Informatik Java Praktikum

Nina Lee Davies, Matrikelnummer 4649777, Mechatronik

Max Martin Paulenz, Matrikelnummer 4927304, Elektrotechnik

Das Programm simuliert einen Roboter. Dieser soll mehrere vom Benutzer einzugebende Points of interest (POI) in sinnvoller Reihenfolge abfahren oder beim Überqueren des Spielfeldes zufällig generierten Rechtecken ausweichen. Der Prozess wird dem User grafisch angezeigt. Das Programm wurde in der Sprache Java im IDE InelliJ IDEA umgesetzt. Wir haben uns sowohl bei der Variablenbenennung als auch den Kommentaren nach Absprache in der Einführungsübung für die Englische Sprache entschieden.

Den Quelltext inklusive detaillierter Beschreibungen der einzelnen Methoden und dem Klassendiagramm finden Sie in der PDF-Datei Quelltext. Sie wurde automatisch mit BlueJ erstellt.

5.

a) Die POI Liste wurde als lokale Variable innerhalb einer Methode konzipiert. Es wäre somit prinzipiell mit wenig Aufwand möglich, einen zweiten Roboter mit anderen POI in das gleiche Spielfeld zu ‚setzen'. Für die Hindernisliste (Obstacle List) ist es dagegen sinnvoller, diese als Attribut der Klasse Spielfeld (Court) zu nutzen, da man pro Spielfeld nur ein Set an Hindernissen bereithält, welches der Roboter umfahren muss.

b) Das Entwurfsmuster Singleton verwendet man bei Klassen von denen nur ein Objekt gleichzeitig existieren soll. In unserem Programm ist dies die Klasse Canvas. Es soll genau ein Fenster offen sein, in welchem genau ein Set an Hindernissen dargestellt wird. Die Umsetzung erfolgte mittels Definition der Klasse Canvas als static.

c) Das Auslagern von Teilen größerer Methoden in kleiner Hilfsmethoden ist immer dann sinnvoll, wenn Teile dieser Methode auch von anderen Methoden im Programm verwendet werden können. Ein gutes Beispiel aus unserem Programm sind die intInput und limitInput Methoden der Klasse court. Viele übergeordnete Methoden wie die Erzeugung der Rechtecke oder die Eingabe der POI erfordern positive Integer, teils mit einem bestimmten Maximalwert. Indem man nun immer wieder auf diese Hilfsmethoden zugreift erspart man sich sowohl Arbeit, als auch Speicherplatz. Zudem erhöht dieses Refactoring auch die Lesbarkeit des Programms für dritte. Anhand der Funktion der Hilfsmethoden lässt sich schnell die Funktion einer übergeordneten Methode erfassen.

d) Javadoc Kommentare werden in bestimmten IDE beim bloßen hovern über dem Aufruf einer Methode angezeigt. So können andere Programmierer eine fremde Methode und ihre Funktion verstehen, ohne deren Quellcode genauer zu inspizieren. Zudem lässt sich später aus den Javadoc Kommentaren ohne viel Aufwand eine Dokumentation erstellen.

e) Probleme traten vor allem im Konflikt zwischen unseren eigenen Vorstellungen und bestimmten detaillierteren Vorgaben seitens der Praktikumsaufgaben auf. Dies führte teilweise zu verwaisten Methoden. Diese haben wir letztendlich im Quellcode belassen um unseren Ideenfindungsprozess etwas ersichtlicher zu machen. Wir hatten große Probleme damit, eine Elegante Methode zum Sortieren der POI zu finden. Der Versuch, dies über die Arraysort Funktion zu lösen entpuppte sich als Mammut-Aufgabe, welche dann bedingt durch Zeitmangel leider aufgegeben werden musste. Auch die Idee, das Programm in Englisch zu verfassen verkomplizierte die Angelegenheit stark und führte zum Teil zu andernfalls vermeidbaren Ungereimtheiten zwischen unseren Vorstellungen und den Praktikumsaufgaben. Wir kamen zu dem Schluss, dass es besser gewesen wäre, hier den Konventionen zu folgen.