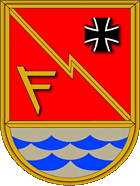
Schule Informationstechnik

der Bundeswehr



**Sprachausbildung Java**



**Projektaufgabe**

**"Weltrennschneckenverband (WRSV)"**

# Vision:

In den letzten zwei Jahren hat die neue Sportart Schneckenrennen mit rasendem Tempo an Bedeutung gewonnen. Zu Beginn fanden sich ein paar Freunde zusammen, um ihre Schnecken auf einer einfachen Strecke gegeneinander kriechen zu lassen. Mittlerweile haben die Schneckenrennen das ganze Land in Euphorie versetzt, die Anzahl der Fans wächst rasant und es wird erwartet, dass sich der Trend der Schneckenrennen auch global verbreiten wird.

Aufgrund der schnellen Ausbreitung in kurzer Zeit, werden die Schneckenrennen weiterhin nur auf Zuruf veranstaltet. Der Trend hat sich zwar über das ganze Land ausgebreitet, die Schnecken treten aber überwiegend nur lokal, immer wieder gegen dieselben Gegner an. Das hat zur Folge, dass auch die Fans immer nur einen lokalen Favoriten haben und sich nicht stark für das überregionale Geschehen interessieren.

Wir sehen in dieser Situation unsere Chance aus dem regional zersplitterten Zustand einen landesweiten Rennschneckenverband zu erschaffen. Unser Ziel ist es eine Dachorganisation aufzubauen, welche den neuen Trendsport landesweit organisiert und vermarktet. Dies Umfasst die Registrierung, Verwaltung und Unterstützung von Rennschnecken, die Veranstaltung von unabhängigen Rennen sowie einer jährlichen Rennserie und die Registrierung und Zertifizierung von zugelassenen Wettbüros.

Wir versprechen uns davon die Bildung von organisierten Vereinen und damit einhergehend den Aufbau von überregionalen Fanclubs, welche die von uns organisierten Rennen besuchen.

Unser langfristiges Ziel ist die Wandlung vom landesweiten zum internationalen Verband für Schneckenrennen, falls sich der Hype, wie erwartet, global ausweitet.

# Situation:

Die Vorbereitungen zur Gründung des Verbandes und der zugehörigen Organisationen laufen bereits auf Hochtouren. Wir planen die Gründung und Bekanntmachung des Verbandes zum 02.08.2018. Das Eröffnungsrennen mit lokalen Favoriten aus unterschiedlichen Regionen wird am 15.08.2018 stattfinden.

Als kritisch betrachten wir zurzeit die Situation der Software für die Rennschnecken-, Rennen- und Wettbüroverwaltung. Aufgrund der Beratung durch einen Bekannten eines Mitarbeiters des Projektes sah die erste Planung die Nutzung von Office Programmen mit umfangreichen Skripts vor. Dieser hatte angeboten, die notwendigen Skripte zu erstellen. Im Laufe der Zeit konnte er allerdings immer mehr Abgabetermine nicht einhalten und ist seit 3 Monaten nicht mehr erreichbar. Aufgrund dieser Situation trete ich an Sie heran.

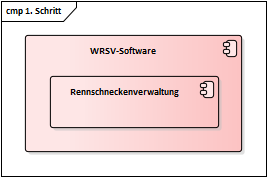
# Auftrag:

Ich beauftrage Sie mit der Erstellung einer Softwareumgebung, zur

1. Registrierung und Verwaltung von Rennschnecken,
2. Erstellung, Verwaltung, Durchführung und Reporting von unabhängigen Rennen,
3. Erstellung, Verwaltung, Durchführung und Reporting einer jährlichen Rennserie und
4. Registrierung, Verwaltung und Zertifizierung von Wettbüros.

Die Software soll in mehreren Implementierungsschritten ausgeliefert werden.

## Schritt (Rennschneckenverwaltung):

Im ersten Implementierungsschritt geht es vor allem um die Verwaltung der ersten Verbandsmitglieder. Die Rennschnecken sollen angelegt, bearbeitet und gelöscht werden können. Das System wird zu diesem Zeitpunkt nur von einem „Bearbeiter“ bedient.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S01-001: Das System muss eine Liste aller registrierten Rennschnecken bereitstellen und anzeigen.

