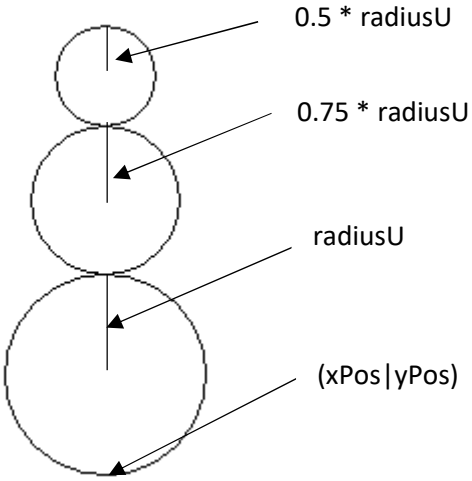
Objektorientiertes Programmieren mit GPanel-Grafik

Das Projekt MeinPark, V

Aufgabe 11: Die Klasse Schneemann

Damit der Park komplett ist, fehlt nur noch ein Schneemann. Dieser besteht aus drei übereinander liegenden Kreisen, wobei der mittlere Kreis einen Radius besitzt, der nur 75% des Radius des unteren Kreises beträgt. Der obere Kreis ist sogar nur halb so groß, wie der untere.
Außer, dass der Schneemann gezeichnet werden kann, soll er (später) auch ein wenig schmelzen können. Dadurch wird er an gleicher Stelle neu gezeichnet, allerdings ist der Radius der Kreise nun ein wenig kleiner. Da es von der Entfernung zur Sonne abhängig sein soll, ob der Schneemann schmilzt, benötigt er außerdem eine Methode zur Berechnung der Distanz zu einem anderen Punkt:



Schneemann
gp: GPanel xPos: Zahl yPos: Zahl radiusU: Zahl
zeichne() schmelzeEinWenig() distanz()

a) Implementiere die Klasse `Schneemann`:

- Der Konstruktor enthält wieder vier Parameter: das GPanel, die x- und y-Position des Punktes, bei dem der untere Kreis den Boden berührt, sowie den Radius des unteren Kreises.

- Die Methode

  ```
  public void zeichne()
  ```

  soll den Schneemann an der entsprechenden Position zeichnen.

Objektorientiertes Programmieren mit GPanel-Grafik

Das Projekt MeinPark, V

- Die Methode

  ```
  public void schmelzeEinWenig()
  ```

  soll den Schneemann um 5% kleiner werden lassen (d.h. den Schneemann löschen und anschließend kleiner neu zeichnen).
  Tipp: Vielleicht hilft dir hier wieder eine private Mehtode `verschwinde()`.

- Die Methode

  ```
  public double distanz(int pX, int pY)
  ```

  berechnet den Abstand des Mittelpunktes des mittleren Kreises des Schneemanns zum Punkt mit den Koordinaten (`pX|pY`).

  Tipp: Den **Abstand zweier Punkte** `(p1X|p1Y)` und `(p2X|p2Y)` kannst du mit dem Satz des Pythagoras ermitteln:

  ```
  Math.sqrt((p1X-p2X)*(p1X-p2X) + (p1Y-p2Y)*(p1Y-p2Y))
  ```

b) Erweitere dein Hauptprogramm `MeinPark` aus Aufgabe 9 derart, dass auch ein Schneemann gezeichnet wird und zwar circa in die Mitte des GPanels. Der Radius des unteren Kreises sollte 50 betragen.
Beachte, dass die Blumen nicht vom Schneemann überdeckt werden sollten und ggf. an anderer Stelle gezeichnet werden müssen.
(Tipp: Betrachte die linke obere Ecke wegen der Sonne und ein Rechteck in der Mitte wegen des Schneemanns als „Sperrgebiet“ und erzeuge für jede Blume solange die zufälligen Blumenkoordinaten, bis sie außerhalb des Sperrgebietes liegen).
Wie bei Aufgabe 9 sollen die Blumen immer wieder neu gezeichnet werden, damit die Sonne sie nicht löschen kann. Der Schneemann soll allerdings „ausradiert“ werden, wenn die Sonne ihn berührt. Auch braucht er an dieser Stelle noch nicht zu schmelzen.

c) Ergänze nun noch die Dokumentation der Klasse Schneemann, so wie du es in Aufgabe 10 gelernt hast.

