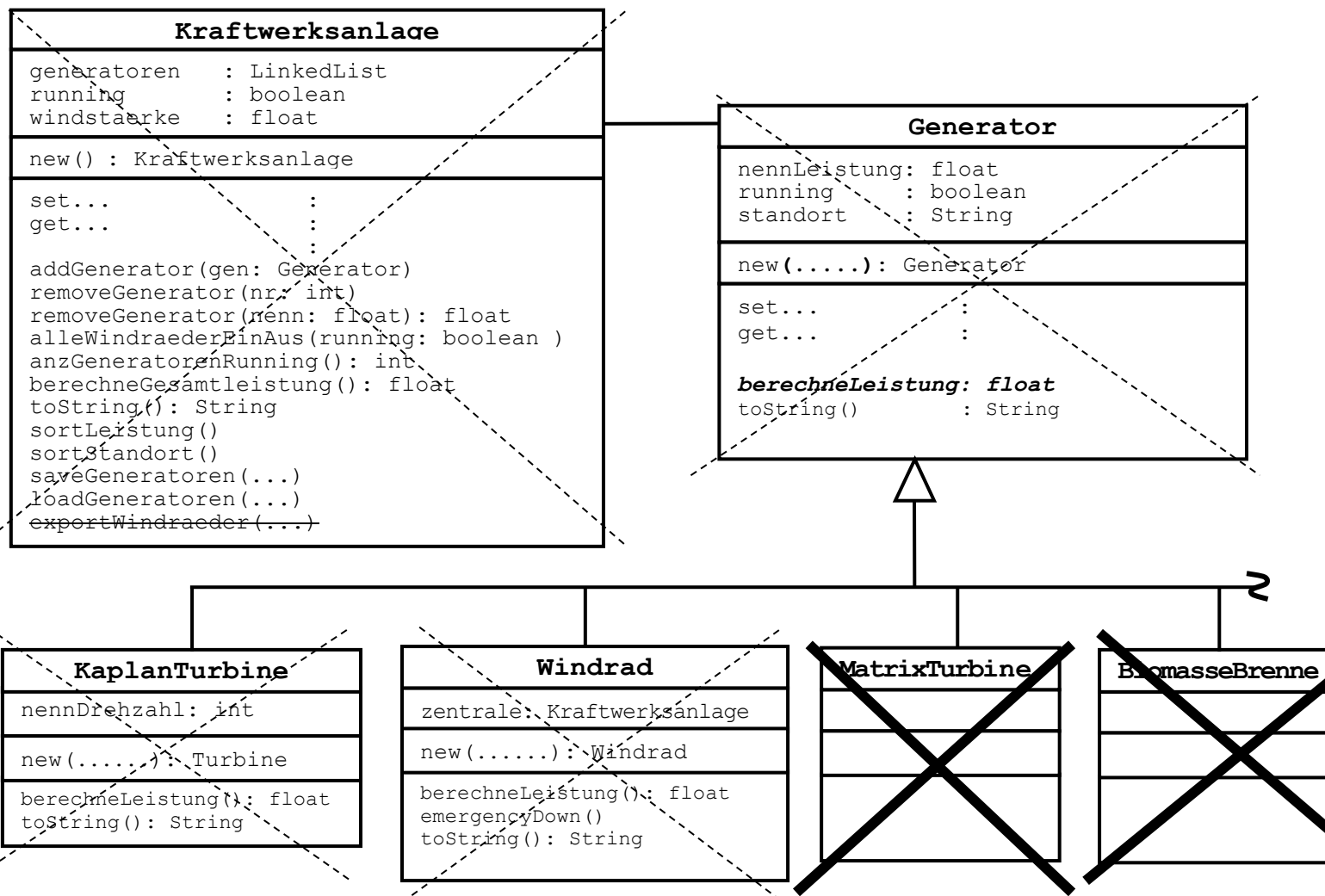


!!!!!!! Ihr Projektname in Eclipse: Nachname_Vorname !!!!!!!

Für eine neue Kraftwerk-Anlage, derzeit bestehend aus mehreren Windrädern sowie Strömungs-(Kaplan-) Turbinen, ist eine neue Steuerungs-Software zu erstellen. Die Applikation soll jederzeit um beliebige (derzeit ev. noch gar nicht bekannte) Generatoren-Typen erweitert werden können.