WRSV-S01-002: Das System muss dem Bearbeiter die Möglichkeit bieten neue Rennschnecken zu registrieren.

WRSV-S01-003: Das System muss dem Bearbeiter die Möglichkeit bieten Eigenschaften von registrierten Rennschnecken zu bearbeiten.

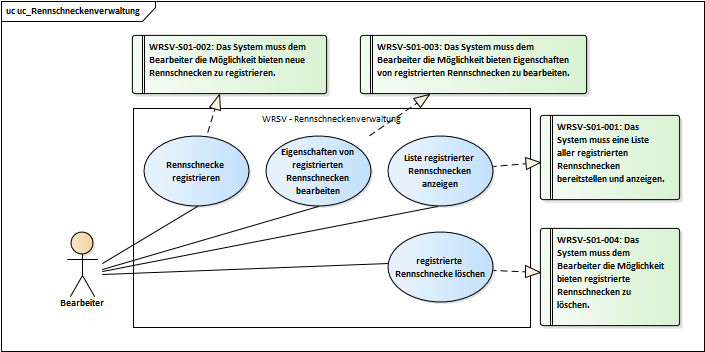
WRSV-S01-004: Das System muss dem Bearbeiter die Möglichkeit bieten registrierte Rennschnecken zu löschen.

WRSV-S01-005: Die Liste der registrierbaren Rennschnecken muss mindestens 20 Plätze groß sein.

### Anwendungsfälle (Use-Cases):

Aus den Anforderungen ergeben sich die folgenden Anwendungsfälle für den Bereich „Rennschneckenverwaltung“.

#### Anwendungsfälle Rennschnecken Verwaltung:



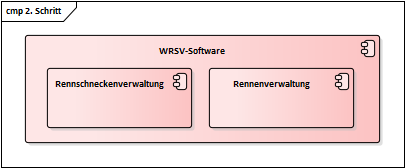
### Implementierungsdetails:

* Rennschnecken haben:
  + eine eindeutige ID
  + einen Namen
  + eine maximale Geschwindigkeit
  + sind über den Verband versichert oder nicht.

### Implementierungshilfen:

1. Beginnen Sie mit der Modellierung (per Hand oder Enterprise Architect) der Modellklassen.
   1. Überlegen Sie welche Attribute (und Attributtypen) benötigt werden.
   2. Überlegen Sie welche Methoden benötigt werden, vor allem in Verbindung mit den Sichtbarkeiten.
2. Modellieren Sie Klassen für die Datenhaltung der Rennschnecken.
   1. Welche Attribute und Methoden könnten Sie benötigen?
3. Skizzieren Sie sich den Programmablauf, vom Start der Anwendung bis zu den einzelnen Operationen (beginnen Sie mit den einfacheren Operationen).
   1. Denken Sie währenddessen über die Darstellung nach und skizzieren Sie ihre Ideen.
4. Sobald Sie alle Schritte skizziert haben und anhand dessen gedanklich durch Ihre Anwendung gehen können, beginnen Sie mit der Implementierung.
   1. Bei der Implementierung gibt es zwei große Ansätze, Top-Down (vom Ablauf zum Detail) oder Bottom-Up (vom Detail zum Ablauf), es ist aber auch eine Mischung möglich, bei der man Funktionalitätsweise (nacheinander einzelne Use-Cases implementieren) vorgeht.
5. Testen Sie Ihre Anwendung ausgiebig.

## Schritt (Rennverwaltung):

Die erste Erweiterung umfasst das Erstellen und Verwalten von einzelnen Rennen. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

Die Software lässt sich zu diesem Zeitpunkt in die Komponenten „Rennschneckenverwaltung“ und „Rennenverwaltung“ aufteilen.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S02-010: Das System muss eine Liste von Rennen bereitstellen und anzeigen.

WRSV-S02-011: Das System muss dem Bearbeiter die Möglichkeit bieten neue Rennen zu erstellen.

