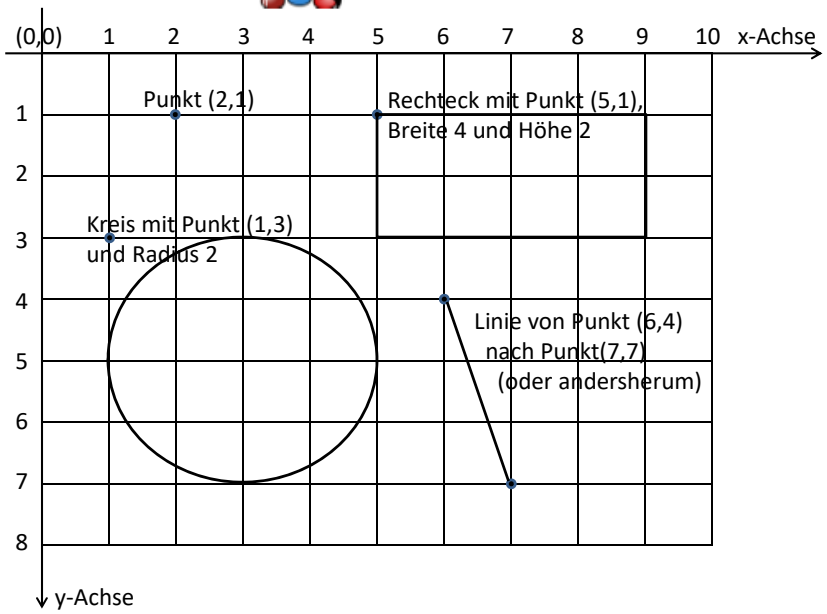
Prof. Dr.-Ing. R. Roosmann, M. Teuber, R. Koller

Aufgabenblatt 4

Ziel dieses Aufgabenblattes ist es, Sie mit Objektmethoden vertraut zu machen sowie Ihnen mehr Sicherheit im Verständnis der Klasse-Objekt Beziehung zu vermitteln. Des Weiteren sollen Sie die Unterschiede zwischen Klassen und elementaren Datentypen in Java verstehen lernen (auch wenn dies nicht wesentlich zum Verständnis der OOP ist, so ist es wichtig zum Programmieren mit Java;-).

Abgabe: 11.04.19; Max. erreichbare Punkt: 16; Mindestpunktzahl: 10

Aufgabe 1: Klassen mit Objektvariblen, Konstruktoren und Objektmethoden (5 Punkte) (2er)



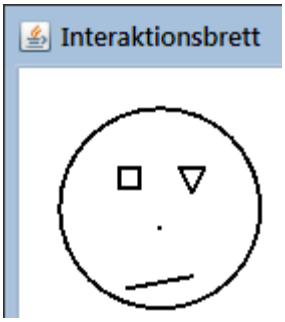
Erstellen Sie je eine Klasse:

a) `Punkt:` besteht aus je einem x- und y-Werte (Koordinaten)
b) `Linie:` mathematisch genauer Strecke, hat konkreten Startpunkt und konkreten Endpunkt
c) `Kreis`: hat einen Aufhängepunkt links-oberhalb des Kreises und einen ganzzahligen Radius
d) `Rechteck:` wird typischerweise ausgehend von der linken oberen Ecke, der Breite und der Höhe definiert
e) `Dreieck`: überlegen Sie, wie man mit möglichst wenig Informationen ein Dreieck genau beschreiben kann

Dabei ist zu beachten, dass die Koordinatenwerte (x- und y-Werte) nur ganzzahlig sein können. Wie auf der Skizze oben angedeutet, ist es bei grafischen Systemen häufig der Fall, dass die positiven Werte der x-Achse von links nach rechts verlaufen, die positiven Werte der y-Achse aber von oben nach unten. So liegt der Ursprung des Bildschirms, der Punkt (0,0), links oben in der Ecke.
Überlegen Sie sich sinnvolle Objektvariablen und definieren Sie dann Konstruktoren. Erweitern Sie Ihre Klassen um getter- und setter-Methoden, um die Werte der Objektvariablen über diese Methoden zu setzen bzw. abzufragen.

Aufgabe 2: getter- und setter-Methoden, Methoden nutzen Methoden (3 Punkte)
 (2er)

Gegeben seien Ihre Klassen `Punkt`, `Linie`, `Kreis`, `Rechteck` und `Dreieck` aus der Aufgabe 1.

