SoSe 2020

Prof. Dr. Margarita Esponda

Objektorientierte Programmierung

6. Übungsblatt

Lernziel: Zusicherungen, Programmverifikation und Einstieg in Java.

1. Aufgabe (10 Punkte)

Folgendes Python-Funktion berechnet ohne Division und Multiplikation die kleinste ganze Zahl, die größer oder gleich der Wurzel einer eingegebenen ganzen Zahl **n** ist.

Es wird für den Algorithmus die Gültigkeit folgender Gleichung verwendet:

$$n^2 = S(n) + S(n-1) \quad mit \quad S(n) = \sum_{i=0}^{n} i = \frac{1}{2}\ n(n+1)$$

```
def root(n):
    {P ≡ n > 0}
    r = 0
    k = 1
    {INV} ≡ {r == S(k-1) ∧ (S(k-1) + S(k-2)) < n}

    while (r + r + k) < n:
        r = r + k
        k = k + 1

    {INV ∧ ¬B} ≡ {(r == S(k-1) ∧ (S(k-1) + S(k-2)) < n) ∧ (r + r + k) ≥ n}

    {Q} ≡ {(k-1)² < n ≤ k²}        (1)

    return k
```

a) Zeigen Sie, dass ***INV*** eine gültige Invariante der **while**-Schleife des Programms ist.

b) Beweisen Sie die partielle Korrektheit der Programmformel (1).

c) Zeigen Sie, dass die Funktion terminiert, d.h. geben Sie eine Funktion τ an und beweisen Sie, dass τ eine Terminierungsfunktion für die **while**-Schleife ist.

Wichtige Hinweise:

1) In allen Beweisen müssen Sie die jeweils verwendeten Hoare-Regeln angeben.
2) Jeder Beweisschritt muss begründet werden.
3) Schreiben Sie bei Bedarf Kommentare, die den Tutoren die Korrektur der Lösungen erleichtern können.

2. Aufgabe (16 Punkte)

a) (1 P.) Programmieren Sie die Klasse **Rectangle**, die einfache Rechtecke (parallel zum Koordinatensystem) darstellt, die in der Vorlesung diskutiert worden ist.

b) (1 P.) Folgende zwei Konstruktoren sollen in der Klasse beinhaltet sein:

```
public  Rectangle( int  x, int  y, int  width, int  height ) {
        this .x = x;
        this .y = y;
        this .width = width;
        this .height = height;
}
public  Rectangle() {
        this ( 0, 0, 100, 100 );
}
```

c) (14 P.) Erweitern Sie die **Rectangle**-Klasse um folgende Instanz-Methoden.

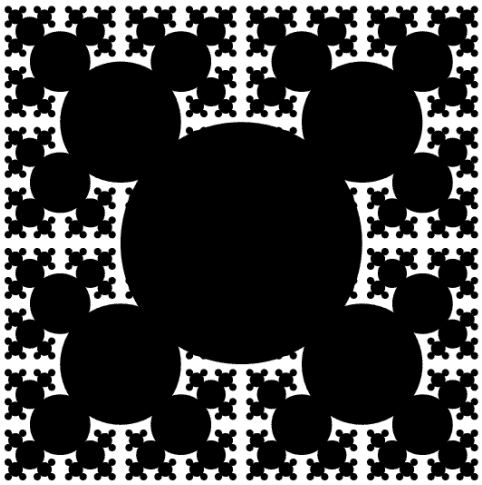
```
/* Ein zweites identisches Rectangle-Objekt wird erstellt */
public Rectangle clone()

/* Ein Rectangle-Objekt vergleicht alle seine Instanzvariablen mit den
   Instanzvariablen eines zweiten Rectangle-Objekts r, das als Parameter der
   Methode übergeben wird. */
public boolean equal( Rectangle r )

/*  Ein Rectangle-Objekt vergleicht seine Fläche mit der Fläche eines zweiten
   Rectangle-Objekts r, das als Parameter übergeben wird und ergibt -1, 0, oder 1, je
   nachdem, ob die Fläche jeweils kleiner, gleich oder größer ist. */
public int compareAreaTo( Rectangle r )

/* Testet, ob ein Rectangle r komplett in dem Rectangle-Objekt, das gerade
   die Methode ausführt, beinhaltet ist */
public boolean contains( Rectangle r)

/* Ein Rechtecks-Objekt überprüft, ob eine Überlappung mit dem Rechteck r
   existiert. */
public boolean overlaps( Rectangle r )

/* Berechnet die rechteckige Schnittfläche, die entsteht, wenn ein Rectangle-Objekt sich mit
   einem zweiten  Rectangle-Objekt überlappt. Wenn die Rectangle-Objekte sich nicht
   überlappen, wird die Konstante null zurückgegeben.*/
public Rectangle section(Rectangle r)

/* Berechnet das kleinste Rectangle-Objekt, das ein Array von Rectangle-Objekten
   umrahmen kann. */
public static Rectangle smallestBoundingRectangle ( Rectangle[] recs )
```

d) Laden Sie die Klasse **TestRectangle** aus den Veranstaltung-Ressourcen runter, und starten Sie die Programmausführung mit der **TestRectangle**-Klasse.

Innerhalb der **TestRectangle**-Klasse werden Ihre Methoden getestet.

3. Aufgabe (4 Punkte)

Mit Hilfe der StdDraw.java Klasse schreiben Sie eine rekursive Funktion, die folgendes "Mickey mouse"-Fractal malt.



Wichtige Hinweise für die Java-Programmierung:

1) Verwenden Sie selbsterklärende Namen von Variablen und Methoden.
2) Für die Namen aller Bezeichner müssen Sie die **Java-Konventionen** verwenden.
3) Verwenden Sie vorgegebene Klassen- und Methodennamen.
4) Methoden sollten klein gehalten werden, sodass auf den ersten Blick ersichtlich ist, was diese Methode leistet.
5) Methoden sollten möglichst wenige Argumente haben.
6) Methoden sollten entweder den Zustand der Eingabeargumente ändern oder einen Rückgabewert liefern.
7) Verwenden Sie geeignete Hilfsvariablen und definieren Sie sinnvolle Hilfsmethoden in Ihren Klassendefinitionen.
8) Zahlen sollten durch Konstanten ersetzt werden.
9) Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.