WRSV-S02-012: Das System muss dem Bearbeiter die Möglichkeit bieten Eigenschaften von erstellten Rennen zu bearbeiten.

WRSV-S02-013: Das System muss dem Bearbeiter die Möglichkeit bieten Rennschnecken an Rennen anzumelden.

WRSV-S02-014: Das System muss dem Bearbeiter die Möglichkeit bieten Rennschnecken von Rennen abzumelden.

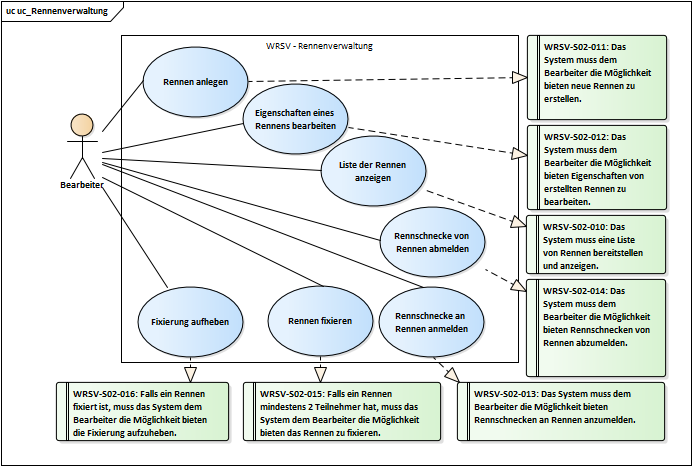
WRSV-S02-015: Falls ein Rennen mindestens 2 Teilnehmer hat, muss das System dem Bearbeiter die Möglichkeit bieten das Rennen zu fixieren.

WRSV-S02-016: Falls ein Rennen fixiert ist, muss das System dem Bearbeiter die Möglichkeit bieten die Fixierung aufzuheben.

WRSV-S02-017: Die Rennenverwaltung muss sicherstellen, dass sich der Zustand eines fixierten Rennens nicht verändern kann. Die einzige Ausnahme ist die Aufhebung der Fixierung.

WRSV-S02-018: Die Liste der Rennen muss mindestens 30 Plätze groß sein.

#### Anwendungsfälle Rennen Verwaltung:



### Wünsche der Anwender:

Hier folgen einige Wünsche der späteren Anwender. Diese sind keine zwingenden Anforderungen und deshalb nicht unbedingt notwendig. Sie können aber helfen, um Designentscheidungen zu treffen.

1. Es soll nur eine Anwendung für die beiden Verwaltungsbereiche geben.
2. Nach dem Start der Anwendung, soll diese in einem Dialog abfragen, in welchem Bereich man arbeiten möchte.
3. Nach dem Wechsel in einen Bereich, sollen die möglichen Operationen dargestellt und abgefragt werden, welche man ausführen möchte.
   1. Falls noch Platz ist könnten noch die aktuellsten Einträge in der Rennschnecken/Rennenliste dargestellt werden.
   2. Falls nicht genug Platz vorhanden ist, kann man die Liste ja immer noch über die entsprechende Operation öffnen.

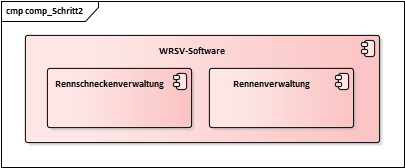
### Implementierungsdetails:

* Rennen haben:
  + eine eindeutige ID
  + einen Namen
  + eine Streckenlänge
  + einen Ort
  + Verweise auf die Teilnehmenden Rennschnecken (max. 8 Teilnehmer pro Rennen).
* Rennen haben die Zustände:
  + angelegt
  + fixiert (Name, Ort, Strecke und Teilnehmer können nicht mehr geändert werden; es werden mindestens 2 Teilnehmer benötigt, um das Rennen zu fixieren)

