Capycity

Szenario

Capybaras sind unglaubliche Ingenieure, erst kürzlich haben sie einen Durchbruch bei der umweltfreundlichen Energie erlangt. Sie beschließen daher eine extra Stadt namens Capycity aufzubauen, welche ihren Fokus auf die Erzeugung von Energie legt und somit umgebende Städte vollständig versorgen kann.

Aktuell sind sie in der Lage mithilfe folgender Technologien Energie zu erzeugen:

- Wasserkraftwerk
- Windkraftwerk
- Solarpanele

Kapitel 1 - Der Aufbau

Die Landfläche

Um Capycity zu bauen, benötigt es zunächst bebaubarer Landfläche. Die Ingenieure wollen nicht ohne ordentlich Planung anfangen. Ein Capybara, auf dessen T-Shirt steht "!false - it's funny because it's true", blickt in den Raum und schlägt vor ein erstes Simulationstool zu entwickeln.

"Wir benötigen ein Tool, welchem wir eine Fläche der Form LxB (Länge mal Breite) mitgeben können, die dann virtuell im Speicher erstellt wird. Danach soll es uns die Möglichkeit geben verschiedene Gebäude an verschiedenen Plätzen zu platzieren. Um Arbeit zu sparen, sollte das Tool gleich nach der Größe, auch wieder in der Form LxB sowie der Position des Gebäudes fragen. Sollte es eine Kollision zwischen zwei Gebäuden geben, dann soll natürlich ein Fehler ausgegeben werden. Es muss daher für Korrekturen die Möglichkeit geben Bereiche wieder als bebaubar zu deklarieren. Und damit wir den Plan ordentlich betrachten können, muss dieser natürlich ausdruckbar sein"

Die anderen Capybaras stimmen nickend zu und machen sich gleich an die Arbeit.

Aufgaben

Schreibe ein erstes Simulationstool (Kommandozeilentool), welches folgende Features besitzt: \

- [ ] Erhalt der Länge und Breite des Baubereichs als Argument über die Kommandozeile.
- [ ] Erstellung eines Arrays, welches den Baubereich repräsentiert. Elemente sollen dabei ein Enum sein, welcher den Gebäudetypen darstellt (bzw. ein Feld als leer darstellt, wenn sich auf einer Position kein Gebäude befindet)
- [ ] Anzeige eines Menüs mit folgenden Einträgen und dahinterstehender Funktionalität
  - [ ] Gebäude setzen (mit darauffolgender Nachfrage nach Art, Länge, Breite und Position)
  - [ ] Bereich löschen (Betroffene Gebäude sollen nicht gelöscht, sondern dadurch nur verkleinert werden)
  - [ ] Ausgeben des aktuellen Bauplans
  - [ ] Beenden des Programms

- [ ] Prüfung ob Teile eines zu bauenden Gebäudes mit anderen Gebäuden kollidiert oder außerhalb des Baubereichs liegt.

Kapitel 2 - Das Review

Die Anpassungen

Der Capybara mit dem nerdigen T-Shirt, Bob ist sein Name, ruft alle andere Capybaras zusammen und lässt sich den Zwischenstand zeigen. Er spielt mit dem Tool ein wenig herum und schaut sich auch den Code an. Kurz darauf nickt er zufrieden. "Ich denke, dass ist ausreichend für die Planung".
Plötzlich tritt seine Kollegin Carla hervor und tippt mit ihren Finger auf einige Codestellen. "Ich denke, dass wir hier noch einiges machen können. Außerdem haben wir gar keinen Überblick über benötigte Materialien und resultierende Kosten. Erstmal sollte die Planung eine eigenständige Klasse sein. Dann wäre es auch gut die Gebäude als Klassen zu definieren, welche einen Grundpreis, ein Label und eine Auflistung benötigter Materialien enthält. Danach sollten wir das Ausdrucken des Plans erweitern, es sollten uns auch benötigte Materialien und eine Auflistung der Preis sowie der Gesamtpreis angezeigt werden."
Bob denkt kurz über Carlas Worte nach, blickt zu den anderen Capybaras und sagt: "Carla hat Recht, wir sollten noch die von ihr genannten Anpassungen und Erweiterungen machen, bisher ist alles doch noch etwas rudimentär"
Erneut stimmen die anderen Capybaras nickend zu und machen sich wieder an die Arbeit

Aufgaben

Mache bei dem Simulationstool folgende Anpassungen:

- [ ] Kapsel die Verwaltung der Gebäude innerhalb einer Klasse namens **CapycitySim**
- [ ] Erstelle eine Klasse für Materialien.
  - [ ] Es soll **Holz**, **Metall**, **Kunststoff** geben
  - [ ] Alle leiten sich von der Basisklasse **Material** ab
  - [ ] Jedes Material soll einen eigenen Preis besitzen
- [ ] Ersetze die Enum der Gebäude durch eigenständige Klassen für die einzelnen Gebäude.
  - [ ] Diese sollen sich von einer Basisklasse **Building** ableiten.
  - [ ] Jedes Gebäude soll einen **Grundpreis** besitzen
  - [ ] Jedes Gebäude soll ein **Label** besitzen, welches beim Ausdrucken des Plans angezeigt wird
  - [ ] Jedes Gebäude soll eine Liste in Form eines Arrays oder Vektors haben, welche benötigte Materialien enthält. Mehr Materialien einer Art benötigen mehr Plätze (z.B. 2xHolz -> [Holz, Holz] bei einem Array)
- [ ] Erweitere das Ausdrucken des Plans um die Darstellung der Gebäude mit ihren Labels, einer Auflistung der Gebäude sowie deren benötigter Materialien, dem Einzelpreis eines Gebäudes sowie dem Gesamtpreis von allen Gebäuden.