Objektorientiertes Programmieren mit GPanel-Grafik

Das Projekt MeinPark, V

Aufgabe 12: Getter und Setter

Im Hauptprogramm soll nun permanent der Abstand zwischen der Sonne und dem Schneemann geprüft werden. Falls er klein genug ist, soll der Schneemann ein wenig schmelzen, die Sonne dagegen ein wenig größer werden (z.B. um 10%).
Mit der Methode `distanz` aus Aufgabenteil 11a) sollte das Ermitteln des Abstands zwischen Sonne und Schneemann eigentlich kein Problem sein, allerdings erwartet diese Methode als Parameter die Koordinaten der Sonne und diese sind **private** Attribute der Klasse Sonne. Von anderen Klassen und insbesondere vom Hauptprogramm aus kann also nicht auf sie zugegriffen werden.
Zur Lösung dieses Problems dienen so genannte **Getter**. Sie stellen Lesemethoden dar, mit deren Hilfe man auf die Werte von Attributen eines Objekts zugreifen kann und sind immer gleich aufgebaut. Für ein `int`-Attribut namens `attributA` sieht der Code der zugehörigen `get`-Methode, z.B. folgendermaßen aus:

```
public int getAttributA() {
  return attributA();
}
```

Verfügt also ein Objekt namens `obj` über ein Attribut namens `attributA`, so kann mit Hilfe der **`get`-Methode** durch `obj.getAttributA()` außerhalb der Klasse auf den Wert des Attributs zugegriffen werden.

a) Erweitere die Klasse `Sonne` um die Methoden

```
public void getPosX()
```

und

```
public void getPosY()
```

mit denen nun auch von anderen Klassen aus auf die Position der Sonne zugegriffen werden kann.

Wie oben beschrieben, soll nicht nur der Schneemann ein wenig schmelzen, wenn der Abstand zur Sonne klein genug ist – auch die Größe der Sonne, d.h. ihr Radius, soll verändert werden. Hier ist also wiederum ein Zugriff vom Hauptprogramm, also von außerhalb, auf ein privates Attribut, nämlich den Radius, nötig. Allerdings muss der Radius der Sonne nicht nur gelesen, sondern auch verändert werden können. Dies

Objektorientiertes Programmieren mit GPanel-Grafik

Das Projekt MeinPark, V

geschieht mit dem dazugehörigen **Setter**: Innerhalb einer **`set`-Methode** wird ein Attribut überschrieben, indem man ihm den Wert des Parameters der Methode zuweist. Für ein `int`-Attribut namens `attributA` sieht der Code der zugehörigen `set`-Methode, z.B. folgendermaßen aus:

```
public void setAttributA(int pAttributA) {
  attributA = pAttributA
}
```

Durch `obj.setAttributA(5)` könnte nun z.B. der Wert von `attributA` des Objekts `obj` von außerhalb der Klasse aus auf 5 gesetzt werden

b) Erweitere die Klasse `Sonne` um die Methoden

```
public int getRadius()
```

und

```
public void setRadius(int pRadius)
```

mit denen nun auch von anderen Klassen aus auf den Radius der Sonne zugegriffen und dieser verändert werden kann.
(Die `set`-Methoden für die Attribute `gp`, `xPos` und `yPos` werden hier nicht benötigt).

c) Erweitere den Code in `MeinPark` nun so, dass immer dann, wenn sich die Sonne bewegt hat, die Distanz von der Sonne zum Schneemann ermittelt wird (genauer: zum Mittelpunkt des mittleren Kreises des Schneemanns). Ist diese Distanz kleiner als der doppelte Durchmesser der Sonne, soll der Schneemann wenig schmelzen, die Sonne dagegen größer werden (z.B. um 5%).
Verwende dabei die `get`- und `set`-Methoden der Klasse `Sonne`!

Tipp: Ist die Sonne einmal in der Nähe des Schneemanns, so bleibt sie dort evtl. für einige Zeit. Dadurch schmilzt der Schneemann sehr schnell immer weiter. Es kann daher sinnvoll sein, nach dem Aufruf der Methode `schmelzeEinWenig()` mit dem `delay`-Befehl eine kleine Pause einzubauen.

d) Damit die Sonne nicht immer weiterwächst: Stoppe sämtliche Bewegungen, wenn der Radius des unteren Kreises des Schneemanns kleiner als 5 Pixel ist. Wie bisher wird durch Mausklick wird das Fenster geschlossen.