### Implementierungshilfen:

1. Beginnen Sie mit der Modellierung der Modellklassen.
   1. Überlegen Sie welche Attribute (und Attributtypen) benötigt werden.
   2. Überlegen Sie welche Methoden zusätzlich benötigt werden, vor allem in Verbindung mit den Sichtbarkeiten.
2. Modellieren Sie Klassen für die Datenhaltung der Rennen.
   1. Welche Bestandteile / Funktionalitäten wiederholen sich aus Schritt 1 und lassen sich eventuell wiederverwenden?
   2. Welche Attribute und Methoden könnten Sie benötigen?
3. Skizzieren Sie sich den Programmablauf, vom Start der Anwendung bis zu den einzelnen Operationen (beginnen Sie mit den einfacheren Operationen).
   1. Denken Sie währenddessen über die Darstellung nach und skizzieren Sie ihre Ideen.
4. Sobald Sie alle Schritte skizziert haben und anhand dessen gedanklich durch Ihre Anwendung gehen können, beginnen Sie mit der Implementierung.
5. Testen Sie Ihre Anwendung ausgiebig.

## Schritt (Sortieren / Filtern & Dynamische Datenstrukturen):

Die zweite Erweiterung umfasst die Sortierung und Filterung der Listendarstellungen. Des Weiteren soll die Datenhaltung mit Hilfe von dynamischen Datenstrukturen flexibler gestaltet werden. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S03-005: Die Liste der registrierbaren Rennschnecken muss mit Hilfe von dynamischen Datenstrukturen flexibel (ohne min-/max-Werte) auf Größenänderungen reagieren können.

WRSV-S03-018: Die Liste der Rennen muss mit Hilfe von dynamischen Datenstrukturen flexibel (ohne min-/max-Werte) auf Größenänderungen reagieren können.

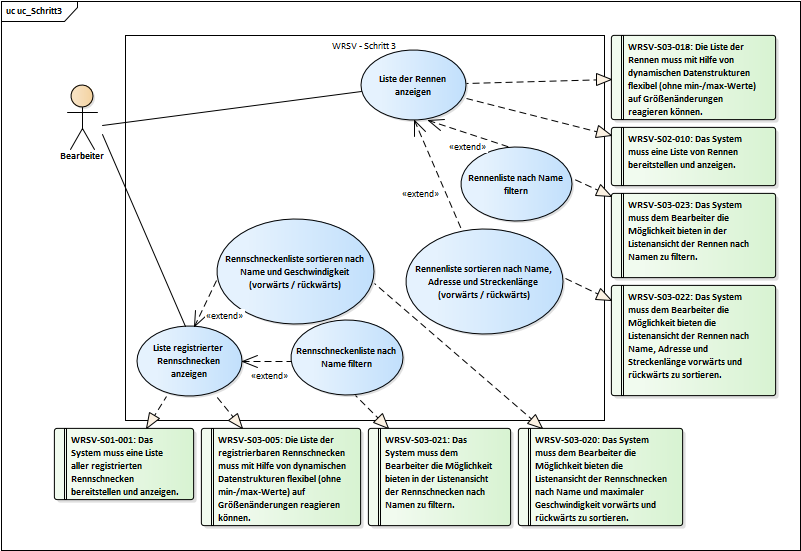
WRSV-S03-020: Das System muss dem Bearbeiter die Möglichkeit bieten die Listenansicht der Rennschnecken nach Name und maximaler Geschwindigkeit vorwärts und rückwärts zu sortieren.

WRSV-S03-021: Das System muss dem Bearbeiter die Möglichkeit bieten in der Listenansicht der Rennschnecken nach Namen zu filtern.

WRSV-S03-022: Das System muss dem Bearbeiter die Möglichkeit bieten die Listenansicht der Rennen nach Name, Adresse und Streckenlänge vorwärts und rückwärts zu sortieren.

WRSV-S03-023: Das System muss dem Bearbeiter die Möglichkeit bieten in der Listenansicht der Rennen nach Namen zu filtern.

