Kapitel 7 Schnittstelle als Außenansicht eines Objekts

Lernziele:

Schnittstelle, Wiederholung Umsetzung von Beziehungen, Objektkommunikation an einem neuen Beispiel

7.0 Vorbereitungen

a) Backend

Für die folgenden Kapitel unterstützt dich eine Programmbibliothek, im Folgenden backend genannt. Sie enthält Klassen, die dich in der Spielentwicklung mit vielen Funktionen unterstützen. Du musst die Bibliothek in BlueJ wie folgt integrieren:

i. Lade die Datei backendVersionX_Y.jar herunter und speichere sie in deinem Verzeichnis.
ii. Öffne in der BlueJ-Menüleiste das Menü *Werkzeuge > Einstellung*
iii. Klicke auf den Reiter „Bibliotheken“ und dann auf den Button „Hinzufügen“
iv. Wähle nun die Datei backendVersionX_Y.jar aus und schließe das Menü mit „OK“. (Abbildung A)
v. Starte zum Abschluss BlueJ neu, damit die Klassensammlung von BlueJ geladen wird.

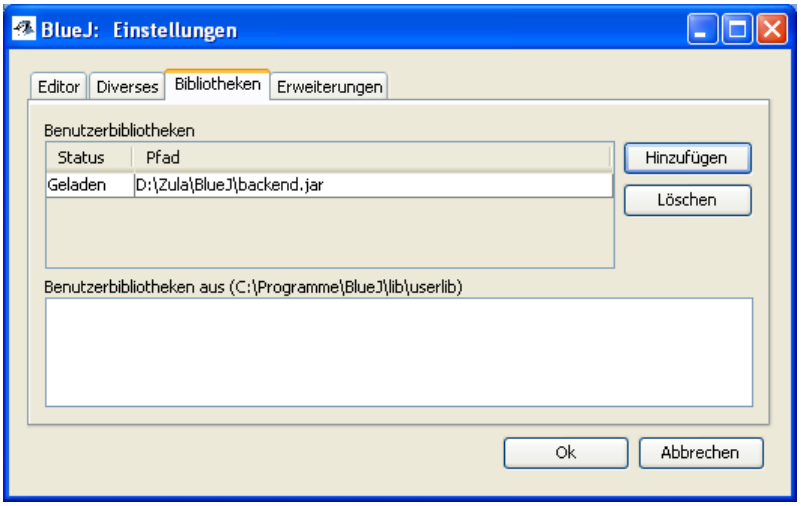


Abbildung A: Einbinden einer Bibliothek in BlueJ

b) Eine Klasse aus einer Datei in ein BlueJ-Projekt laden

Zur graphischen Anzeige wirst du in Zukunft von deinem Lehrer immer wieder neue Klassen erhalten, die du in dein BlueJ-Projekt integrieren musst. Aus den ersten Kapiteln kennst du schon die Klasse KREISFORM, die eine graphische Anzeige von MAMPFI ermöglicht hat.
In diesem Kapitel benötigst du die Klasse PUNKTELISTEANZEIGE, um die Verlaufsliste aus dem Kapitel 6 anzuzeigen. Gehe wie folgt vor:

i. Lade die Datei PUNKTELISTEANZEIGE.zip herunter, entpacke die darin enthaltene Datei PUNKTELISTEANZEIGE.java und speichere sie in deinem Informatik Ordner.
ii. Öffne dein BlueJ-Projekt mit dem Endergebnis von Kapitel 6 und speichere es unter einem anderen Namen, z.B. kap07 ab (Erinnere dich, es ist wichtig auf Versionen zurückgreifen zu können, die funktionieren und bereits getestet wurden.)

iii. Öffne in der BlueJ-Menüleiste das Menü *Bearbeiten* > *Klasse aus Datei hinzufügen* (Abbildung B)

