Wir implementieren ein Fahrrad. Dabei geht es darum welche Bestandteile des Fahrrades bei Fahren Schaden nehmen und wie das unsere Fahrt behindert.

Part

Die abstrakte Klasse Part hat

- Ein float Attribut durability, das seine Haltbarkeit angibt
- Einen Konstruktor der einen float erhält und die durability damit initialisiert
- Eine rein virtuelle Funktion ridingDamage, die zwei Integer erhält (Tempo und Gewicht) und nichts zurückgibt. Diese Funktion wird für jeden gefahrenen Kilometer aufgerufen.

Chain (Kette)

Die Klasse Chain erbt von Part und hat

- Einen Konstruktor der einen float erhält und den Konstruktor der Klasse Part damit aufruft
- Eine Implementierung der Funktion ridingDamage. Dabei wird die durability um 1% des Tempos reduziert. Sollte dabei die durability <= 0 werden, wird eine out_of_range exception geworfen.

Tube (Schlauch)

Die Klasse Tube erbt von Part und hat

- Einen Konstruktor der einen float erhält und den Konstruktor der Klasse Part damit aufruft
- Eine Implementierung der Funktion ridingDamage. Dabei wird die durability um speed * weight durch 3000 reduziert. Sollte dabei die durability auf oder unter 0 fallen, wird ein domain_error geworfen.

Spoke (Speiche)

Die Klasse Spoke erbt von Part und hat

- Einen Konstruktor der einen float erhält und den Konstruktor der Klasse Part damit aufruft
- Eine Implementierung der Funktion ridingDamage. Dabei wird die durability um speed durch 500 reduziert. Außerdem gibt es eine Wahrscheinlichkeit (rand() % 5000 < weight), dass die durability um weitere 10 fällt. Sollte dabei die durability auf oder unter 0 fallen, wird ein runtime_error geworfen.

Bicycle (Fahrrad)

Die Klasse Bicycle hat

- Ein Attribut parts. Parts speichert mehrere Part-Objekte, unabhängig ob es Speichen, Kette oder Schlauch sind.
- Ein Attribut brokenSpokes, das zählt wieviele Speichen bereits kaputt sind
- Einen Konstruktor, der die beiden Attribute initialisiert. Dabei werden alle Parts mit einer durability von 100 am Heap angelegt. Es gibt eine Kette, 2 Schläuche und 40 Speichen.
- Eine Funktion ride, die eine Distanz, ein Tempo und ein Gewicht (als Ganzzahlen) erhält und dann entsprechend viele Kilometer mit dem übergebenen Tempo und Gewicht ausführt. Dabei wird für jedes Bauteil des Fahrrads entsprechend die Funktion ridingDamage aufgerufen. Die Funktion retourniert die tatsächlich gefahrenen Kilometer.
- Dann soll die ride-Funktion um einen try-catch Block erweitert werden, der auf die entsprechenden exceptions reagiert:
    - Chain – es werden 50 geplante Kilometer „übersprungen“ also nicht gefahren. Die Kette wird durch eine neue Kette ersetzt. Dabei soll natürlich ein Speicherleck vermieden werden.
    - Tube – es werden 10 geplante Kilometer nicht gefahren. Der Schlauch wird durch einen neuen ersetzt. Dabei soll natürlich ein Speicherleck vermieden werden.
    - Spoke – die Speiche wird gelöscht und in Zukunft wird ihre ridingDamage-Funktion nicht mehr aufgerufen. Der Zähler brokenSpokes wird inkrementiert. Wenn dies die sechste gebrochene Speiche ist, wird die Fahrt sofort beendet.

Achtet auf Speicherverwaltung!