Implementieren Sie dafür folgende Klassen und Methoden (und gehen Sie hierbei von den auf Ihrem Desktop vorliegenden Basisklassen aus):



Die **get**- und **set**-Methoden zu den Attributen der jeweiligen Klassen sowie die Konstruktoren sind im Diagramm nicht bzw. nur ansatzweise dargestellt; sie sind jedoch **verpflichtend** !!

Sämtliche Fehler-/Probleme sind mittels sauberem Exception-Handling abzufangen!!

ACHTUNG !! Die Reihenfolge der Aufgaben ist unbedingt einzuhalten !!!

Aufgabe1: Implementieren Sie zur Fehlerbehandlung eine **KraftwerkException**.

Aufgabe2:

Generator

Konstruktor

übernimmt nur den Standort, sonst **KEINE** Werte!!!.
Neue "Generatoren" sind nie "running"!!!!

ACHTUNG!!

Die Plausibilitäts-Prüfungen für die den jeweiligen Konstrukteuren von **Windrad** bzw. **KaplanTurbine** übergebenen **nennLeistung**-Werte sind in der jeweiligen Klasse (set-Methoden!!) zu implementieren!!!!
Die korrekten Werte werden anschließend einfach der **set**-Methode der Generator-Klasse übergeben (die prüft natürlich nicht mehr) !

berechneLeistung() Muß zwingend bei der jeweiligen Generator-Art implementiert werden.

toString() Gibt Informationen über den **Generator** zurück. Der Standort wird auf 4 Stellen gekürzt!

Form: Standort ", Nennleistung " nennLeistung "MW, " Info-ob-läuft

Beispiel: Mödl, Nennleistung 0.5 MW, läuft
Gmun, Nennleistung 3.5 MW, steht

Aufgabe3:

KaplanTurbine

Konstruktor

übernimmt Werte für Standort, Nennleistung u. Nenn-Drehzahl.
Die Nennleistungen derzeitiger Kaplan-Turbinen liegen zw. 5 MW und max. 100 MW. Diese Nenn-Leistung wird bei der sogen. Nenn-Drehzahl erreicht, die bei heutigen Turbinen zw. 50 und 120 Umdrehungen pro Minute liegt.

berechneLeistung() Die aktuelle Leistung entspricht der Nenn-Leistung, sofern die Turbine "running" ist. // (einfache Matura-Variante!).

toString() Gibt Informationen über die Turbine zurück.

Form: "Kaplan:" stdort ", Nennleistung " nennLeistung "MW, " Info-ob-läuft " -> derzeit " Leistung " MW"

Beispiel: Kaplan: Mödl, Nenn-Leistung 10 MW, läuft -> derzeit 10 MW
Kaplan: Gmun, Nenn-Leistung 35 MW, steht -> derzeit 0 MW
Kaplan: Wien, Nenn-Leistung 25 MW, läuft -> derzeit 25 MW

Aufgabe3b:

TestKaplan

zum ausführlichen Testen dieser Klasse.

Aufgabe4:

Kraftwerksanlage

// für Windrad notwendiger Zwischenschritt !!!!!!!!!!!

Definieren Sie alle Attribute; implementieren Sie nur die Methoden:

getWindstaerke und **setWindstaerke** (vorerst ohne Plausibilitätsprüfung - siehe später Aufgabe6)!

alleWindraederEinAus(running) Schreiben Sie hier jetzt nur den Methoden-Kopf und einen komplett leeren Methoden-Rumpf(!), so daß die Methode für den Compiler überhaupt existiert (auch wenn sie zur Zeit noch nichts macht)!

Aufgabe5:

Windrad

Konstruktor

übernimmt Werte für Standort u. Nenn-Leistung. Windräder der aktuellen Generationen haben Nenn-Leistungen zw. 0.1 MW und 10 MW.

(Hinweis: Diese Nenn-Leistung wird bei der sogenannten Norm-Windgeschwindigkeit von 50 km/h erreicht.)

berechneLeistung() Die tatsächliche, aktuelle Leistung eines Windrades ist abhängig von der aktuellen Windstärke: pro km/h Differenz zur Norm - Windgeschwindigkeit ändert sich die Leistung gegenüber der Nenn-Leistung um 2%.

