technische universität dortmund

fakultät für informatik

Andre Droschinsky
Roman Kalkreuth
Denis Kurz
Bernd Zey

Dortmund, den 06. Dezember 2018

Praktikum zur Vorlesung Einführung in die Programmierung WS 18/19

Blatt 7

Es können 24 Punkte erreicht werden.

Grundlage: Namenskonventionen

Um den Quellcode lesbarer zu gestalten, sollten bestimmte Richtlinien zur Namensgebung und Formatierung eingehalten werden. Typischerweise variieren diese Richtlinien von Programmiersprache zu Programmiersprache und sogar innerhalb derselben Programmiersprache konkurrieren oft verschiedene Denkschulen. Eine heute weitverbreitete Namenskonvention wurde von der Programmiersprache Java geprägt. Diese beinhaltet folgende Regeln (siehe `https://en.wikipedia.org/wiki/Naming_convention_%28programming%29#Java`):

- Klassennamen sollten mit einem Großbuchstaben beginnen.
- Funktionen, Methoden und Variablen sollten mit einem Kleinbuchstaben beginnen.
- Für die oben genannten Namen gilt: klein weiterschreiben, bei zusammengesetzten Wörtern fangen Teilworte wiederum mit einem Großbuchstaben an (sogenannte *CamelCase*-Schreibweise).
- Konstanten sollten komplett in Großbuchstaben geschrieben werden.
- Verständliche (das heißt ausgeschriebene) Variablennamen benutzen, außer bei Schleifenvariablen, Zählern, oder ähnlichem.

Variablennamen lassen sich in Eclipse komfortabel projektweit ändern, indem erst der zu ändernde Name markiert und dann im Menü `Refactor` → `Rename` gewählt wird (alternativ Tastaturkombination: `Umschalt + Alt + R`).
Namenskonventionen sind Bestandteil des allgemeinen Programmierstils, der auch die übrige Formatierung des Textes (Einrückungen, etc.) umfasst (siehe `https://de.wikipedia.org/wiki/Programmierstil`). Durch die Tastenkombination `Strg + Umschalt + F` wird Eclipse veranlasst, den Quelltext selbstständig zu formatieren. Diese Funktion haben wir durch die auf Blatt 1 vorgenommenen Einstellungen weitgehend automatisiert.

Aufgabe 1: Elektrofahrzeuge (14 Punkte)

In dieser Aufgabe soll eine Klasse für Elektrofahrzeuge implementiert werden. Legen Sie dazu ein neues Projekt `Aufgabe_8` mit der Quelltextdatei `elektrofahrzeug.cpp` und der zugehörigen Header-Datei `elektrofahrzeug.h` an. Die Header-Datei soll die Klassendefinition enthalten und die Quelltextdatei die zugehörige Implementierung.

Einschränkung: Bei dieser und den folgenden Aufgaben sollen alle Methoden in einer separaten Quelltextdatei (`.cpp`) und nicht innerhalb der Header-Datei implementiert werden. Generell darf weder in der Deklaration noch in der Implementierung eine `main`-Funktion existieren.

a) Definieren Sie eine Klasse `Elektrofahrzeug` mit den nötigen *privaten* Attributen, um die Eigenschaften in der folgenden Tabelle zu modellieren.

Eigenschaft	Einheit	Wertebereich	Standardwert
Energie	kWh	reellwertig, größer 0	50
Durchschnittsverbrauch	kWh/100 km	reellwertig, größer 0	12.5
Ladestatus	%	reellwertig, 0 bis 100	50
Kilometerstand	km	reellwertig, nicht negativ	30 000
Höchstgeschwindigkeit	km/h	ganzzahlig, größer 0	160

_____ (2)

b) Implementieren Sie einen Konstruktor für die Klasse `Elektrofahrzeug`, der einen *durchschnittlichen* PKW mit den Standardwerten aus der obigen Tabelle erzeugt.

_____ (1)

c) Fügen Sie der Klasse für jedes Attribut je eine Methode zum Setzen und eine Methode zum Abfragen hinzu; lediglich der Kilometerstand soll keine Methode zum Setzen haben. Stellen Sie sicher, dass den Attributen keine ungültigen Werte zugewiesen werden: negative Werte sind nicht erlaubt und nur Batterieladestatus und Kilometerstand können 0 sein. Falls eine Methode zum Setzen mit einem ungültigen Wert aufgerufen wird, soll eine Fehlerausgabe darauf aufmerksam machen und das entsprechende Attribut soll auf denselben Wert wie beim durchschnittlichen PKW gesetzt werden.