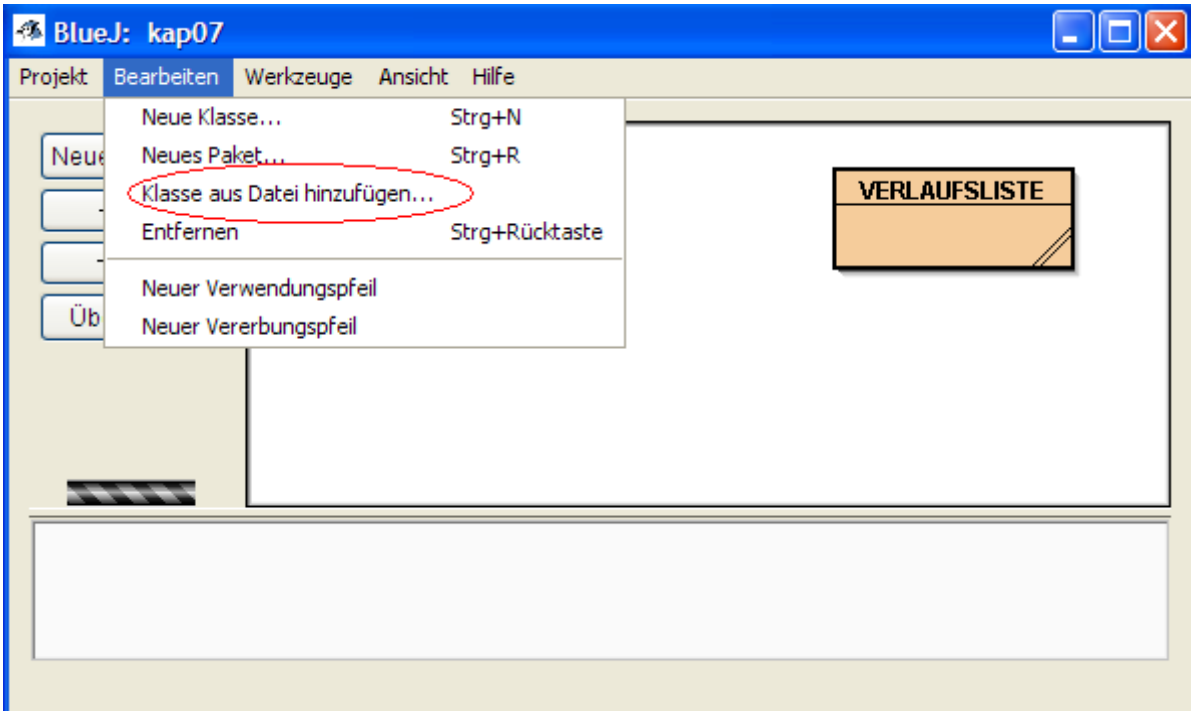


Abbildung B: Einbinden einer Java Datei in BlueJ ein BlueJ Projekt

iv. Wähle in deiner Verzeichnisstruktur die unter i) gespeicherte Klasse PUNKTELISTEANZEIGE.java aus und bestätige die Auswahl dann durch einen Klick auf den Button „Hinzufügen".

7.1 Methoden einer Klasse als Schnittstelle

In Kapitel 6 hast du eine Verlaufsliste erstellt. Jedoch war das Anzeigen der Liste mit Hilfe von zwei Objektinspektoren sehr umständlich und für ein Computerspiel nicht geeignet. Passender wäre eine Darstellung wie in Abbildung 1.