Achtung: Die Ausgangsleistung eines abgeschalteten Windrades ist natürlich 0.

toString() Gibt Informationen über das **Windrad** zurück.

Form: "Windrad:" stdort ", Nennleistung " nennLeistung "MW, " Info-ob-läuft " -> derzeit " Leistung " MW"

Beispiel: Windrad: Mödl, Nenn-Leistung 0.5 MW, läuft -> derzeit 0.9 MW
Windrad: Lose, Nenn-Leistung 3.5 MW, steht -> derzeit 0 MW
Windrad: Bade, Nenn-Leistung 2.5 MW, läuft -> derzeit 4.5 MW

Aufgabe5b:

TestWindrad

zum ausführlichen Testen dieser Klasse.

Aufgabe6:

Kraftwerksanlage

// Fortsetzung

Konstruktor

Instanziert die Collection.

getWindstaerke

setWindstaerke(windstaerke) vorerst ohne Plausibilitätsprüfung - siehe spätere Teil-Aufgabe!

setRunning(running) Schaltet die ganze "Anlage" ein/aus (alle vorhandenen Generatoren müssen dabei natürlich einzeln (!) ein- oder ausgeschaltet werden!).

addGenerator(generator) Ist der zu übernehmende Generator ein Windrad, ist dafür zu sorgen, dass jedes Windrad "Zugriff" auf die Zentrale "erhält" (z.B. um jederzeit die Wind-Geschwindigkeit abfragen zu können).

(**Tipp!!:** Schreiben Sie einen (sinnvollen) Teil der toString zum Testen bereits jetzt!!)

removeGenerator(pos) Entfernt den Generator an der übergebenen Position **pos**.

removeGeneratoren(nenn) Entfernt alle Generatoren, deren Nennleistung unter der übergebenen Nennleistung liegt und gibt die entfernte Gesamt-Nennleistung zurück.

alleWindraederEinAus(running) Schaltet sämtliche Windräder ein oder aus.

setWindstaerke(.....) Trägt die aktuelle Windgeschwindigkeit ein. Steigt hierbei die Windstärke über 110 km/h, müssen die Windräder sofort automatisch abgeschaltet und später (z.B. bei leichterem Wind) erst "manuell" wieder eingeschaltet werden.

berechneGesamtleistung() Ermittelt die aktuelle Gesamt-Leistung des Kraftwerks.

anzGeneratorenRunning() Ermittelt die Anzahl der derzeit eingeschalteten Generatoren.

toString() Gibt Informationen über die gesamte Kraftwerksanlage zurück.

Beispiel (und vorgeschriebene Form!):

Anlage läuft (5 Generatoren, davon 3 running)
Derzeitige Windgeschwindigkeit: 100 km/h

Generator 1: Windrad: Mödl, Nenn-Leistung 3.5 MW, steht -> derzeit 0 MW

Generator 2: Kaplan: Wien, Nenn-Leistung 35 MW, läuft -> derzeit 35 MW

Generator 3: Windrad: Lose, Nenn-Leistung 2.5 MW, läuft -> derzeit 5.0 MW

Generator 4: Kaplan: Linz, Nenn-Leistung 10 MW, steht -> derzeit 0 MW

Generator 5: Kaplan: Dürn, Nenn-Leistung 25 MW, läuft -> derzeit 25 MW

...

Aufgabe6b: **TestKraftwerk** zum ausführlichen (laufenden!!!) Testen dieser Klasse.

Aufgabe7:

sortLeistung() Sortiert die Collection mit Hilfe eines entsprechenden **Comparators** absteigend nach der aktuellen Leistung.

sortStandort() Sortiert die Collection mit Hilfe eines entsprechenden **Comparators** aufsteigend nach Standort.

Aufgabe8 (Files):

saveGeneratoren(filename) speichert die Generatoren in einer Datei beliebigen Namens (z.B. "generatoren.ser").

Aufgabe8b: TestSave

!!! ACHTUNG !!! Wenn hierbei eine Exception auftritt, deren Ursache Sie auch mit Hilfe des Debuggers(!!) nicht finden können, wenden Sie sich an Hr. Schmid!!!

Für später folgende Aufgaben (GUI) wäre es vorteilhaft, die Generatoren 2x zu speichern: einmal in eine Datei z.B. "generatoren_aus.ser" (sind ausgeschaltet), und ein zweites Mal in eine Datei z.B. "generatoren_ein.ser" (vorher alle Generatoren einschalten!!).