_____ (2)

d) Implementieren Sie einen weiteren Konstruktor für die Klasse `Elektrofahrzeug`, durch den die privaten Attribute direkt gesetzt werden können. Wird ein ungültiger Wert für ein Attribut übergeben, soll analog zu den Methoden zum Setzen verfahren werden.

_____ (1)

e) Fügen Sie der Klasse eine Methode `aufladen` hinzu, welche ein Fahrzeug voll auflädt und die Kosten zurückgibt. Der Preis in Cent/kWh soll als Parameter an die Funktion übergeben werden.

_____ (1)

f) Fügen Sie der Klasse eine Methode `maximaleReichweite` hinzu, die berechnet, wie weit mit dem derzeitigen Ladestand noch maximal gefahren werden kann. Gehen Sie dabei davon aus, dass sich der durchschnittliche Energieverbrauch auf der Fahrt durch eine sparsame Fahrweise um 10% senken lässt.

_____ (1)

g) Implementieren Sie eine private, statische Methode `verbrauchFuerStrecke`. Diese soll eine Strecke in km und einen durchschnittlichen Verbrauch in kWh/100 km als Parameter erhalten und zurückgeben, wieviel kWh für diese Parameter verbraucht wird.

_____ (1)

h) Fügen Sie der Klasse eine Funktion `fahren` hinzu, welche ein Fahrzeug eine gewisse Strecke fahren lassen soll. Der Verbrauch soll hierbei dem Durchschnittsverbrauch des Fahrzeugs entsprechen. Die gewünschte Distanz soll als Parameter an die Funktion übergeben werden. Prüfen Sie zu Beginn, ob das Fahrzeug die Distanz mit dem aktuellen Ladestand schaffen kann. Wenn dies nicht der Fall ist, soll das Auto nicht fahren und `false` zurückgegeben werden; ansonsten soll `true` zurückgegeben und die entsprechenden Attribute angepasst werden.

EINSCHRÄNKUNG: Verwenden Sie die Methode `verbrauchFuerStrecke` aus Teilaufgabe **g)**.

_____ (1)

i) Schreiben Sie eine Methode `zielSchnellErreichbar`, die berechnet, ob sich ein Ziel mit einer gewissen Entfernung bei voller Geschwindigkeit in einer vorgegebenen Zeit erreichen lässt. Gehen Sie dabei davon aus, dass der Energieverbrauch beim Fahren mit Höchstgeschwindigkeit 30 % über dem Durchschnittsverbrauch liegt. Die Entfernung sowie die Zeitbeschränkung sollen der Funktion als Werte übergeben werden. Überlegen Sie sich sinnvolle Parameter und einen passenden Rückgabewert.

EINSCHRÄNKUNG: Verwenden Sie die Methode `verbrauchFuerStrecke` aus Teilaufgabe **g)**.

_____ (1)

j) Erstellen Sie in einem Testprogramm drei `Elektrofahrzeug`-Objekte in einem Array, die wie folgt initialisiert werden:

Eigenschaft	durchschnittlicher PKW	Sportwagen	Kleinstwagen
Energie	Standardwert	90	40
Durchschnittsverbrauch	Standardwert	20	9
Ladestatus	Standardwert	30	90
Kilometerstand	Standardwert	15 000	40 000
Höchstgeschwindigkeit	Standardwert	300	110

Verwenden Sie dabei zur Initialisierung jeden implementierten Konstruktor mindestens ein Mal. Die Objekte sollen nicht in das Array kopiert, sondern direkt im Array erzeugt werden. Testen Sie nun Ihre Implementierung und geben Sie zu Beginn und jeweils nach den Test-Operationen 1 bis 6 die Daten aller Fahrzeuge aus. Verwenden Sie für den gesamten Test die zuvor angelegten `Elektrofahrzeug`-Instanzen. Nutzen Sie dabei aus, dass die Fahrzeuge in einem Array stehen, um nicht unnötig Programmcode zu duplizieren.

