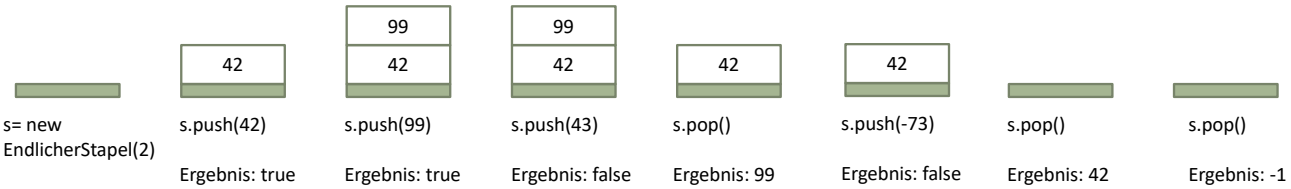
Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Programmierung 1

Wintersemester 2023/24
Aufgabenblatt 9

Beachten Sie, dass eine Probeklausur online ist, aus ihr kann auf den Aufbau der Klausur geschlossen werden, wobei sich die endgültige Klausur natürlich auf den *gesamten Veranstaltungsinhalt* bezieht.

29. Aufgabe (4 Punkte, Testen; Testerstellung klausurrelevant)



Ein Stapel (Stack) ist eine Datenstruktur, die Objekte eines bestimmten Typs aufnehmen kann. Dabei werden Objekte mit einer Methode push(.) so auf den Stapel gelegt, dass das neuste Element immer oben liegt. Mit der Methode pop(.) wird das oberste Element zurückgegeben und vom Stapel gelöscht. Weiterhin ist es sinnvoll, wenn es Methoden istLeer() bzw. istVoll() gibt, mit denen man prüfen kann, ob ein Stapel voll oder leer ist.

a) Realisieren Sie eine Klasse EndlicherStapel für *positive* int-Werte, mit einem Konstruktor, der die maximale Größe des Stapels als Parameter hat. *Nutzen Sie zur Realisierung des Stapels eine ArrayList<Integer>*.
b) Realisieren Sie die Methoden istVoll() und istLeer() zur Überprüfung des Stapelzustands, die jeweils ein Boolesches Ergebnis liefern.
c) Realisieren Sie eine Methode push(int), mit der ein int-Wert auf den Stapel gelegt wird und die einen Booleschen Wert als Ergebnis liefert. Wird versucht einen nicht positiven Wert auf den Stapel oder versucht einen Wert auf einen vollen Stapel zu legen, soll das Ergebnis false, bei einem erfolgreichen push true sein.
d) Realisieren Sie die Methode pop(), die den zuletzt auf den Stapel gelegten int-Wert zurück gibt und vom Stapel löscht. Wird versucht, einen Wert von einem leeren Stapel zu nehmen, ist der Rückgabewert -1.
e) Schreiben Sie zu allen realisierten Methoden JUnit-Tests, die alle möglichen Ergebnisse testen. Nutzen Sie dazu eine Test Fixture (Test-Objekte) mit einem leeren, einem vollen und einem „normal“ gefüllten Stack, auf denen Sie jeweils alle Ihre Methoden in einzelnen Tests ausprobieren.
   Hinweis: Wenn Sie später einen Stapel in Java benötigen, sollte die Klasse Java.util.Stack genutzt werden.

30. Aufgabe (4 Punkte, geschachtelte Schleifen, Fortsetzung letztes Aufgabenblatt)

Nutzen Sie die Klasse Messreihe, die als Objektvariablen den Messort und eine Sammlung von ganzzahligen Messwerten enthält, vom vorherigen Aufgabenblatt. Es gelten die gleichen Randbedingungen. Auf der Veranstaltungsseite finden Sie ein Projekt mit Testklassen, stellen Sie am Ende sicher, dass alle Tests laufen.
Hinweise: Natürlich dürfen Sie weitere Methoden zum Strukturieren Ihres Programms oder bei Code-Wiederholungen schreiben.
Wieder gilt, als Klausurvorbereitung sollten Sie den Code erst auf Papier schreiben.

a) Schreiben Sie eine Methode alleEinmal(), die ein neues Messreihen-Objekt zurückgibt, das alle unterschiedlichen Messwerte aus der ursprünglichen Messreihe genau einmal enthält, aus {4,3,2,4,3} soll z. B. {4,3,2} werden (wobei es keine Anforderungen an die Reihenfolge gibt, es also 6 korrekte Ergebnisse gibt, von denen Sie eines berechnen sollen).
b) Schreiben Sie eine Methode gemeinsameEinfach(Messreihe), die ein Messreihen-Objekt übergeben bekommt und die ein neues Messreihen-Objekt zurückgibt, das die

Messwerte genau einmal enthält, die in beiden Messreihen vorkommen, aus {4,4,3,2,3} und {4,1,1,2,2} soll z. B. {4,2} werden (Reihenfolge wieder egal).

