Konstruktor mit einem Parameter besitzen, nämlich den Pfad zum *Dictionary*.
2. Um das erweiterte HangMan-Spiel zu starten, instantiieren Sie die Benutzeroberfläche, in dem Sie den Konstruktor *HangManMainWindowExt(HangManGameExt game)* aufrufen und ihm eine Instanz von Ihrer erweiterten HangMan Klasse übergeben.

**Korrekturhinweise:**

* Wurde Vererbung eingesetzt?
* Wird der Superklassen-Konstruktor aufgerufen? Konstruktoren werden in Java nicht vererbt…
* Wie modular ist der Code? Können viele Teile wiederverwendet werden oder gibt es größere Redundanzen?

**Aufgabe 3: Integer Zahl umdrehen 15 Punkte**Schreiben Sie eine rekursive Funktion mit einem Parameter, die eine Integer Zahl umdreht. Auch negative Werte sollen umgedreht werden.

Bsp.: 123 🡪 321 1230 🡪 321 123321 🡪 123321 -234 🡪 -432

1. Schreiben Sie eine statische Funktion *reverseInt(int num)*, die eine Integer Zahl umdreht.
2. Erstellen Sie ein kleines Hauptprogramm, mit dem man die Funktion im Testat prüfen kann. (Benutzer kann Zahl eingeben 🡪 Zahl wird umgedreht 🡪 Umgedrehte Zahl wird ausgegeben)

**Hinweise:**

* Es dürfen keine Strings verwendet werden.
* Sie dürfen die Funktion überladen.

**Korrekturhinweise:**

* Wurde die Funktion (sinnvoll) überladen?
* Wie/ Wo sind die einzelnen Rekursionsschritte implementiert? (Rekursionsstart, Rekursionsschritt, Rekursionssabbruch)
* Wie wird auf die einzelnen Ziffern zugegriffen? Modulo?