# Benutzerhandbuch

## Allgemein:

Beim Projekt handelt es sich das Programmierspiel Core War (Krieg der Kerne). Ein ausführliches Handbuch für Einsteiger liegt von Sascha Zapf vor (2003). Nachfolgend werden vorgenommenen Änderungen am Quellcode erfasst.

### Klasse: jMARS.java

In der Klasse *jMARS.java* werden zurzeit sämtliche Teile der Software durchgeführt (Programmstart, Durchführung Schleife und Ausgabe der Statistik).

1. Zur Vereinfachung des Codes wurde die args-Verarbeitung, welche durch if/else-Bedingungen abgehandelt wurde, mit einem switch-Statement ersetzt.
2. Es wurde eine Config-Klasse (*Config.java*) *erstellt, welche die Programmargumente verarbeitet und als Objekt anschließend zur Verfügung stellt*
3. Klasse Result wird für Methodenaufrufe in *jMARS.java verwendet*.

### Klasse: Result.java

Ermöglicht das Arbeiten mit Result Objekten anstelle des direkten Objekts. Hier wird das eigentliche Methodenergebnis im Result-Objekt verpackt. Anhand des boolean *isOk* wird erkannt ob die Durchführung erfolgreich war. Falls es erfolgreich war gibt *getValue()* das Objekt zurück. War es nicht erfolgreich kann über *getMessage()* die Fehlermeldung abgerufen werden. Somit werden keine Exceptions geworfen und etwaige Fehlermeldung werden bis zu oberster Ebene hochgereicht.

### Klasse: WarriorsManager.java

In der Klasse *jMARS.java* werden zurzeit sämtliche Teile der Software durchgeführt. Zur Entkopplung wurde eine Klasse WarriorsManger implementiert, welche für alle Operationen der Warriors zuständig ist. So kann die Methode *LoadWarriors() oder LoadWarriorsFromFile verwendet werden um alle benötigten WarriorObj zu generieren.*

Die Klasse wird zum Laden der Warriors verwendet. Dabei wird in der Methode *LoadWarrior()* hierfür die Anzahl der der Warrior aufgerufen und ein neuer Warrior erstellt. Die neue Anzahl wird dann am Ende an die Main-Methode zurückgegeben.

### Klasse: WarriorObj.java

In der Klasse *jMARS.java* wurden die Variablen für die Werte eines Warriors mithilfe eines Konstruktors und 4 Getter-Methoden initialisiert. Da dies unnötig ist, wurde der Konstruktor in *WarriorObj.java* überladen und in diesem die Variablen mit übergeben. Somit spart man sich das Aufrufen dieser Methoden in *jMARS.java* und erstellt die neue Klasse direkt im Konstruktor.

### Klasse: AsTest.java

Hier wurden Teile des Codes in eigene Funktionen ausgelagert. Ebenfalls werden die Methodenergebnisse als Result verpackt an die Hauptmethode zurückgegeben. Anstatt einzelne Konsolenausgaben zu machen, werden die Textteile mit einem StringBuilder zusammengesetzt und einmalig als Ganzes ausgegeben.

### Klasse jMARSTest.java

Hier wurde die Zeit von Thread.sleep() von 500ms auf 2000ms angehoben, da das Assert ansonsten geprüft wird, ohne dass die Anwendung fertig durchlaufen ist.

### Klasse Config.java

In diese Klasse werden die Programmargumente verarbeitet und in Instanzvariablen gespeichert. Das anschließende Config-Objekt wird durch die Anwendung durchgereicht, sodass die Konfigurationen an allen benötigten Stellen zur Verfügung stehen. Dadurch fielen viele Instanzvariablen in der jMARS-Klasse weg.

Der Default-Konstruktor enthält die Standardeinstellungen. Damit die Programmargumente verwendet werden können, steht eine statische Methode bereit.