# Schlussprojekt Bitcoin Tracing

## 1. Auftrag

Eine Firma beauftragt Sie eine Sicherheits-Software zu Schreiben, um den Verlauf von gestohlenen Bitcoins zu verfolgen. Dabei soll das Programm auf dem LIFO-Prinzip die Bitcoins verfolgen. Dem Benutzer soll eine grafische Oberfläche angeboten werden, welche die Startadresse aufnimmt und den Verlauf des Geldes aufzeigt.

# 2. Informationen

## 1. Bitcoin

Bitcoin ist eine digitale Währung. Überweisungen werden von einem Zusammenschluss von Rechnern über das Internet mithilfe einer speziellen Peer-to-Peer-Anwendung abgewickelt, sodass anders als im herkömmlichen Bankverkehr keine zentrale Abwicklungsstelle benötigt wird. Eigentumsnachweise an Bitcoin können in einer persönlichen digitalen Brieftasche gespeichert werden.

Die Bitcoin-Daten werden in einer sogenannten Blockchain «gehalten». Diese beinhaltet alle Bitcoin-Transaktionen, die seit der Veröffentlichung von Bitcoin durchgeführt wurden. In der Blockchain befinden sich sogenannte **Adressen**, die entsprechende Bitcoins halten können. Diese Adressen gehören zu den genannten digitalen Brieftaschen (Wallets). Ein Beispiel einer Adresse sieht wie folgt aus:

**1JC41YHmjKEcW1rLH6pmMWEFHkoNwSmhnC**

Von dieser Adresse aus können **Transaktionen** durchgeführt werden, wodurch Bitcoins von anderen oder zu anderen Adressen gesendet werden. Nur der Besitzer einer Brieftasche/Adresse kann Bitcoins von seiner Adresse aus zu einer anderen Adresse senden.

Ein einfaches Gedankenmodell zur Beschreibung des Bitcoinssystems sieht dabei wie folgt aus:

- Sie und Ihre Freunde möchten einen Geldbetrag unter sich aufteilen. Um jedoch nicht sofort die Münzen zu verteilen, schreiben Sie sich alles zuerst auf ein Papier auf. Jeder dieser Gruppe kann auf das Papier sehen.
- Nun schreiben Sie auf das Papier, dass jeder einen Teil des Geldes bekommt. Dabei schreiben Sie als Information den Namen der Person auf. Der Name ist dabei die **Adresse** in der Blockchain.
- Ein Mitglied möchte jedoch nun einer Person einen Teil seines Geldes übergeben. Dazu schreiben Sie wieder eine Zeile auf das Papier, dass ein gewisser Betrag von einer Person zu der anderen Person übergeht. Diese Zeile stellt die **Transaktion** in der Blockchain dar.

Jede Transaktion erhält auch eine eindeutige Nummer, die wie folgt aussehen kann:

**a3992ae328c167d34f06493909de2113107585b41b4e1e4b30882307fddca517**

## 2. Tracing

Da in der Bitcoin-Blockchain nur frei generierte Adressen und keine identifizierbaren Informationen hinterlegt sind, wird dieses System sehr gerne für kriminelle Aktivitäten genutzt. Auch das direkte Stehlen von Bitcoins stellt ein Problem dar, da die kriminellen Aktivisten nur erschwert verfolgt werden können.

Die Bitcoin-Blockchain ist jedoch öffentlich zugänglich und alle Transaktionen können eingesehen werden. Dies erlaubt ein verfolgen und **markieren** der Transaktionen, welche einen kriminellen Ursprung haben. Ein einfaches Markieren aller Transaktionen ist jedoch nicht sinnvoll, da schon in kurzer Zeit die meisten Blockchain-Transaktionen als markiert angesehen werden.

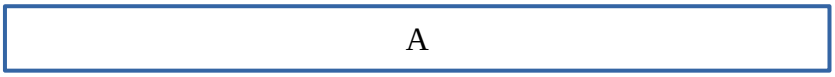
Eine weitere Variante, um die Bitcoins zu markieren, ist das «Last In First Out»-Prinzip. Dazu geht man davon aus, dass die Bitcoins, welche als letztes auf die Adressen geladen werden, auch die sind, welche bei einer ausgehenden Transaktion zuerst genutzt werden. Dadurch werden nur spezifische Bitcoins markiert und die Verfolgung ist dadurch überschaubar.