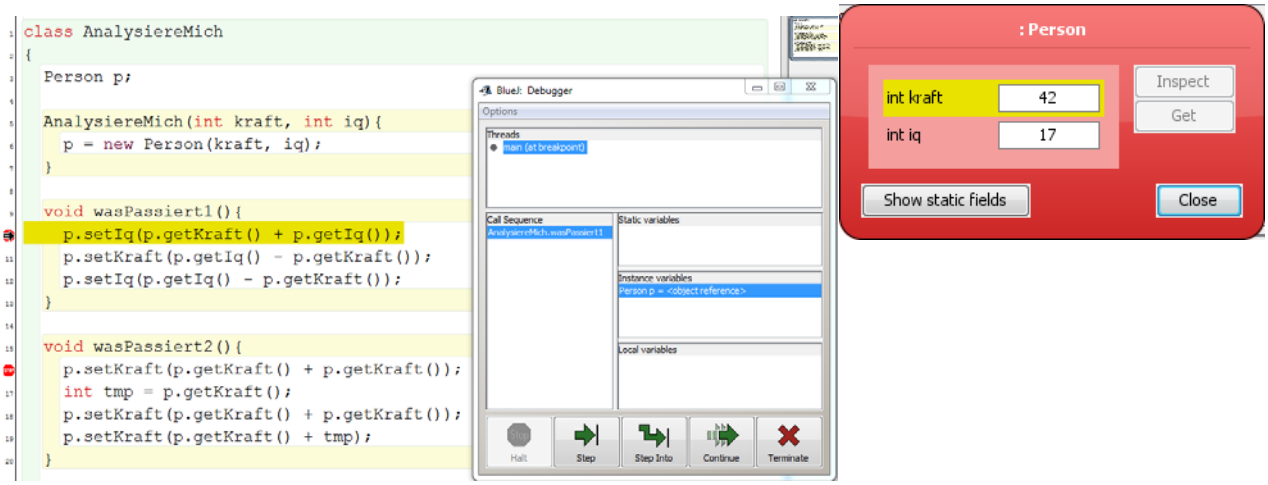


a) Ergänzen Sie in jeder Ihrer Klassen eine Methode `void darstellen(Interaktionsbrett ib)` mit der sich das jeweilige Objekt auf dem übergebenen Interaktionsbrett zeichnet. Nutzen Sie dabei die getter-Methoden der in Aufgabe 1 erstellten Klassen, um die Parameter für die Zeichenmethoden von `ib` zu erhalten.

b) Ergänzen Sie eine Klasse `GeoobjektSpielerei` mit einer Methode, die ein Interaktionsbrett-Objekt nutzt und die dann von jeder Ihrer Klassen ein Objekt erzeugt und die `darstellen`-Methode aus a) so aufruft, dass ein Bild vergleichbar dem rechts-oben entsteht.

Aufgabe 3: Nutzung des Debuggers (3 Punkte)  (2er)

Laden Sie aus OSCA die Datei Aufgabe3.zip und entpacken diese auf Ihrem lokalen Z-Laufwerk. Importieren Sie die Java-Dateien in ein neues BlueJ-Projekt namens Aufgabe_3. Betrachten Sie sich zunächst die Klasse `Person` und dann die Klasse `AnalysiereMich`. Analysieren Sie die Methoden `wasPassiert1` und `wasPassiert2` mit dem Debugger von BlueJ. Markieren Sie dazu jeweils die erste Zeile der Methoden in der Klasse `AnalysiereMich` als Breakpoint. Erzeugen Sie dann ein Objekt der Klasse `AnalysiereMich` und rufen Sie die Methoden einzeln auf. Fotografieren Sie jeden Schritt des Debuggers während der Ausführung der Methoden und zeigen Sie dazu die momentanen Werte der Objektvariablen des referenzierten Objektes an (Doppelklick auf die Instanz Variable). Das untere Bild zeigt z. B. die Ausgangssituation unmittelbar nach dem Start von `wasPassiert1`, nachdem ein Objekt vom Typ `AnalysiereMich` mit `new AnalysiereMich(42,17)` erzeugt wurde.