Tipp: Um die Klausur zu simulieren, lösen Sie die Aufgaben zuerst auf Papier und schreiben Sie sie dann genau ab.

31. Aufgabe (4 Punkte, Wiederholung, einfache Klasse)

Zu realisieren ist in dem von der Veranstaltungsseite herunterladbaren Projekt eine Klasse Person, wobei jede Person einen Namen, eine Personalnummer und eine Sammlung von Qualifikationen hat. Vereinfachend kann angenommen werden, dass jede Qualifikation ein einfacher String repräsentiert ist. Realisieren Sie folgenden Konstruktor und folgende Methoden. Stellen Sie sicher, dass alle gegebenen Tests erfolgreich abgeschlossen werden.

Constructor Summary
Person(String name, Integer pnr) Konstruktor, dem ein Name und eine Personalnummer übergeben werden, die Qualifikationssammlung ist anfänglich leer

Method Summary	
Integer	**getPnr**() Methode gibt die Personalnummer zurück.
ArrayList <String>	**getQualifikationen**() Methode gibt Sammlung aller Qualifikationen der Person zurück.
void	**qualifikationHinzu**(String quali) Methode fügt die als Parameter übergebene Qualifikation hinzu, wobei, falls die Qualifikation schon existiert, sie nicht doppelt eingetragen wird.
int	**anzahlQualifikationen**() Methode gibt die Anzahl der Qualifikationen der Person zurück.
boolean	**hatQualifikation**(String quali) Methode überprüft, ob eine Person die als Parameter übergebene Qualifikation hat.
boolean	**hatQualifikationen**(ArrayList<String> qualis) Methode prüft, ob die Person die als Parameter übergebenen Qualifikationen alle erfüllt.
boolean	**gleichQualifiziert**(Person m) Methode stellt fest, ob die übergebene Person genau die gleichen Qualifikationen der betrachteten Person (this) hat.
boolean	**besserQualifiziertAls**(Person m) Methode stellt fest, ob die Person (this) besser qualifiziert als die als Parameter übergebene Person ist; dies ist der Fall, wenn die Person alle Qualifikationen der übergebenen Person und mindestens eine zusätzliche hat.
String	**toString**() Methode zum Ausgeben einer Person.

Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

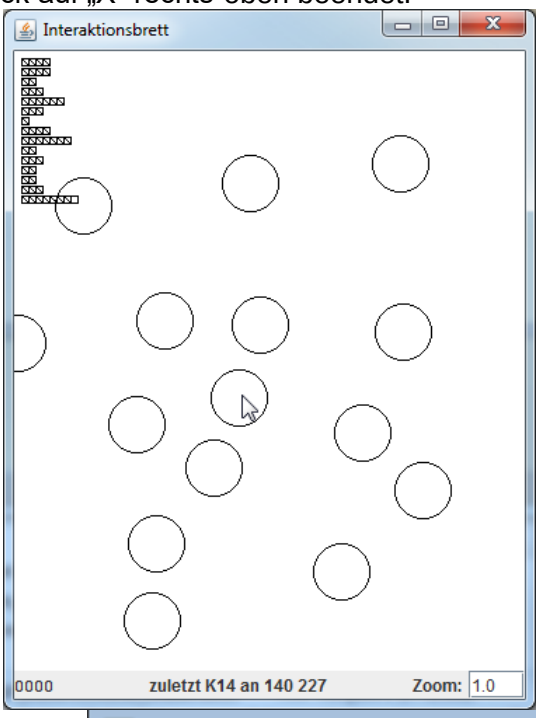
Grundlagen Programmierung

Wintersemester 2023/24
Aufgabenblatt 9

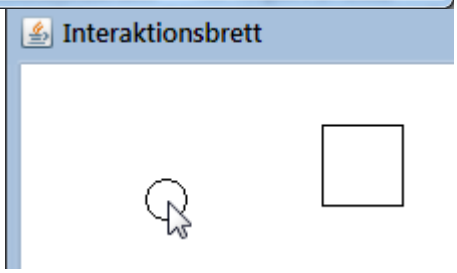
32. Aufgabe (4 Punkte, reaktive Programmierung, Teil 2)

