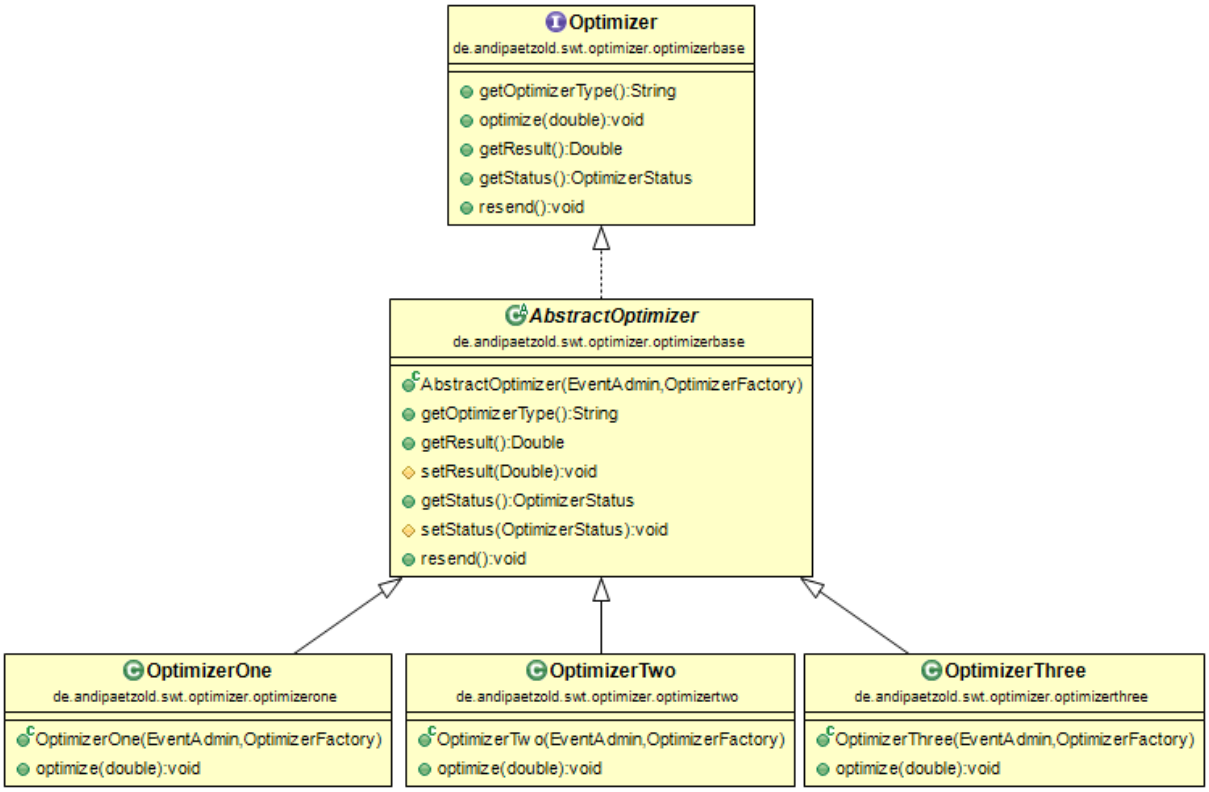
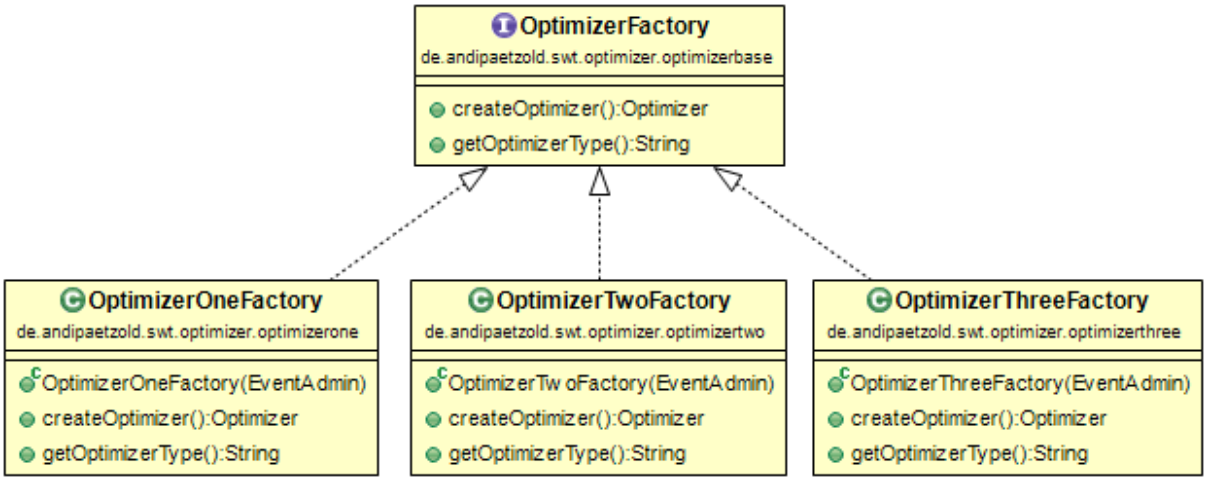
Übungsaufgabe 1 (JavaBeans & OSGi)