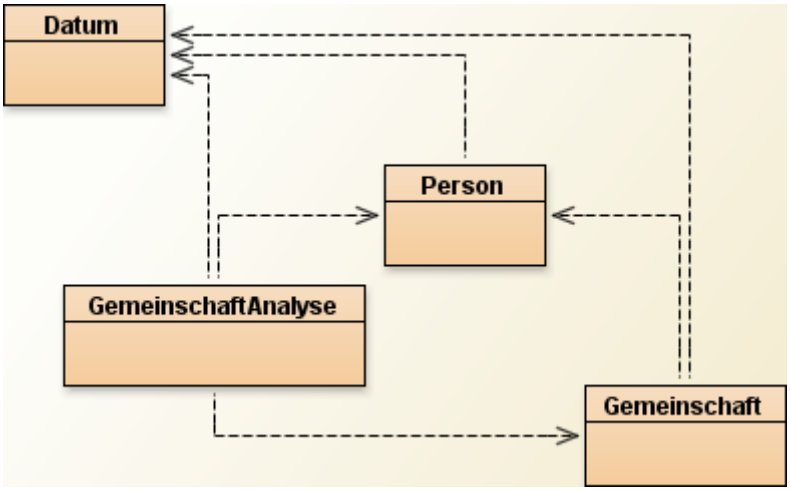
Was vermuten Sie, welchen Effekt haben die beiden Methoden der Klasse `AnalysiereMich`?

Öffnen sie jetzt die Klasse `Person` mit dem Source-Code Editor und ändern Sie den Typ der Objektvariablen `kraft` und `iq` jeweils auf `Integer`. Kompilieren Sie die Klasse `Person`, öffnen dann mit dem Source-Code Editor die Klasse `AnalysiereMich`, ändern in Zeile 17

den Typ der lokalen Variablen `tmp` in `Integer` und setzen in die Zeile 18 einen Breakpoint (hinter die Zuweisung `Integer tmp = p.getKraft();`). Entfernen Sie andere Breakpoints, falls vorhanden. Debuggen Sie jetzt den Source-Code, inspizieren Sie die lokale Variable tmp sowie die Instanz Variable p und dokumentieren wiederum Ihr Ergebniss.

Aufgabe 4: Visualisierung von Referenzen (5 Punkte)  (2er)

Laden Sie die Datei Aufgabe4.zip aus OSCA herunter und entpacken Sie die Dateien auf Ihrem lokalen Z-Laufwerk. Öffnen Sie das BlueJ-Projekt und analysieren Sie die vorhandenen Java-Klassen.
Gegeben sind die Klassen `Datum`, `Person` und `Gemeinschaft`. Weiterhin gibt es die Klasse `GemeinschaftAnalyse` mit folgendem Inhalt.




```
class GemeinschaftAnalyse {

  void referenzen(){
    Datum d = new Datum(1,1,1990);
    Person p = new Person("Ute","Mai",d);
    Gemeinschaft g = new Gemeinschaft(p,
                         new Person("Oleg","Loc"),
                         new Datum(2,2,2013));
    g.getPartner1().setNachname("Loca");
    g.setPartner2(g.getPartner1());
    p.setVorname("Udo");
  }
}
```

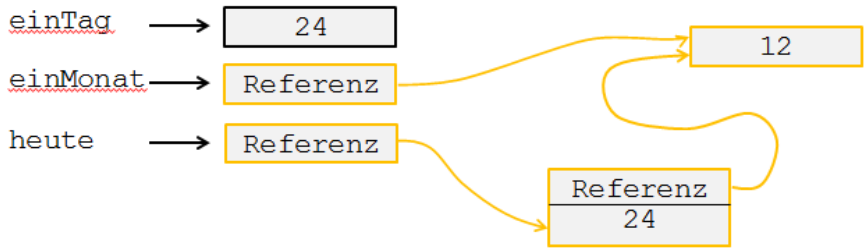
Erstellen Sie ein Objekt der Klasse `GemeinschaftAnalyse` und zeichnen Sie für jede Programmzeile der Methode `referenzen()`, die Situation nach der Ausführung bzgl. der Variablen und der Referenzen auf Objekte im Speicher.

Hinweis: Evtl. zeichnen Sie direkt die Situation nach der dritten Zeile, da man die anderen dann durch Löschen und Hinzufügen leichter herstellen kann. Handzeichnungen mit lesbaren Markierungen von Veränderungen sind natürlich auch erlaubt.

Orientieren Sie sich bei der Darstellung an der Vorgehensweise in der Vorlesung. Auf der nächsten Seite finden Sie das verwendete Beispiel aus der Vorlesung.