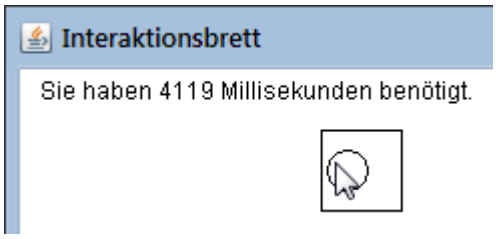
Zur Veranschaulichung des Ergebnisses der Teilaufgaben sind von der Veranstaltungsseite die Programme miniinterkation.exe und basketball.exe in Zip-Dateien zu laden und in Windows 10 auszuführen. Da die Programme nicht in Java geschrieben sind, sieht das Interaktionsbrett etwas anders aus. Die Programme werden mit einem Klick auf „X“ rechts-oben beendet.

a) Lesen Sie zunächst die erweiterten Hintergrundinformationen zur reaktiven Programmierung mit der Klasse Interaktionsbrett am Ende des Aufgabenblatts durch.
Probieren Sie das von der Veranstaltungsseite erhältliche Projekt AufgabeMiniInteraktion aus, verschieben Sie dabei die angezeigten Kreise. Lesen Sie sich das Programm genau durch und versuchen Sie seine Arbeitsweise zu verstehen. Das Programm funktioniert für zwei Kreise und ist dabei erschreckend mit Copy & Paste entstanden. Schreiben Sie das Programm so um, dass es für eine flexible Anzahl von Kreisen funktioniert. Die Anzahl wird mit dem Konstruktor übergeben. Die Graphiken am oberen Rand können Sie ignorieren oder optional programmieren. Wichtig ist, dass in der Fußzeile erkennbar ist, welcher Kreis zuletzt angeklickt wurde.



b) [etwas schwieriger] Schreiben Sie ein neues Programm mit Hilfe der Klasse Interaktionsbrett, das einen Basketballwurf simulieren soll. Erzeugen Sie dazu einen verschiebbaren Kreis (Ball) und ein unverschiebbares Rechteck (Korb) an einer beliebigen sichtbaren Position auf dem Interaktionsbrett. Der Ball gilt als in den Korb geworfen, wenn der Ball mit der Maus verschoben und innerhalb des Rechtecks losgelassen wurde. Ergänzen Sie dazu die notwendigen Implementierungen der Methoden mitMausverschoben, mitMausAngeklickt und mitMausLosgelassen. Befindet sich der Ball im Korb, soll er nicht mehr bewegt werden können. Die Bilder zeigen zwei typische Spielsituationen nachdem auch c) gelöst wurde. Ergänzen Sie das Programm so, dass am Ende des Spiels die verbrauchte Zeit angezeigt wird





c) Wahrscheinlich haben Sie in den vorherigen Aufgaben das gesamte Programm in einer Klasse realisiert. Ändern Sie das Programm so ab, dass der Korb eine eigene Klasse wird. Dabei soll der Korb im Konstruktor das genutzte Interaktionsbrett als Parameter erhalten und sich dann an zufälliger Stelle zeichnen. Weiterhin soll der Korb eine Methode getroffen beinhalten, mit der überprüft wird, ob der Ball den Korb getroffen hat, der Korb muss dazu natürlich wissen, an welcher Position er steht.

