Aufgabe: Cyborg (Mehrfachvererbung, Interfaces, Komposition, Enums)

a) Ein **`Mensch`** lässt sich modellieren, indem seine üblichen Tätigkeiten abgebildet werden. Darunter fallen: Essen, Schlafen, Arbeiten und Autofahren.

Ein **`Roboter`** mit einer künstlichen Intelligenz hat einen ähnlichen Satz Tätigkeiten: Aufladen, Warten, Arbeiten und neuerdings – durch den Trend zu selbst fahrenden Autos – auch Autofahren.

Beim Autofahren müssen sowohl Mensch als auch Roboter auf Gefahrensituationen reagieren können. Diese sollen durch ein **`enum`** mit drei Werten modelliert werden: **`GEFAHR_LINKS`**, **`GEFAHR_RECHTS`**, **`GEFAHR_VORNE`**.

Bei einer **`GEFAHR_LINKS`** wird nach **rechts** ausgewichen.

Bei einer **`GEFAHR_RECHTS`** wird nach **links** ausgewichen.

Bei einer **`GEFAHR_VORNE`** wird **gebremst**.

Sowohl Mensch als auch Roboter sollen über eine Methode **`entscheide`** verfügen, in der sie auf eine gegebene Gefahrensituation reagieren. Die Entscheidungen sollen auch durch ein **`enum`** mit den Werten: **`RECHTS`**, **`LINKS`**, **`BREMSEN`**, **`UNENTSCHIEDEN`** abgebildet werden.

Der Mensch schätzt die Situation im Gegensatz zum Roboter in 25% der Fälle nicht genau ein und ist **`UNENTSCHIEDEN`**.

Implementieren Sie die Klassen **`Mensch`** und **`Roboter`** und setzen beide einer Gefahrensituation aus.

b) Ein **`Cyborg`** ist sowohl ein **`Mensch`** als auch ein **`Roboter`**. Trotz des Stresses, den das Aufladen, Essen, Warten, Schlafen, etc. mit sich bringt, fährt auch ein Cyborg gerne Auto und wird dort Gefahrensituationen ausgesetzt.

Wenn sich der Menschanteil und der Roboteranteil in ihrer Entscheidung einig sind, trifft der Cyborg die gleiche Entscheidung.

Wenn der Menschanteil und der Roboteranteil unterschiedlicher Ansicht sind, dann trifft der Cyborg zufällig eine der beiden Entscheidungen.

Ergänzen Sie Ihr Programm um eine Klasse **`Cyborg`**, die dieses Verhalten berücksichtigt.

Schreiben Sie eine **`main`**-Methode, in der Mensch, Cyborg und Roboter Autofahren und geben Sie aus, wie sie auf verschiedene Gefahrensituationen reagieren.

c) Schreiben Sie einen sinnvollen JUnit-Test, der ihr Programm ausreichend testet. Überlegen Sie sich, wie Sie die den zufälligen Teil Ihres Programms testen können.