Beispiel zur Darstellung der Situation der Variablen und Referenzen auf Objekte im Speicher aus der Vorlesung:

```
class Datum {
      int tag; Integer monat;
      //Konstruktor
      Datum(int tag, Integer monat) {this.tag=tag;
         this.monat=monat;}
} …
int einTag = 24;
Integer einMonat = 12;
Datum heute = new Datum(einTag, einMonat);
```

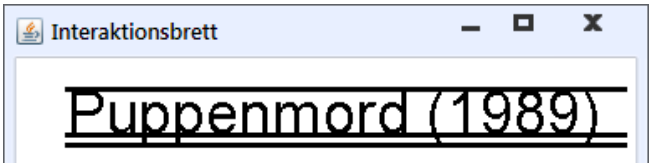


Freiwillige Aufgabe

Aufgabe 5 Spielerei mit Interaktionsbrett, Einstieg in reaktive Programmierung: nicht unsere Software entscheidet welche Methode ausgeführt wird, sondern dies hängt z. B. von Mausaktionen ab (0 Punkte)

Schreiben Sie eine neue Klasse `InteraktionsbrettSpielerei.java`, mit der die Klasse `Interaktionsbrett` weiter untersucht werden soll. Ihre Klasse soll zumindest eine Objektvariable der folgenden Art enthalten: `Interaktionsbrett ib;`

a) Die Klasse bietet u. a. Methoden (s.u.), mit denen ein Text auf dem Interaktionsbrett platziert werden und mit dem man die Länge eines Textes bestimmen kann (Font und die Fontgröße 12 pt sind nicht änderbar). Der Ausgangspunkt des Textes befindet sich auf der Basislinie der Zeichen. Um diesen Begriff zu verdeutlichen, schreiben Sie eine Methode, die möglichst genau das obige Bild als Ergebnis hat (Zoom 3).



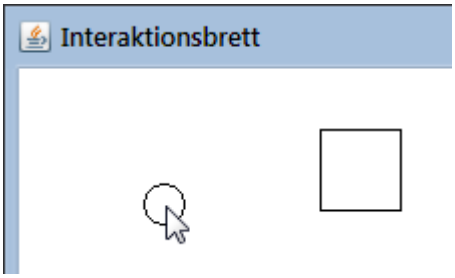
Method Summary	
void	**neuerText**(int x, int y, String text) Methode zur Ausgabe eines Textes.
int	**textlaenge**(String text) Berechnet die Länge eines Textes für eine mögliche graphische Ausgabe.

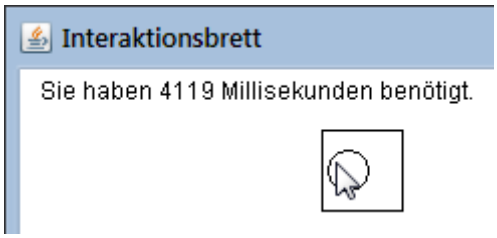
b) Lesen Sie zunächst die Hintergrundinformationen zur reaktiven Programmierung mit der Klasse `Interaktionsbrett` am Ende des Aufgabenblatts durch und probieren Sie das BlueJ-Projekt aus Aufgabe5.zip (siehe OSCA) aus. Ergänzen Sie in Ihrer Klasse `InteraktionsbrettSpielerei` folgende Methoden.

```
public void bewegDich(){
    ib.neuerKreis(this,"K1",30,30,10);
}

public Boolean mitMausAngeklickt(String name, int x, int y){
    return true;
}

public Boolean mitMausVerschoben(String name, int x, int y){
    return true;
}
```

Was beobachten Sie, wenn Sie versuchen, den Kreis mit gedrückter linker Maustaste (in der virtuellen Umgebung langsam) zu verschieben? Was beobachten Sie, wenn Sie den Rückgabewert in der Methode `mitMausAngeklickt` auf `false` setzen?

c) [ab hier schwierig] 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 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 d) gelöst wurde. Die folgende Methode kann beim Platzieren von Ball und Korb hilfreich sein.

Method Summary

int	**zufall**(int start, int ende) Methode zur Erzeugung einer ganzzahligen Zufallszahl zwischen (einschließlich) den übergebenen Grenzen.

d) Ergänzen Sie das Programm so, dass am Ende des Spiels die verbrauchte Zeit angezeigt wird.

e) 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 bei der Erzeugung der Objekte jeweils eine zweite Methode mit zwei zusätzlichen Parametern mit den Typen Objekt 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 ihrem 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 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). MitMausLosgelassen ist nur erlaubt, wenn mitMausverschoben erlaubt ist.

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 befinde. Dann wird das zuletzt benutzte Element genutzt und z. B. wieder verschoben.

Objekte können dem Interaktionsbrett mitteilen, dass sie über gedrückte Tasten informiert werden wollen. Hierzu dient die Methode `willTasteninfo()`. Das zu informierende Objekt muss dann eine Methode der folgenden Form realisieren:
`public void tasteGedrueckt(String s)`

Viel Erfolg!!