Eine weitere Erklärung zu diesem Thema wird bei folgendem Video angeboten:

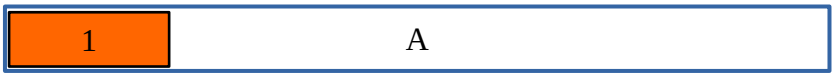
https://www.youtube.com/watch?v=UlLN0QERWBs

Eine visuelle Darstellung des LIFO-Prinzips sieht dabei wie folgt aus:

1. Betrachtung der Adresse *A*, welche keine Bitcoins momentan aufweist.

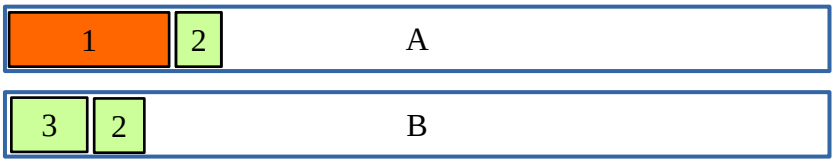



2. Adresse *A* erhält die gestohlenen Bitcoins. Diese werden entsprechend markiert.

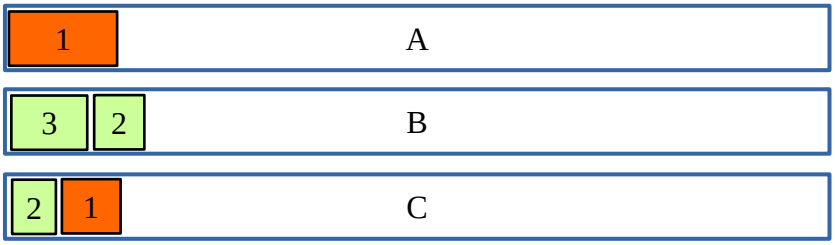



3. Adresse *A* erhält weitere (legale) Bitcoins.


4. Adresse *A* sendet Bitcoins an Adresse *B*.




5. Adresse *A* sendet weitere Bitcoins an Adresse *C*.



## 3. Web-API

Web-APIs (REST) sind weitverbreitete Schnittstellen, die es erlauben externe Strukturen zu lesen oder anzusteuern. In diesem Projekt können Sie über den Server insight.bitpay.com eine Web-API zur Betrachtung der Bitcoin-Blockchain nutzen.

https://insight.bitpay.com/

Die API ist dabei so aufgebaut, dass Sie Daten einer Adresse über folgende URL abfragen können:

https://insight.bitpay.com/api/addr/**[BITCOIN-ADRESSE]**

Daten über eine Transaktion können über folgende API-Adresse abgerufen werden:

https://insight.bitpay.com/api/tx/**[BITCOIN-TRANSAKTION]**

Beachten Sie, dass diese Seite bei zu vielen Zugriffen teilweise einen Fehler auswirft (**Error 429 Too Many Request**). Fangen Sie diesen Fehler ab und wiederholen Sie die Abfrage, bis eine erfolgreiche Rückmeldung gesendet wird.

Die Meisten API-Seiten geben ein kompaktes JSON-Format zurück, was die Betrachtung erschwert. Es empfiehlt sich, diese Daten mit dem Browser Firefox aufzurufen, da dieser die Daten formatiert und daher lesbarer ausgibt.