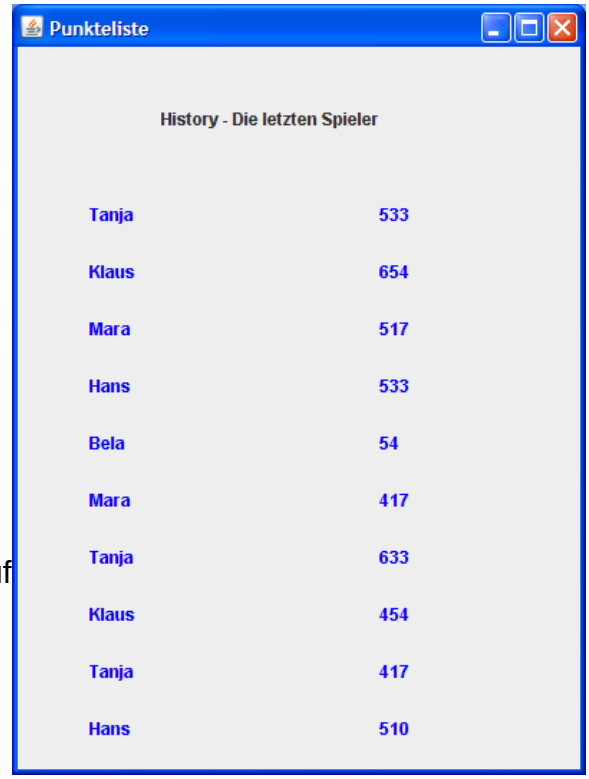


Abbildung 1: Verlaufsliste – Namen und Punkte dargestellt als Tabelle in einem Fenster

Aufgabe 7.1
Wie könnte ein Objekt einer neuen Klasse PUNKTELISTEANZEIGE helfen, eine Verlaufsliste wie in Abbildung 1 darzustellen? Welche Parallele gibt es zur Klasse MAMPFI? (Dort benötigte die Klasse MAMPFI die Hilfe einer anderen Klasse.)

Mampfi muss auch in Form eines Symbols auf dem Bildschirm dargestellt werden. Möglich war das mit der Klasse KREISFORM.
Bei der Programmierung der Klasse MAMPFI konntest du die Klasse KREISFORM einfach verwenden, ohne dass du dich mit dem Quelltext auseinander setzen musstest. Für dich war nur wichtig, welche Methoden bzw. Attribute die Klasse hat. Diese Informationen hast du über ein Klassendiagramm im Anleitungstext erhalten. Du musstest nicht wissen, wie die Methoden intern funktionieren, sondern du hast sie verwendet. Du hast das Objekt wie einen Dienstleister über einen Methodenaufruf Arbeit erledigen lassen. Beispielsweise konntest du durch einen Methodenaufruf das Objekt der Klasse KREISFORM dazu bringen, dass es seine Füllfarbe ändert. Dazu war der aussagekräftige Name der Methode *FuellFarbeSetzen* ausreichend, um zu wissen, was die Methode tut. Neben dem Namen einer Methode ist auch noch die Information wichtig, ob und gegebenenfalls welche Eingabewerte sie benötigt. Im Fall der Methode *FuellFarbeSetzen* ist als Eingabewert eine Farbangabe in Form einer Zeichenkette (String) nötig.

Diese Vorgehensweise ist auch bei großen Softwareprojekten üblich. Es arbeiten mehrere Informatiker zusammen. Jeder schreibt Klassen. Benötigt jemand eine Klasse eines Kollegen, so befasst er sich nicht mit deren Quelltext, sondern vereinbart vorher, welche Methoden, d.h. welche Fähigkeiten die Objekte der Klasse haben sollen. So kann sich einer auf die Implementierung[1] konzentrieren und die anderen nutzen die Klassen einfach – genauer gesagt die Methoden der Objekte dieser Klassen. Die Nutzer sparen sich damit viel Zeit und Arbeit, da sie sich nicht mit den Details der Implementierung beschäftigen müssen.

Die Menge der Methoden, die ein Objekt „nach außen“ zur Verfügung stellt, wird als seine **Schnittstelle** bezeichnet.

1 Realisierung eines Entwurfs über eine konkrete Programmiersprache wie Java als lauffähiges Programm.

Auch du arbeitest täglich mit Schnittstellen:
Du bedienst den Fernseher per Fernbedienung, telefonierst mit dem Handy per Tastatur ohne jeweils zu wissen, was an technischen Details (Stromfluss durch Leiterbahnen mit Kondensatoren...) in den Geräten abläuft. Für dich ist ausreichend zu wissen, mit welcher Tastenkombination du eine gewünschte Funktionalität[2] , z.B. das Verschicken einer SMS an die beste Freundin, erhältst. Die Funktionalitäten eines Geräts und die Tastenkombination, mit denen man sie aufruft, sind die Schnittstelle.

