Name:

Matrikelnummer:

Bearbeitungszeit: 90 min.

Aufgabe 1. In einem wiederholt durchgeführten Experiment wird festgestellt, ob in einem bestimmten Zeitintervall ein radioaktiver Zerfall stattfindet oder nicht. Die Versuchsreihe soll mit Hilfe der zu implementierenden Klassen `Versuchsreihe` und `Versuchsstatistik` simuliert, und die zu erwartenden Ergebnisse analysiert werden.

Die Klasse `Versuchsreihe` soll über den Konstruktor

```
public Versuchsreihe(int anzahl, double wahrscheinlichkeit)
```

verfügen, der eine Versuchsreihe mit `anzahl` Experimenten erstellt, wobei in jedem einzelnen Experiment mit der angegebenen Wahrscheinlichkeit (ein Wert zwischen 0 und 1) ein Zerfall festgestellt wird. Die Methode

```
public int laengsteZerfallfolge()
```

soll die größte unmittelbar aufeinander folgende Anzahl von Zerfällen in der Versuchsreihe zurückgeben. (Beispiel: In 20 Experimenten FTTTFFTTFFFTTTTFFTTT soll T einen Zerfall bezeichnen und F keinen. Das Ergebnis der Methode sollte dann 4 sein. Die 4 relevanten Versuche sind unterstrichen.)

Die Klasse `Versuchsstatistik` soll über folgenden Konstruktor und folgende Methoden verfügen:

```
public Versuchsstatistik(int anzExperimente, int anzVersuchsreihen)
```

soll `anzVersuchreihen` Versuchsreihen mit je `anzExperimente` und einer Zerfallswahrscheinlichkeit von 0.5 erstellen.

```
public int anzZerfallfolgen(int laenge)
```

soll die Anzahl der Versuchsreihen zurückgeben, die eine Folge von zumindest `laenge` unmittelbar aufeinander folgenden Zerfällen aufweisen.

```
public int laengeZerfallfolgen()
```

soll die größtmögliche Länge von Zerfallsfolgen zurückgeben, die in mindestens der Hälfte der Versuchsreihen vorhanden ist. (Beispiel: Wenn von 100 Versuchsreihen 53 eine Zerfallsfolge mit zumindest Länge 6 enthalten, und nur 49 eine solche der Länge 7, dann soll die Methode 6 zurückgeben.)

Aufgabe 2. Gegeben ist die Klasse `Position`, in der die Methode

```
public boolean equals(Object obj)
```

passend überschrieben wurde. Weiters ist das Interface `Roboter` mit den Methoden

```
public Position getPosition()
public void dreheLinks()
public void dreheRechts()
public boolean istFrei()
public void macheSchritt()
```

gegeben. Diese Methoden sollen folgendes leisten: `getPosition()` liefert die aktuelle Position des Roboters, `dreheLinks()` bzw. `dreheRechts()` drehen den Roboter um 90^o nach links bzw. rechts, `istFrei()` liefert `true` wenn der Weg für einen Schritt nach vorne frei ist (ansonsten `false`), und `macheSchritt()` lässt den Roboter einen Schritt nach vorne machen. Wenn der Weg für einen solchen Schritt nicht frei ist, verändert der Roboter seine Position nicht.

Die gegebene Klasse `DefekterRoboter` implementiert das Interface `Roboter`, allerdings liefert die Methode `istFrei()` immer `false`. Weiters verfügt sie über den Konstruktor

```
public DefekterRoboter(String bezeichnung).
```

Schreiben Sie eine Unterklasse `KorrekterRoboter` von `DefekterRoboter`, für die `istFrei()` korrekt arbeitet, z.B. indem der Roboter probeweise einen Schritt ausführt und dann wieder zur Ausgangsposition zurückkehrt.
Hinweis: Sie können annehmen, dass die Position, die der Roboter gerade verlassen hat, frei ist.