Hintergrund: Reaktive Programmierung mit der Klasse Interaktionsbrett

Man spricht bei einer Software von einem reaktiven System, wenn es seine Berechnungen nicht selbständig ausführt, sondern von außen angestoßen wird. Grafische Oberflächen sind

ein Beispiel für ein reaktives System, da z. B. erst etwas ausgeführt wird, wenn der Nutzer einen Knopf anklickt.
Bei der Programmierung muss man die Reaktion auf eine solche Aktion programmieren. Typischerweise schreibt man dazu Methoden, die vom umgebenden System (z. B. Betriebssystem oder virtuelle Maschine) aufgerufen werden. Damit das System weiß, welche Methoden genutzt werden, muss dies „irgendwo“ vereinbart sein.
Die Klasse Interaktionsbrett bietet die Möglichkeit, mit jeder der unterstützten Objektarten (Punkt, Linie, Kreis, Rechteck, Text) zu interagieren, d. h. auf Mausaktionen mit diesen Objekten zu reagieren. Dazu gibt es beim Zeichnen der Objekte jeweils eine zweite Methode mit zwei zusätzlichen Parametern mit den Typen Object (es kann also ein Objekt einer beliebigen Klasse übergeben werden) und String, wie folgender Ausschnitt aus den Methoden für einen Kreis zeigt.

Method Summary

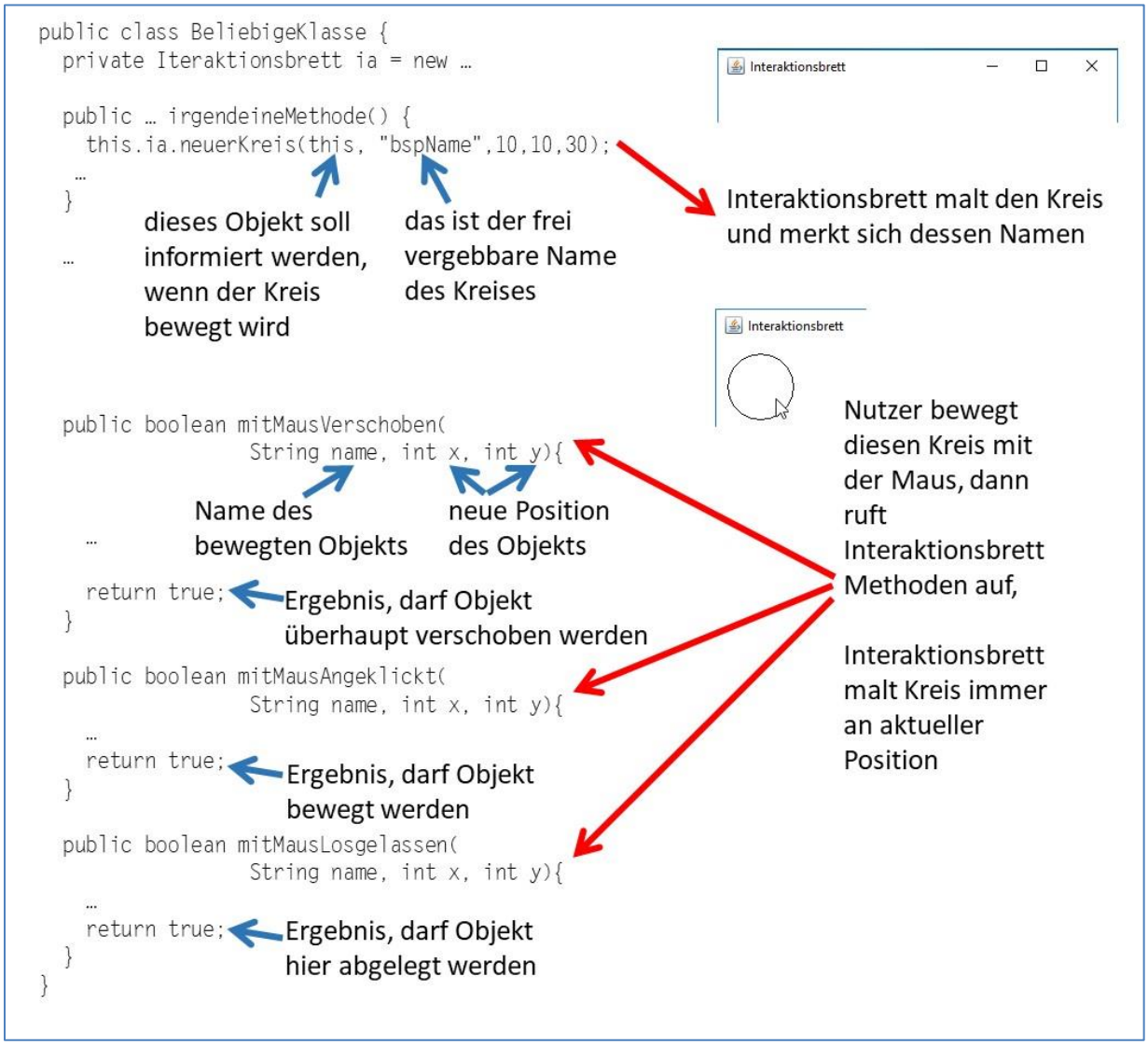
void	**neuerKreis**(int x, int y, int radius) Methode zum Zeichnen eines neuen Kreises.
void	**neuerKreis**(Object quelle, String name, int x, int y, int radius) Methode zum Zeichnen eines neuen Kreises, der verändert und dessen Nutzung beobachtet werden kann.