Aufgabe9:

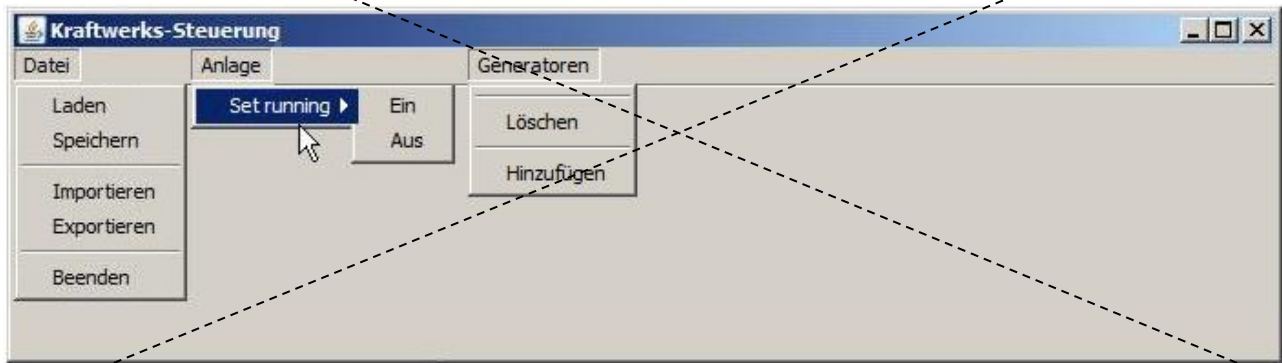
loadGeneratoren(filename) lädt die Collection aus einer Datei beliebigen Namens und fügt die Generatoren der bestehenden Collection hinzu (ACHTUNG! Hier dürfen Sie nicht über eine der LinkedList.add...-Methoden gehen!!)

Aufgabe9b: TestLoad

!!! ACHTUNG !!! Wenn hierbei eine Exception auftritt wenden Sie sich an Hr. Schmid!!!

Aufgabe10 (GUI):

Erstellen Sie für die Kraftwerks-Steuerung eine graphische Oberfläche (Klasse **Hauptfenster**, Größe: 700*200 Pixel), die folgendes Aussehen und Menü-Struktur aufweist (jedoch noch ohne jegliche Funktionalität!!) und in der Mitte des Bildschirms angeordnet ist:



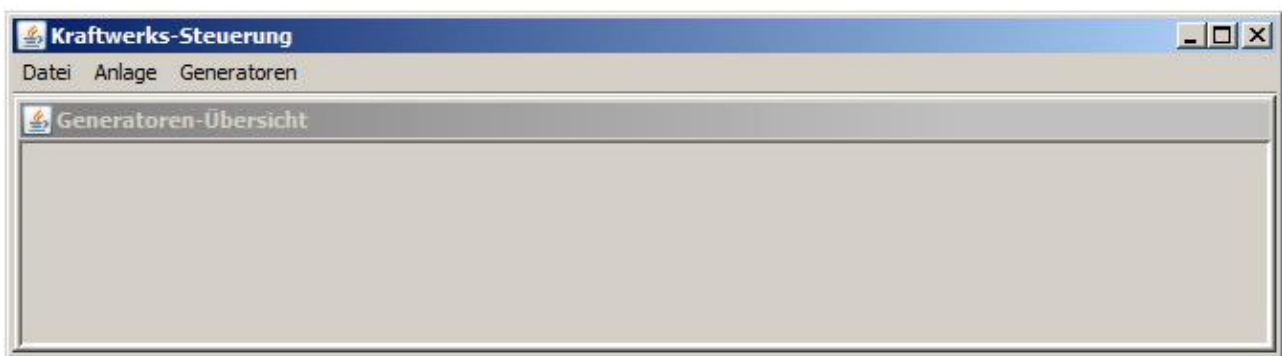
Hinweis: Die Menüpunkte "Speichern", "Exportieren" und "Löschen" sind zu Beginn nicht aktivierbar, da dies vor dem Laden von Daten keinen Sinn macht!