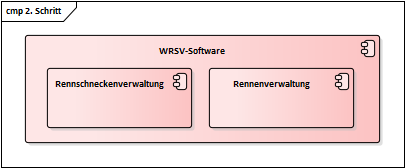
#### Anwendungsfälle Filtern und Sortieren:



### Implementierungshilfen:

1. Beginnen Sie mit der Modellierung der neuen dynamischen Datenhaltung und ersetzen Sie die alte Datenhaltung.
   1. Falls noch nicht geschehen, eignet sich dieser Schritt, um eine einheitliche Schnittstelle für die Datenhaltung zu implementieren.
2. Fahren Sie mit den Sortiermethoden fort.
   1. Überlegen Sie sich einen zu verwendenden Sortieralgorithmus. Zum Beispiel:
      1. InsertionSort
      2. BubbleSort
      3. QuickSort
      4. …
   2. Planen Sie den Algorithmus mit einem Aktivitätsdiagramm.
   3. Realisieren Sie das Sortieren der Listen nach den geforderten Attributen.
3. Fahren Sie mit den Filtermethoden fort.
   1. Überlegen Sie sich einen zu verwendenden Filteralgorithmus.
   2. Planen Sie den Algorithmus mit einem Aktivitätsdiagramm.
   3. Implementieren Sie den Filteralgorithmus.
4. Skizzieren Sie sich den Programmablauf, innerhalb der Listenansicht in Bezug auf die Filterung und das Sortieren.
   1. Denken Sie währenddessen über die Darstellung nach und skizzieren Sie ihre Ideen.
5. Sobald Sie alle Schritte skizziert haben und anhand dessen gedanklich durch Ihre Anwendung gehen können, beginnen Sie mit der Implementierung.
6. Testen Sie Ihre Anwendung ausgiebig.

## Schritt (Generische Klassen):

Schritt 4 ist ein Update hinter den Kulissen, um neu erlernte Sprachmittel zu vertiefen und das Design flexibler gestalten zu können. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S04-030: Die vielen unterschiedlichen Listen der WRSV-Software müssen eine generische Listenimplementierung als Basis nutzen.

WRSV-S04-031: Die generische Listenimplementierung muss iterierbar sein.

WRSV-S04-032: Die generische Listenimplementierung muss sortiertes Einfügen ermöglichen und dafür eine natürliche Ordnung der Listenelemente fordern.

WRSV-S04-033: Die generische Listenimplementierung muss das Sortieren mit Comparatoren ermöglichen.

WRSV-S04-034: Die generische Listenimplementierung muss eine öffentliche Schnittstelle zum Filtern, mit Hilfe eines Predicates, anbieten.

WRSV-S04-035: Das System muss die neuen Möglichkeiten zum Sortieren und Filtern nutzen. (Siehe WRSV-S04-032, WRSV-S04-033 und WRSV-S04-034.)

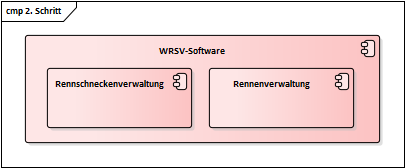
#### Anwendungsfälle Schritt 4:

In diesem Schritt gibt es keine neuen Anwendungsfälle.

### Rahmenbedingungen / Implementierungshilfen:

1. Beginnen Sie mit der Modellierung der neuen generischen Datenhaltung und ersetzen Sie die alte Datenhaltung.
   1. Es sollte, von außen, kein direkter Zugriff auf die Datenhaltung möglich sein (Datenkapselung, Schnittstellen). Erstellen Sie ein Klassendiagramm mit allen beteiligten Klassen.
   2. Welche Interfaces / Klassen benötigen Sie für etwas Iterierbares?
2. Sortiertes Einfügen in eine generische Liste:
   1. Welche Probleme entstehen beim sortierten Einfügen in die generische Liste?
   2. Was ist eine natürliche Ordnung und welches Interface bildet diese ab?
   3. Wie kann die natürliche Ordnung in die Liste mit übernommen werden?
   4. Modellieren Sie die Funktionalität mit einem Aktivitäts- oder Sequenzdiagramm.
3. Sortieren nach anderen Kriterien:
   1. Wie helfen Ihnen Comparatoren beim Sortieren einer Liste?
   2. Wie sehen diese aus und warum ermöglichen sie Ihnen das Sortieren auf möglichst abstrakter Ebene?
   3. Modellieren Sie die Funktionalität mit einem Aktivitäts- oder Sequenzdiagramm.
4. Filtern von Listen:
   1. Überlegen Sie ob sie Ihre ursprüngliche Liste erhalten wollen oder nicht.
   2. Überlegen Sie wie sie die Methode filter möglichst abstrakt realisieren können, sehen Sie sich dazu das Interface Predicate an.
   3. Modellieren Sie die Funktionalität mit einem Aktivitäts- oder Sequenzdiagramm.
5. Implementieren Sie alle notwendigen Änderungen.
6. Testen Sie Ihre Anwendung ausgiebig!