Hinweis:
Durch einen Klick mit der rechten Maustaste auf eine Klasse in BlueJ lässt sich über den Kontextmenüpunkt „Display UML“ (Abbildung 2) das erweiterte Klassendiagramm anzeigen. Es werden alle Methodenköpfe angezeigt. Somit erhält man mit dem Methodenbezeichner, der Anzahl der Eingabewerte und ihren Datentypen wichtige Informationen zur Schnittstelle der Objekte einer Klasse (Abbildung 3a).

Abbildung 2: Kontextmenü der Klassen in BlueJ

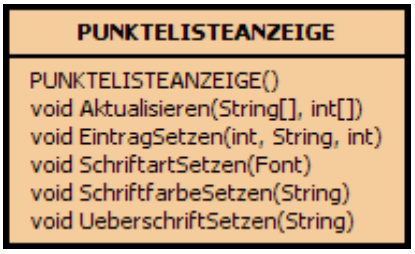


Abbildung 3a: Anzeige der Schnittstelle über die BlueJ Extension UML

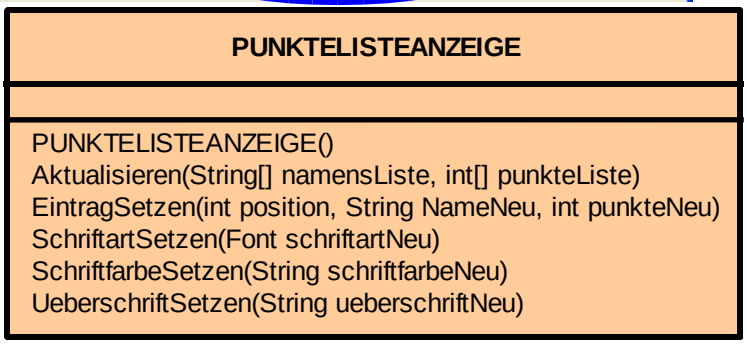


Abbildung 3b: Erweitertes Klassendiagramm der Klasse PUNKTELISTENANZEIGE als Schnittstellenbeschreibung

Leider werden bei der UML-Anzeige in BlueJ (Abbildung 3a) nicht die Namen der Eingabeparameter angezeigt. Diese sind oft eine Hilfe, um die Bedeutung der Eingabeparameter zu verstehen. Um die Schnittstelle der Klasse PUNKTELISTEANZEIGE besser zu verstehen, sind die Parameternamen in Abbildung 3b ergänzt.

2 Fähigkeit eines Produkts eine bestimmte Aufgabe zu lösen.

Aufgabe 7.2
Versuche knapp zu beschreiben, was ein Aufruf der Methoden *Aktualisieren* bzw. *EintragSetzen* bewirkt.

Der Methode *Aktualisieren* muss man beim Aufruf zwei Felder übergeben. Eines mit den (zehn) Namen, das andere mit den zugehörigen Punkten. Die Methode sorgt dann für ein Aktualisieren aller zehn Einträge im Objekt der Klasse PUNKTELISTEANZEIGE. Eine einzelne Zeile dagegen kann man mit der Methode *EintragSetzen* verändern. So bewirkt der Methodenaufruf *EintragSetzen(4, "Bela", 345),* dass in der 5. Zeile der Name Bela und die Punkte 345 eingetragen werden. Fünfte Zeile deshalb, weil wie beim Index die erste Zeile die Positionsnummer 0 hat.

7.2 Verlaufsliste in einem Fenster anzeigen

Wie in Kapitel 3 bis 5 ein Objekt der Klasse KREISFORM zur Darstellung von Mampfi geholfen hat, soll nun die Verlaufsliste durch ein Objekt der Klasse PUNKTELISTEANZEIGE dargestellt werden (Abbildung 4).