Aufgabe11:

Die Auswahl der Menüpunkte "**Beenden**", "**Laden**" oder "**Speichern**" soll zu entsprechenden Aktivitäten führen. Sowohl beim Laden als auch beim Speichern sollen die entsprechenden Dateien bzw. Speicherorte mit Hilfe eines Datei-Dialogfensters ausgewählt werden können, das automatisch den Ordner "c:\scratch" zur Datei-Auswahl anbietet.

Aufgabe12:

Implementieren Sie für die Darstellung der Generatoren-Daten eine Klasse **Uebersichtsfenster** (das für einen ersten Test "leer" auf dem Hauptfenster erscheinen kann!). Es soll später aber erst sichtbar werden, wenn Daten geladen wurden!!



Aufgabe13:

Stellen Sie die Generatoren-Daten in diesem internen Fenster dar (siehe **Aufgabe 12**). Verwenden Sie hierzu entweder eine **JList** oder eine **JTable**.

Implementieren Sie zu diesem Zweck bei Verwendung der **JList** einen entsprechenden **Renderer** oder (!! bei Verwendung der **JTable** die folgenden Methoden (erweitern Sie zur Darstellung d. Generatoren-Daten im Übersichts-Fenster zuerst die Klasse **Generator** - bzw. je nach Lösungsansatz auch die Sub-Klassen - um folgende Methode):

Vector getData(): liefert folgende Zeichenketten zurück:

Form:	[Art]	[Standort]	[aktuelle Leistung]
Beispiele:	"Kaplan"	"Gmunden"	"5.0"
	"Kaplan"	"Ebensee"	"15.0"
	"Windrad"	"Loser"	"1.5"

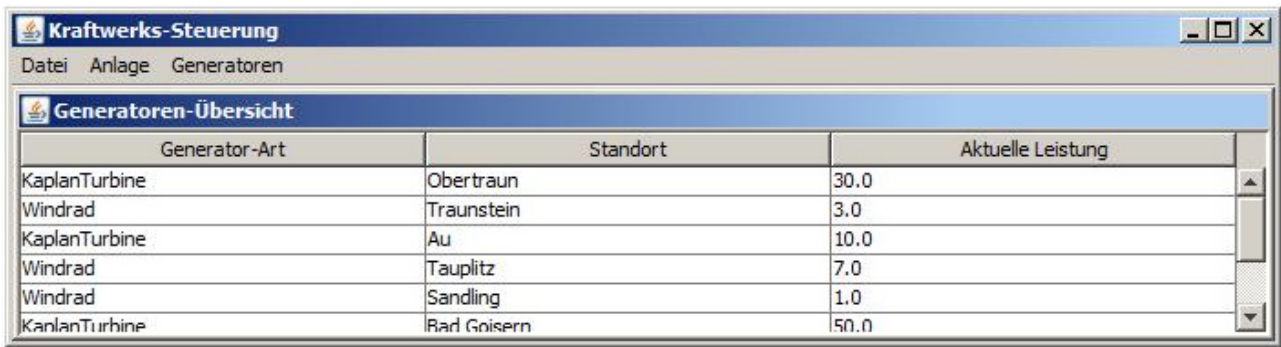
Ergänzen Sie anschließend die Klasse **Kraftwerksanlage** um die entsprechende Methode **Vector getData()** sowie um eine Methode **Vector getTitel()**; diese liefert folgende Spalten-Überschriften:

"Generator-Art" "Standort" "Aktuelle Leistung"

Achtung:

Damit nach dem Laden auch die angezeigten Windräder eine Leistung bringen (in der Kraftwerksanlage, die aus der GUI gesteuert wird, steht ja die Windstärke noch immer auf 0 !!), kann beim Laden (!! ausnahmsweise die Windstärke in der Kraftwerksanlage zu Test-Zwecken auf z.B. 50 kmh gesetzt werden!!). Dadurch können auch die Windräder in das "Sortieren nach Leistung" integriert werden.

Aussehen der Fenster nach dem Laden der Generator-Daten aus einer Datei (und Aktualisierung der Übersicht):



Generator-Art	Standort	Aktuelle Leistung
KaplanTurbine	Obertraun	30.0
Windrad	Traunstein	3.0
KaplanTurbine	Au	10.0
Windrad	Tauplitz	7.0
Windrad	Sandling	1.0
KaplanTurbine	Bad Goisern	50.0

oder



Generator-Art	Standort	Aktuelle Leistung
KaplanTurbine	Obertraun	30.0
Windrad	Traunstein	3.0
KaplanTurbine	Au	10.0
Windrad	Tauplitz	7.0
Windrad	Sandling	1.0
KaplanTurbine	Bad Goisern	50.0
KaplanTurbine	Gmunden	50.0

!!! ACHTUNG !!!