## Schritt (Collections):

Schritt 5 ist ein Update hinter den Kulissen, um neu erlernte Sprachmittel zu vertiefen und die Verwendung der Standardbibliothek „Collections“ zu üben. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

### Rahmenbedingungen:

**Nummer: Beschreibung:**

WRSV-S05-040: Die generische Liste muss durch eine Liste des Java Collections Framework ersetzt werden.

WRSV-S05-041: Das System muss die Methoden Collections.sort() und Collections.sort(Comparator) zum Sortieren benutzen.

WRSV-S05-042: Die Verwaltungsklassen für Rennen und Rennschnecken sollen weiterhin die Funktionalität „Filtern“ zur Verfügung stellen und dazu Predicates entgegen nehmen.

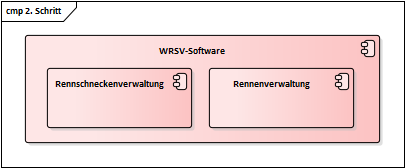
#### Anwendungsfälle Schritt 5:

In diesem Schritt gibt es keine neuen Anwendungsfälle.

### Implementierungshilfen:

1. Liste aus dem Collections Framework.
   1. Sehen Sie sich die unterschiedlichen Listen im Collections Framework an und notieren Sie sich Vor- und Nachteile.
   2. Entscheiden Sie sich für eine der Implementierungen.
2. Sortieren:
   1. Wie werden Listen des Collections Frameworks sortiert?
3. Filtern von Listen:
   1. Welche Möglichkeit bietet Java 8 um Collections zu filtern?
   2. Welche Vor-/Nachteile hat diese Variante?
   3. Verschieben Sie Ihre filter Methode aus der generischen Liste in die Verwaltungsklasse.
4. Implementieren Sie alle notwendigen Änderungen.
5. Testen Sie Ihre Anwendung ausgiebig!

## Schritt (Exceptions):

Schritt 6 ist ein Update hinter den Kulissen, um neu erlernte Sprachmittel zu vertiefen und die Verwendung von Exceptions zu üben. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

### Anforderungen / Rahmenbedingungen:

**Nummer: Beschreibung:**

WRSV-S06-050: Bei Nutzereingaben muss das System auftretende Fehler abfangen, sinnvolle Fehlermeldung ausgeben und nach einer erneuten Eingabe fragen.

WRSV-S06-051: Bei der Nutzung von Comparatoren muss das System entsprechend der JavaAPI von Comparator auf fehlerhafte Übergabewerte reagieren.

WRSV-S06-052: Das Modell muss fehlerhafte Methodenaufrufe sofort mit entsprechenden Sprachmitteln abbrechen.

WRSV-S06-053: Alle Aufrufer des Modells müssen eventuell auftretende Ausnahmen abfangen und behandeln.

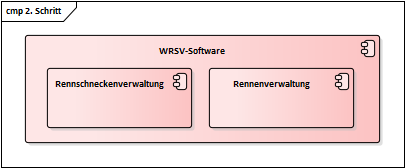
#### Anwendungsfälle Schritt 6:

In diesem Schritt gibt es keine neuen Anwendungsfälle.

### Implementierungshilfen:

1. Nutzereingaben:
   1. Wo finden Nutzereingaben statt?
   2. Welche Nutzereingaben können zu Fehlern führen? Wo werden eventuell Exceptions verursacht?
2. Comparator:
   1. Welche Werte sollten eine Exception innerhalb eines Comparators auslösen?
3. Modell:
   1. Welche Methoden sollten abgesichert werden?
   2. Welche existierenden Exceptions haben die richtige semantische Bedeutung?
   3. Was müssen die aufrufenden Methoden machen? Wie müssen sie angepasst werden?
4. Implementieren Sie alle notwendigen Änderungen.
5. Testen Sie Ihre Anwendung ausgiebig!