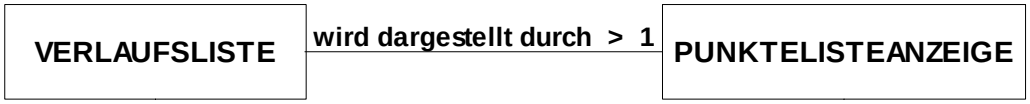


Abbildung 4: Beziehung zwischen Objekten der Klassen VERLAUFSLISTE und PUNKTELISTEANZEIGE

Aufgabe 7.3
Wie wird eine Objektbeziehung in einem Java Programm umgesetzt?

Eine Umsetzung der Beziehungen erfolgt mit Referenzattributen.

Aufgabe 7.4
Bei welcher der beiden Klassen aus Abbildung 4 muss ein Referenzattribut ergänzt werden? Begründe deine Antwort und zeichne das entsprechende erweiterte Klassendiagramm.

Da das Objekt der Klasse VERLAUFSLISTE seinem Darstellungsobjekt Änderungen mitteilen muss, damit dieses entsprechend der Änderungen eine korrekte Darstellung liefert, muss es das Darstellungsobjekt ansprechen können. Dazu ist eine Referenz nötig. Es muss also in der Klasse VERLAUFSLISTE ein Referenzattribut auf ein Objekt der Klasse PUNKTELISTEANZEIGE ergänzt werden (Abbildung 5).

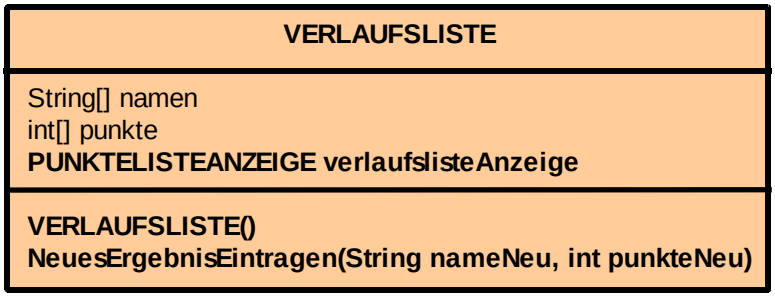


Abbildung 5: Erweitertes Klassendiagramm der Klasse VERLAUFSLISTE ergänzt um das Referenzattribut verlaufslisteAnzeige

Aufgabe 7.5

a) Überlege, warum in der Klasse VERLAUFSLISTE auch der Konstruktor und die Methode *NeuesErgebnisEintragen* verändert werden müssen.
(Solltest du bei dieser Teilaufgabe Schwierigkeiten haben, so findest du Tipps im Anhang)

b) Setze die Änderungen der Klasse VERLAUFSLISTE in deinem BlueJ Projekt um.

c) Erzeuge ein Objekt der Klasse VERLAUFSLISTE.

d) (Vermutlich im Hintergrund) erscheint ein Anzeigefenster. Wähle dort im Menü Labyrinth den Unterpunkt Punkteliste. Wenn du richtig programmiert hast erscheint eine Punkteliste wie in Abbildung 1.

e) Rufe nun mehrfach die Methode *NeuesErgebnisEintragen* auf und überprüfe, ob sich die Verlaufsliste füllt.

Abbildung 6: Anzeigefenster

Um deutlich zu machen, dass die Vorgehensweise bei der Darstellung der Verlaufsliste identisch zur Darstellung von Mampfi ist, sind folgenden Schritte der beiden Klassen in Abbildung 7 gegenübergestellt:

- Deklaration eines Referenzattributs, welches die Beziehung zum Darstellungsobjekt umsetzt
- Erzeugen des Darstellungsobjekts und Zuweisung zum Referenzattribut
- Setzen von Attributwerten des referenzierten Objekts durch Methodenaufrufe, um einen passenden Startzustand zu haben.
- Werden in Methoden des dargestellten Objekts Attributwerte verändert, müssen in der Regel auch Attributwerte des darstellenden Objekts angepasst werden.