1. Geben Sie für die Fahrzeuge die maximale Reichweite aus.
2. Testen Sie für die Fahrzeuge, ob Ziele in 150 km und 300 km erreichbar sind.
3. Lassen Sie alle Fahrzeuge 200 km fahren.
4. Laden Sie die Fahrzeuge und geben Sie die Kosten für einen Strompreis von 120 Cent/kWh aus.
5. Testen Sie, ob ein Ziel in 150 km Entfernung beim Fahren mit Höchstgeschwindigkeit im durchschnittlichen PKW in zwei Stunden erreichbar ist, wenn in der Batterie noch 20 kWh übrig sind.
6. Laden Sie das Fahrzeug voll und testen Sie erneut.

_____ (3)

Aufgabe 2: Ladegutscheine (5 Punkte)

In dieser Aufgabe soll eine Klasse für Ladegutscheine implementiert werden. Fügen Sie diese dem Projekt `Aufgabe_8` aus der ersten Aufgabe hinzu. Ein Ladegutschein hat eine ID und ein Guthaben, welches nicht negativ werden darf.

a) Erstellen Sie eine Header-Datei `ladegutschein.h`, in der Sie die Klasse `Ladegutschein` deklarieren. Diese soll über einen Konstruktor verfügen, der die notwendigen Daten initialisiert. Überlegen Sie sich sinnvolle Datentypen für die Attribute, welche als `private` deklariert und über `get`-Funktionen abrufbar sein sollen. Benutzen Sie im Konstruktor eine statische Variable, um die IDs automatisch zu generieren. Deklarieren Sie eine Methode `reduziereGuthaben`, die das Guthaben um einen als Parameter übergebenen Betrag, höchstens aber das komplette Guthaben (also auf 0) verringern soll.

_____ (2)

b) Implementieren Sie die in der ersten Teilaufgabe deklarierten Methoden in der Datei `ladegutschein.cpp`. Stellen Sie sicher, dass ein Ladegutschein nicht mit ungültigen Guthabenwerten initialisiert wird.

_____ (2)

c) Erweitern Sie das Testprogramm aus Aufgabe 1, indem Sie je einen Gutschein mit 10 €, 20 € und 30 € erstellen und anschließend deren Daten ausgeben. Die Objekte sollen analog zu den Fahrzeugen aus Aufgabe 1 in einem Array abgelegt werden.

_____ (1)

Aufgabe 3: Ladestation (5 Punkte)

In dieser Aufgabe soll eine Klasse für Ladestationen implementiert werden, welche die beiden zuvor erzeugten Klassen `Ladegutschein` und `Elektrofahrzeug` verwendet. Arbeiten Sie dazu weiterhin mit dem Projekt `Aufgabe_8`.

a) Erstellen Sie eine Header-Datei `ladestation.h` und die zugehörige Quelltextdatei `ladestation.cpp`. Hierbei sollen, wie bei den anderen Aufgaben, die Attribute `private` sein und über `get`-Funktionen abrufbar sein. Als Attribut der Klasse `Ladestation` dient der aktuelle Strompreis pro Kilowattstunde. Dieser soll durch den Konstruktor gesetzt werden können.

_____ (2)

b) Fügen Sie eine Methode `laden` hinzu, welche je ein Objekt vom Typ `Elektrofahrzeug` und `Ladegutschein` erhält. Das Elektrofahrzeug soll nach Aufruf der Methode aufgeladen sein und das Guthaben des Ladegutscheins um den entsprechenden Betrag verringert werden. Reicht das Guthaben nicht aus, so wird das Elektrofahrzeug aufgeladen, bis das Guthaben des Ladegutscheins aufgebraucht ist.

_____ (2)

c) Erweitern Sie das Testprogramm aus den vorherigen Aufgaben, indem Sie eine Instanz der Klasse `Ladestation` mit einem beliebigen Namen und einem Energiepreis von 120 Cent/kWh erzeugen. Setzen Sie dann den Ladestatus jedes Fahrzeugs auf 50 % und lassen Sie jedes mit dem zugehörigen Ladegutschein an der Ladestation aufladen: das erste Fahrzeug wird mit dem ersten Gutschein aufgeladen, und so weiter. Geben Sie vor und nach den Ladevorgängen die Werte aller Objekte aus.

_____ (1)