## Schritt (Files and Streams):

In Schritt 7 sollen die Rennschnecken- sowie die Rennenverwaltung um die Möglichkeit, den aktuellen Zustand auf einem nichtflüchtigen Medium zu speichern, erweitert werden. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S07-060: Die Rennschneckenverwaltung muss dem Bearbeiter die Möglichkeit bieten den aktuellen Zustand in eine Datei auf der Festplatte zu speichern.

WRSV-S07-061: Die Rennenverwaltung muss dem Bearbeiter die Möglichkeit bieten den aktuellen Zustand in eine Datei auf der Festplatte zu speichern.

WRSV-S07-062: Die Rennschneckenverwaltung muss dem Bearbeiter die Möglichkeit bieten einen gespeicherten Zustand aus einer Datei auf der Festplatte zu lesen.

WRSV-S07-063: Die Rennenverwaltung muss dem Bearbeiter die Möglichkeit bieten einen gespeicherten Zustand aus einer Datei auf der Festplatte zu lesen.

WRSV-S07-064: Beim Start der Anwendung muss das System die Dateinamen für die Rennschnecken- / Rennenverwaltung per Kommandozeilenparameter übergeben bekommen.

WRSV-S07-065: Falls beim Start der Anwendung keine Kommandozeilenparameter übergeben wurden, muss das System dem Bearbeiter eine Fehlermeldung: „Zuwenig / Falsche Argumente übergeben“ auf dem Bildschirm darstellen.

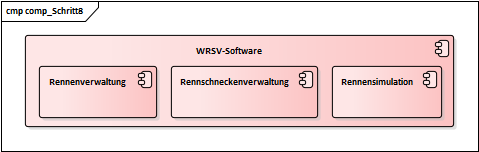
#### Anwendungsfälle Schritt 7:

### 

### Implementierungshilfen:

1. Speichern / Laden:
   1. Stellen Sie das Laden und Schreiben mit einem Sequenzdiagramm dar.
2. Implementieren Sie alle notwendigen Änderungen.
3. Testen Sie Ihre Anwendung ausgiebig!

## Schritt (Threads):

In Schritt 8 soll das System um einen Simulationsbestandteil erweitert werden. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S08-070: Die Rennensimulation muss dem Bearbeiter die Möglichkeit bieten angelegte Rennen zu simulieren.

WRSV-S08-071: Während der Simulation eines Rennens muss die Rennensimulation dem Bearbeiter zyklisch alle 5 Sekunden Name und Distanz des Rennens sowie erste und letzte Rennschnecke ausgeben.

WRSV-S08-072: Die Rennensimulation muss dem Bearbeiter die Möglichkeit bieten bis zu fünf Rennen gleichzeitig zu simulieren und anzuzeigen.

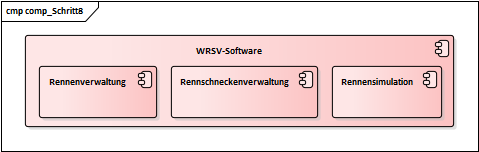
#### Anwendungsfälle Schritt 8:

### 

### Implementierungshilfen:

1. Simulation:
   1. 1 Thread pro Rennensimulation.
   2. 1 Thread für die Anzeige.
   3. Synchronisation über Synchronisationsobjekt.
   4. Planen Sie die Synchronisation mit Hilfe eines Aktivitäts-, Sequenz- oder Zustandsdiagramms.
2. Implementieren Sie alle notwendigen Änderungen.
3. Testen Sie Ihre Anwendung ausgiebig!

## Schritt (MVC / Observer):

In Schritt 9 soll die Rennsimulation im Sinne einer MVC Anwendung mit Hilfe vom Observer-Pattern verbessert werden. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S09-080: Die Rennsimulation muss entsprechend dem MVC-Pattern aufgebaut sein.

WRSV-S09-081: Die Rennsimulation muss das Observer-Pattern umsetzen.