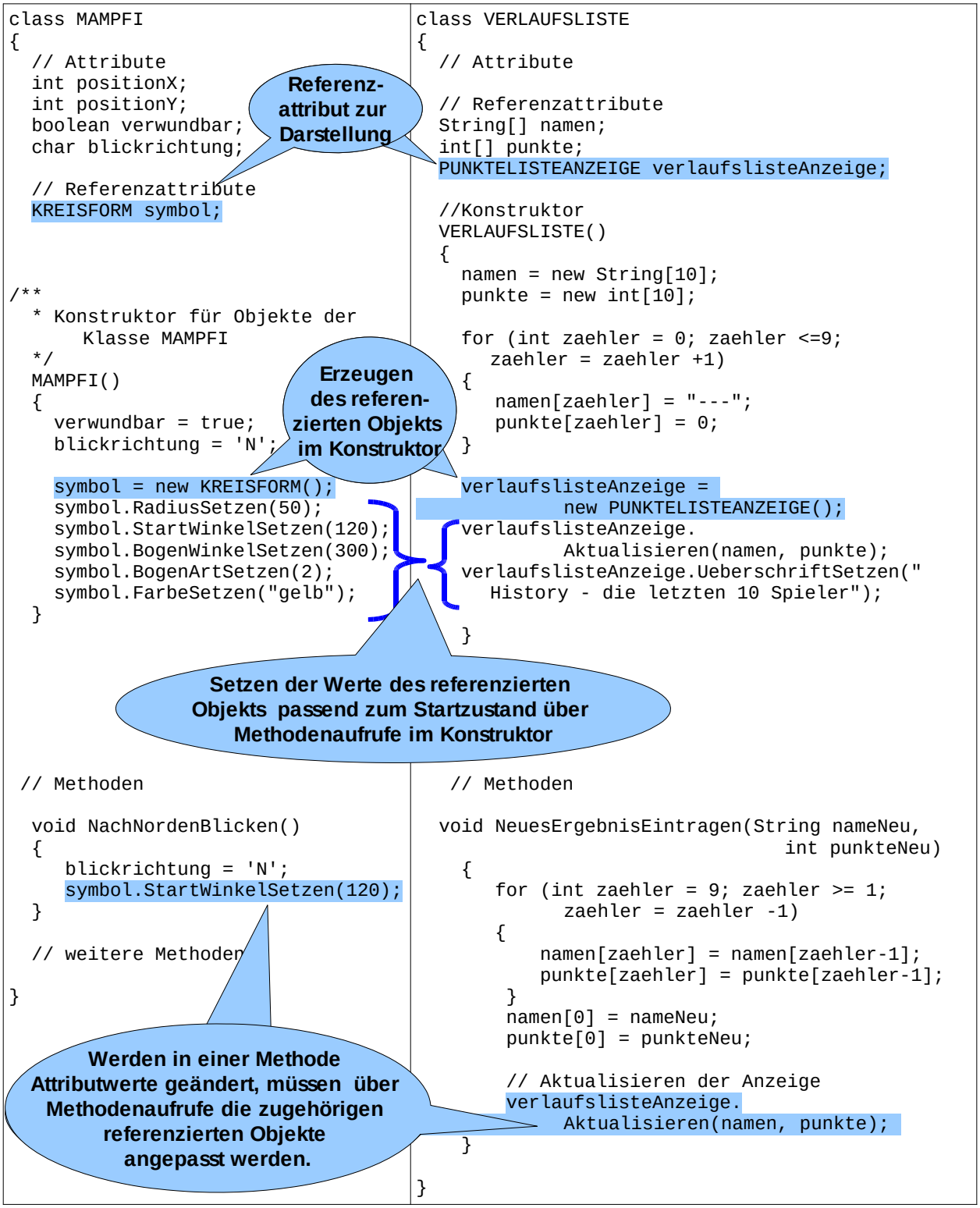


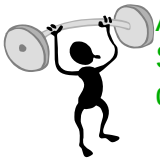
Abbildung 7: Vergleich der Quelltexte der Klassen MAMPFI und VERLAUFSLISTE

Aufgabe 7.6
Skizziere ein Sequenzdiagramm zur Veranschaulichung der Objektkommunikation bei einem Aufruf der Methode *NeuesErgebnisEintragen.*
Hinweise:
- Mach dir bewusst, dass du mit dem Sequenzdiagramm das veranschaulichst, was du bei der Bearbeitung der Aufgabe 7.4 e) auslöst.
- Mach dir bewusst, dass du zu dem Sequenzdiagramm auch ein Rollenspiel durchführen kannst, auch wenn es nicht so anschaulich ist wie bei der Kommunikation zwischen mampfi und seinem symbol.