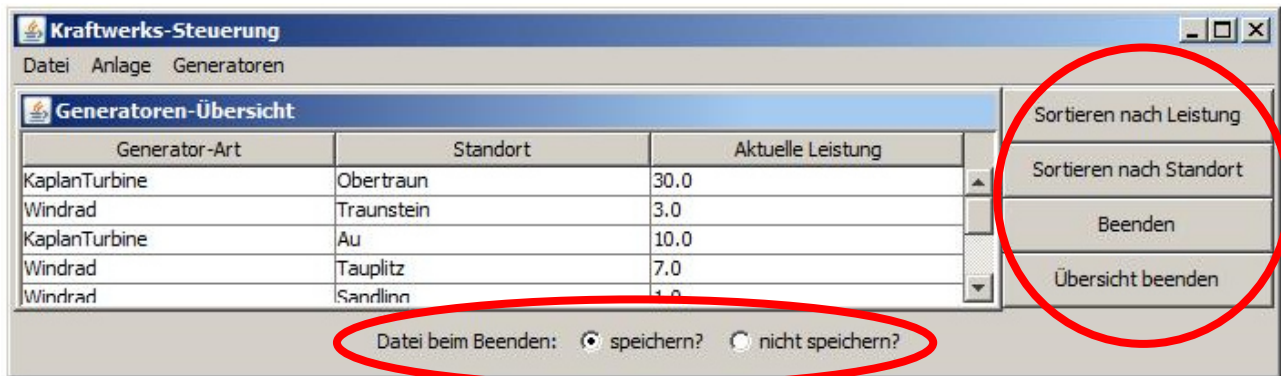
Denken Sie daran, vor dem ersten Test ev. Ihre `TestSave.main()` - Methode erneut aufzurufen, da sich die Basis-Klassen geändert haben, die serialisierten Objekte aber noch mit den alten Klassen erzeugt wurden!!

!!!!!!!!!!!!!!!!!!!!

Um auch die Windräder in den Test "Sortieren nach Leistung" einbeziehen zu können, müssen Sie (ausnahmsweise) in der Laden-Methode der GUI die Windstärke der Kraftwerkszentrale auf z.B. 50 km/h setzen.

Aufgabe14:

Erweitern Sie das Hauptfenster wie im Screenshot zu sehen ist; die Buttons sollen **noch keinerlei** Funktion haben (siehe Aufgabe 17)! Auch die Buttons sollen erst nach dem Laden von Daten sichtbar werden (zum ersten Test können auch sie gleich sichtbar sein).



Generator-Art	Standort	Aktuelle Leistung
KaplanTurbine	Obertraun	30.0
Windrad	Traunstein	3.0
KaplanTurbine	Au	10.0
Windrad	Tauplitz	7.0
Windrad	Sandling	1.0

Sortieren nach Leistung
Sortieren nach Standort
Beenden
Übersicht beenden

Datei beim Beenden: ☒ speichern? ☐ nicht speichern?

Aufgabe15:

Statten Sie nun die Buttons im unteren sowie im rechten Panel sowie die Menü - Schaltflächen "Anlage -> Set running -> Ein/Aus" mit der ihnen entsprechenden Funktionalität aus! Bei "Übersicht beenden" sollen alle Mails entfernt und die GUI in den "Originalzustand" versetzt werden.

Aufgabe16:

Bei Markierung von Generatoren und Auswahl des Menüpunktes "Generatoren -> Löschen" sollen die markierten Generatoren (auch mehrere!) aus der Collection entfernt werden.

Aufgabe17:

Der "Knopf" ☒ speichern? ist standardmäßig auf "ausgewählt" zu setzen! Außerdem ist sicherzustellen, dass jeweils immer nur einer der beiden "Knöpfe" ausgewählt werden kann!

Aufgabe18: (Bonus)

Beim "Beenden" (sowohl Menüpunkt "Beenden" als auch über den Button) sollen die derzeit geladenen Mails je nach Auswahl ("..speichern? / nicht speichern?") automatisch zur Speicherung angeboten werden!