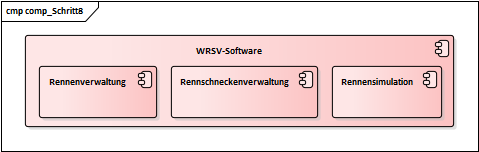
#### Anwendungsfälle Schritt 9:

In diesem Schritt gibt es keine neuen Anwendungsfälle.

### Implementierungshilfen:

1. MVC:
   1. Welche Bestandteile der Software repräsentieren den View, das Modell und den Controller?
   2. Teilen Sie ihre Anwendung entsprechend dem MVC-Muster auf.
2. Observer-Pattern:
   1. Welche Komponenten müssen wann von wem über neue Daten informiert werden?
   2. Entwerfen Sie das Zusammenspiel zwischen den Observern und den Observables.
3. Implementieren Sie alle notwendigen Änderungen.
4. Testen Sie Ihre Anwendung ausgiebig!

## Schritt (GUI):

In Schritt 10 soll eine auf Swing basierende grafische Benutzeroberfläche für die gesamte WRSV-Software entwickelt werden. Das System wird weiterhin nur von einem „Bearbeiter“ bedient.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S10-090: Das System muss dem Bearbeiter eine grafische Benutzeroberfläche zur Verfügung stellen.

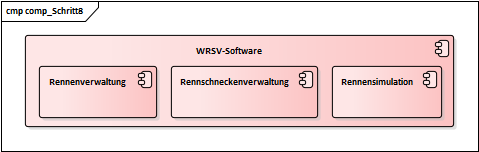
#### Anwendungsfälle Schritt 10:

In diesem Schritt gibt es keine neuen Anwendungsfälle.

### Implementierungshilfen:

1. GUI:
   1. Überlegen Sie zuerst, wie Sie die einzelnen Bestandteile der Software grafisch darstellen wollen und welche grafischen Elemente Sie dazu benötigen. (Erstellen Sie eine Skizze.)
   2. Sobald Sie die grafischen Elemente skizziert haben, machen Sie sich Gedanken über die eventbasierte Controllersteuerung.
   3. Skizzieren Sie die Steuerung in einem Aktivitätsdiagramm für besonders umfangreiche Fälle.
   4. Stellen Sie komplizierte Stellen in Sequenzdiagrammen dar.
2. Implementieren Sie alle notwendigen Änderungen.
3. Testen Sie Ihre Anwendung ausgiebig!

## Schritt (Netzwerk):

Hier fehlt noch eine Aufgabe für eine Netzwerkfähigkeit. Irgendwas mit Sockets (TCP und UDP).

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S11-100: Das System muss...

#### Anwendungsfälle Schritt 11:

In diesem Schritt gibt es keine neuen Anwendungsfälle.

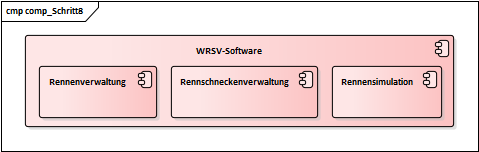
### Implementierungshilfen:

1. Netzwerk:
2. Implementieren Sie alle notwendigen Änderungen.
3. Testen Sie Ihre Anwendung ausgiebig!

Idee:

Für die Liveüberwachung eines Rennens existiert bereits eine kombinierte Hardware/Software Lösung. Diese erfasst den Fortschritt der Rennschnecken auf der Strecke über eine Kombination unterschiedlicher Sensoren und stellt die Informationen im Sekundenintervall bereit. Die Daten sollten anschließend von der Softwareumgebung zur Durchführung des Rennens analysiert und auf eine geeignete Weise dargestellt werden.

## Schritt (Datenbank):

Hier fehlt noch eine Aufgabenstellung für einfachen Datenbankzugriff.

### Anforderungen:

**Nummer: Beschreibung:**

WRSV-S12-110: Das System muss ….

#### Anwendungsfälle Schritt 12:

In diesem Schritt gibt es keine neuen Anwendungsfälle.

### Implementierungshilfen:

1. Datenbank:
   1. ...
2. Implementieren Sie alle notwendigen Änderungen.
3. Testen Sie Ihre Anwendung ausgiebig!