7.3 Zusammenfassung

Aufgabe 7.7
a) Fasse die wesentlichen Inhalte dieses Kapitels in deinem Heft zusammen. Wichtige neue Fachbegriffe sind Schnittstelle, Implementierung und Funktionalität. (Die letzten beiden wurden zwar nicht fett markiert, werden aber dennoch in Zukunft auch verwendet.)
b) Im Lehrtext oder/und bei der praktischen Umsetzung kommen jedoch auch folgende Begriffe aus den früheren Kapiteln vor: Referenzattribut, Konstruktor, Zuweisung, new-Operator, Sequenzdiagramm, Objektkommunikation, Umsetzung von Beziehungen zwischen Objekten in einem Java Programm (Wiederum am Beispiel, dass Objekte durch andere Objekte dargestellt werden).
Solltest du bei den Begriffen über deren Bedeutung unsicher sein, so lies in deiner bisherigen Zusammenfassung nach. Solltest du immer noch unsicher sein, frage beim Lehrer nach und ergänze dann deine Zusammenfassung.

c) In Kapitel 4 und 5 hast du gelernt, wie man sich ausgehend von einer Zielvorstellung (Objekt mampfi im Kontext eines Spiels) über Modellierung, Programmierung und Testen schrittweise einer Lösung annähert. Diese Vorgehensweise wurde in Kapitel 6 und 7 wiederholt.
Liste wichtige Schritte bei der Umsetzung von einem Modell zum Programm auf (falls dies noch nicht Teil deiner Zusammenfassung von Kapitel 4 bzw. 5 ist). Vergleiche DANACH dein Ergebnis mit dem Lösungsvorschlag im Anhang.



Aufgabe 7.8
Sorge auch für eine graphische Darstellung deiner Bestenliste aus Kapitel 6. Beachte, dass pro BlueJ-Projekt nur eine Punkteliste angezeigt werden kann.

Anhang: Tipps /Lösungen

Tipp zu Aufgabe 7.5
Im Konstruktor muss das Darstellungsobjekts erzeugt und dem Referenzattribut zugewiesen werden. Weiterhin müssen die Attributwerte des referenzierten Objekts durch Methodenaufrufe richtig gesetzt werden, um einen passenden Startzustand zu erhalten.

Lösung zu Aufgabe 7.7 c)
Hinweise zum Vorgehen:
a) Erstelle zunächst ein Klassendiagramm, das als ersten Schritt nur Attribute und den Konstruktor enthält.
b) Notiere als ersten Schritt zur Umsetzung das erweiterte Klassendiagramm. Im Vergleich zum Klassendiagramm aus a) sind dort die Datentypen der Attribute ergänzt.
c) Setze den bisherigen Teil in Java um.
d) Teste das Programm, indem du ein Objekte erzeugst und dessen Attributwerte mit Hilfe des Objektinspektors überprüfst.
e) Plane die Umsetzung von Objektbeziehungen mit Hilfe von Referenzattributen. Ergänze dazu das erweiterte Klassendiagramm.
f) Ergänze die Klasse in Java um die Referenzattribute und dazu passende Anweisungen im Konstruktor.
d) Teste durch das Erzeugen eines Objekts, ob das Objekt sich richtig verhält, (ob die Objekte korrekt im Zeichenfenster dargestellt werden). Bei Bedarf hilft wieder der Objektinspektor.
Je nach Situation kann es auch sinnvoll sein mehrere Objekte zu erzeugen.