## 4. Webzugriff

Um den Inhalt einer Website zu lesen, können Sie die Klasse *java.net.URL* aus der Standard-Bibliothek nehmen. Mit folgendem Beispielcode erhalten Sie durch diese Klasse einen InputStreamReader von der API-Seite https://api.predic8.de/shop/:

```
String API_URL = "https://api.predic8.de/shop/";
URL url;
url = new URL(API_URL);
URLConnection request = url.openConnection();
request.connect();
InputStreamReader reader = new InputStreamReader((InputStream)
                           request.getContent());
```

## 5. Json-Format

Die JavaScript Object Notation, kurz **JSON**, ist ein kompaktes Datenformat in einer einfach Textform zum Zweck des Datenaustauschs zwischen Anwendungen. In Java können Sie das JSON-Format mit der externen Programmbibliothek *gson* in Java-Elemente umwandeln:

https://github.com/google/gson

Die JAR-Datei können Sie unter folgendem Link beziehen:

https://repo1.maven.org/maven2/com/google/code/gson/gson/2.8.2/gson-2.8.2.jar

Ein kurze Einführung in gson finden Sie hier:

https://futurestud.io/tutorials/gson-getting-started-with-java-json-serialization-deserialization

Um den *InputStreamReder* aus dem Webzugriff-Beispiel mit gson zu verwenden, kann folgendes Code-Beispiel betrachtet werden:

```
...
Gson gson = new Gson();
Elements elements = gson.fromJson(reader,Elements.class);

System.out.println(elements.description.swagger_ui);
```

Dabei wurde folgende Strukturklasse verwendet:

```
public class Elements{
    public Description description;
    public Links links;

    public static class Description{
        public String swagger;
        @SerializedName("swagger-ui")
        public String swagger_ui;
    }

    public static class Links{
        public String product_url;
        public String vendors_url;
        public String categories_url;
        public String customers_url;
        public String orders_url;
    }
}
```

# 3. Endprodukt

Das Programm soll über eine GUI-Schnittstelle eine Anfangsadresse aufnehmen. Zudem soll der Benutzer einen Wert bezüglich der Suchtiefe angeben. Dieser Wert gibt eine Begrenzung an, wie viele Transaktionsschritte in Serie beachtet werden sollen. Die Anzahl paralleler Transaktionen soll von diesem Wert nicht betroffen sein.

Beim Start des Programmes werden alle Bitcoins der Anfangsadresse markiert. Am Schluss soll der Benutzer den Verlauf aller Adressen aufgezeigt bekommen, wo markierte Bitcoins transferiert wurden. Alle Adressen, die nicht von markierten Bitcoins betroffen sind, müssen nicht beachtet und ausgegeben werden.

Sind alle Transaktionen bis zur angegeben «Tiefe» untersucht, wird das Programm beendet und der Verlauf dargestellt.

Die Darstellung des Verlaufes kann frei gewählt werden. Eine einfache textbasierte Lösung reicht hierbei schon aus.

# 4. Hinweise

1. Bei Bitcoin können ausgehende Adressen sich selber bei Transaktionen angeben (Sender und Empfänger). Diese Transaktionen können ignoriert werden.
2. Beachten Sie, dass die API nur bis zu maximal 1000 Transaktionen darstellen kann. Die Verfolgung der Bitcoins kann ab diesem Punkt abgebrochen und der aktuelle Stand ausgegeben werden.
3. Bei Transaktionen können mehrere Inputs sowie Outputs bestehen. Dies ist im folgendem Beispiel der Fall:


   Um einen Ablauf zu erhalten, können Sie die Transaktion als «Behälter» (wie eine Bitcoin-Adresse) betrachten. Die Verteilung geht dabei im JSON-Format «von oben nach unten». Im oberen Beispiel gehen die Bitcoins nach folgender Reihenfolge in den Behälter ein:

   1. 1PJbi...
   2. 12RDw…
   3. 1CMvv…

   Entsprechend werden die Empfänger in folgender Reihenfolge bedient:

   4. 1Pqvx…
   5. 39NSV…

   Die Aufteilung aus dem Behälter verläuft dabei wieder im LIFO-Prinzip. Das bedeutet, dass die Bitcoins von 1CMvv… als erstes an die Empfänger (zuerst 1Pqvx… und dann 39NSV…) verteilt werden.

4. Es kann vorkommen, dass bei Transaktionen weitere Transaktionen (Input oder Output) vorkommen. Um die Komplexität des Projektes klein zu halten, können diese Inputs bzw. Outputs ignoriert werden (Bitcoins gehen verloren).