Architektur

Optimizer



OptimizerFactory



Module

Die Module sind sehr kompakt gehalten und beinhalten jeweil nur ein Package. Dementsprechend wird dieses importiert. Zudem wird ausschließlich das Modul *optimizer-base* importiert, da die restliche Kommunikation über *Services* oder *EventAdmin* geschieht. Die restlichen importierten Module sind das osgi-Framework, der Tracker für die Services oder EventAdmin. Deren Import ergibt sich aus der nachfolgenden Erläuterung der einzelnen Module.

optimizer-base

Dieses Modul enthält ausschließlich abstrakte Klassen und Interfaces und dient als Basis für die implementierten Optimierer.

AbstractOptimizer implementiert das Interface *Optimizer* und stellt bereits die grundlegenden Funktionen der Optimierer fest. Hierzu zählen die Verarbeitung der Stati sowie Ergebnisse und deren Verbreitung über den *EventAdmin*.

optimizer-one / optimizer-two / optimizer-three

Diese Module implementieren die zuvor erläuterten Basisklassen bzw. -interfaces. Die Optimierer enthalten lediglich eine Methode, die einen zufälligen Wert zurück gibt und den Thread davor einige Sekunden schlafen legt. Eine konkrete Implementierung war hier nicht gefordert.

Abgesehen von der Registrierung des *Services* und dem Erzeugen des *EventAdmins*, wird hier keinerlei Funktionalität implementiert. Dies geschieht in der zuvor erläuterten optimizer-base.

optimizer-manager

Der *Manager* verwaltet alle Optimierer (*OptimizerFactory*) die sich als Service registriert haben. Beim Starten eines Optimierungslaufs werden diese genutzt, um die entsprechenden Optimierer zu erzeugen. Jeder Optimierer wird in einem separaten Thread gestartet, um ein paralleles abarbeiten zu gewährleisten. Anhand dieser Threads kann auch festgestellt werden, ob ein neuer Optimierungslauf gestartet werden darf oder nicht.

Da die Kommunikation mit der Benutzeroberfläche über den *EventAdmin* abläuft, dient die Klasse *ManagerEventHandler* zum Empfangen der gesendeten Nachrichten.

optimizer-frontend

Die Benutzeroberfläche wurde mit JavaFX realisiert. Die Oberfläche wird auf Basis einer fxml-Datei erzeugt und mit einem Controller (*FrontendController*) verbunden. Dieser Controller enthält öffentliche Methoden zur Bearbeitung der Benutzeroberfläche. Hierüber können Optimierer hinzugefügt bzw. entfernt werden oder deren Stati und Ergebnisse aktualisiert werden.

Auf diese Funktionen wird über einen *EventHandler* zugegriffen, der die über den *EventAdmin* gesendeten Nachrichten verarbeitet. Dieser EventHandler erhält die Stati und Ergebnisse der Optimierer direkt aus den entsprechenden Klassen. Informationen über die verfügbaren Optimierer werden durch den Manager versendet.