Die beiden Parameter haben folgende Bedeutung. Das übergebende Objekt wird gegebenenfalls benachrichtigt, falls der Kreis mit der Maus „bearbeitet“ (geklickt, verschoben, neu platziert) wurde. Das „gegebenenfalls“ bezieht sich darauf, dass das benachrichtigte Objekt bestimmte Methoden anbieten muss, die die Maus-Aktionen verarbeiten. Diese Methoden werden gleich vorgestellt. Der übergebene String steht für einen Namen des Objektes, der bei der Verarbeitung der Mausaktionen mit übergeben wird. Das Objekt, das die Mausaktionen verarbeitet, kann so mehrere Objekte an ihren Namen unterscheiden.
Möchte man, dass ein Objekt, das einen Kreis zeichnet, auch auf die zugehörigen Mausaktionen reagieren kann, muss sich das Objekt selbst (this als erster Parameter) als zu benachrichtigendes Objekt übergeben.
Möchte man im benachrichtigten Objekt auf Maus-Aktionen reagieren, muss man zumindest eine der drei folgenden Methoden realisieren, die wie folgt auch in der Klassendokumentation beschrieben sind. Es gilt dabei, dass man mitMausVerschoben nur nutzen kann, wenn mitMausAngeklickt erlaubt ist (Rückgabe true). Die Methode mitMausLosgelassen ist nur erlaubt, wenn mitMausVerschoben erlaubt ist. In den meisten Fällen ist es sinnvoll, dass alle drei Methoden *true* als Ergebnis liefern.
Das Verschieben übernimmt dabei das Interaktionsbrett und zeichnet das verschobene Objekt selbst neu.

Will man die Maussteuerungsmöglichkeiten nutzen, muss das mit dem graphischen Element übergebene Objekt eine oder mehrere der folgenden Methoden implementieren, die dann vom Interaktionsbrett bei einer Maus-Aktion aufgerufen werden.

- `public boolean mitMausVerschoben(String name, int x, int y)`
  Das Objekt wird informiert, dass ein graphisches Element mit Namen name an die Position (x,y) verschoben wurde, die zugehörige Mausbewegung ist beendet. Mit dem Rückgabewert kann man mitteilen, ob die Verschiebung überhaupt gewünscht ist (true) oder nicht (false).

- `public boolean mitMausAngeklickt(String name, int x, int y)`
  Das Objekt wird informiert, dass ein graphisches Element mit Namen name an der Position (x,y) gerade angeklickt wurde, die zugehörige Mausbewegung beginnt gerade. Mit dem Rückgabewert kann man mitteilen, ob eine Bearbeitung (konkret eine Verschiebung) überhaupt gewünscht ist (true) oder nicht (false).
- `public boolean mitMausLosgelassen(String name, int x, int y)`
  Das Objekt wird informiert, dass ein graphisches Element mit Namen name gerade an die Position (x,y) verschoben und an dieser Position losgelassen wurde, die zugehörige Mausbewegung endet gerade. Mit dem Rückgabewert kann man mitteilen, ob das Ablegen des Elements an dieser Stelle überhaupt gewünscht ist (true) oder nicht (false). Der sicherlich selten genutzte Fall false hat nur Auswirkungen, wenn der Nutzer zum nächsten Zeitpunkt auf eine Stelle klickt, an der sich kein auswählbares graphisches Element befindet. Dann wird das zuletzt benutzte Element genutzt und z. B. wieder verschoben.



Weitere Informationen zum Interaktionsbrett sind auch https://youtu.be/vGryrMcNj6s (38:04) zu entnehmen.