12. Aufgabenblatt zur Vorlesung

Konzepte der Programmierung

WiSe 2024/25

Wolfgang Mulzer

Abgabe am Freitag, den 31. Januar 2025, 12 Uhr

Dies ist das letzte reguläre Aufgabenblatt.

Aufgabe 1 π approximieren *10 Punkte*

Schreiben Sie ein Scala-Programm zur Approximation des Wertes von π mittels eines so genannten *Monte-Carlo-Verfahrens*. Erzeugen Sie dazu Paare von Gleitkomma-Zufallszahlen $(x, y) \in [0, 1]^2$. Testen Sie dann, ob $x^2 + y^2 \leq 1$ ist. Wenn ja, so liegt der Punkt in dem Viertelkreis mit Fläche $\pi/4$. Sie sollten also $\pi/4$ annähern können, indem Sie das Verhältnis der Anzahl der erfolgreichen Versuche zur Gesamtzahl der Versuche bestimmen. Experimentieren Sie sowohl mit der Anzahl der Versuche wie auch mit der Genauigkeit der Zahlen. Dokumentieren Sie dies.

Hinweis: Um Zufallszahlen in Scala zu erzeugen, können Sie wie folgt vorgehen: Importieren Sie zunächst mit `import java.util.Random` einen *Pseudozufallsgenerator*. Dann können Sie mit `val r: Random = new Random()` ein neues Exemplar des Pseudozufallsgenerators anlegen. Neue Zufallszählen zwischen 0 und 1 erhalten Sie dann mit `r.nextFloat` und `r.nextDouble`.

Aufgabe 2 Objekte in Scala *10 Punkte*

(a) Deklarieren Sie in Scala eine Klasse Fahrzeug, die typische Eigenschaften von Fahrzeugen modellieren soll. Ihre Klasse solle mindestens vier Attribute und drei Methoden besitzen. Legen Sie mindestens vier Exemplare der Klasse Fahrzeug an und verdeutlichen Sie die Referenzsemantik für Objekte in Scala.

(b) Nun soll jedesmal, wenn ein neues Fahrzeug erzeugt wird, ein eindeutiges Nummernschild erzeugt werden. Erweitern Sie dazu Ihre Klasse Fahrzeug um eine geeignete Begleit-Objekt und schreiben Sie einen passenden Konstruktur. Schreiben Sie auch eine Methode, welche das Nummernschild eines Fahrzeugs ausgibt.

(c) Definieren Sie mindestens drei Unterklassen für Ihre Fahrzeugklasse und definieren Sie eine geeignete Methode in der Oberklasse, die von den drei Unterklassen überschrieben wird. Geben Sie nun ein Beispiel, das die Inklusionspolymorphie in Scala und den Unterschied zwischen dem statischen und dem dynamischen Typ einer Variable in Scala verdeutlicht.

Aufgabe 3 Vererbung *10 Punkte*

Betrachten Sie die folgenden Scala-Klassen und zeichnen Sie das Vererbungsdiagramm. Was ist die Ausgabe bei einem Aufruf der main-Funktion `test`? Begründen Sie Ihre Antwort.

```
class Place:
    def printMe = println("Buy it.")

class Region extends Place:
    override def printMe = println("Box it.")

class State extends Region:
    override def printMe = println("Ship it.")

class Maryland extends State:
    override def printMe = println("Read it.")

@main
def test: Unit =
    var mid: Region = State()
    var md: State = Maryland()
    var obj: Any = Place()
    var usa: Place  = Region()
    md.printMe
    mid.printMe
    obj.asInstanceOf[Place].printMe
    obj = md
    obj.asInstanceOf[Maryland].printMe
    obj = usa
    obj.asInstanceOf[Place].printMe
```