Stichpunkte der Programmierung von Java's Crypt

- Grundaufbau → Implementation des Websockets
    - Welche Packages müssen eingebunden werden
    - Welche Annotationen und überlagerte Methoden
    - In welcher Form werden Nachrichten gesendet bzw. empfangen (→ Ausgaben implementieren)
- Javascript-Anbindung
    - Welche Methoden?
    - Wie sieht die URL aus?
    - Wie müssen Nachrichten versendet werden?
- Weitergehende Programmierung: Implementierung des Chats
    - HTML-Elemente einbauen (+ IDs)
    - Listener zum Senden des Textes einbauen
    - Wie werden Nachrichten verschickt (JSON bauen)
    - Wie werden Nachrichten wieder entschlüsselt in Java?
    - „Verteilung" der Nachricht an alle User in einer Session
        - Encoder und Decoder für Typ: Chat implementieren
        - Encoder und Decoder mit Websocket beim Start verbinden (→ Einbauen der „StartListener.java"-Klasse)
        - Aufbau der web.xml (zum Einbinden der StartListener-Klasse)
- Weitergehende Programmierung: Implementierung des Canvas
    - HTML-Element einfügen (+ID)
    - Listener hinzufügen
    - Nach gewähltem Typ (Auswahl folgt per Klick auf eines der Rechtecke oberhalb des Canvas) unterscheiden
    - Methoden derart einfach aufbauen, dass sie allgemein zu verwenden sind und in jedem Falle (sowohl beim Zeichnen als auch beim Empfangen von Zeichnungen) Verwendung finden
        - Lösung stellt hier das Arbeiten mit JSON-Dateien dar
        - Dadurch ist es egal, wer die Datei versendet bzw. empfängt
    - JSON-Datei mit allen nötigen Punkten aufbauen und zum Websocket schicken
    - Websocket entschlüsselt die JSON-Datei und je nach Typ wird ein jeweiliges Objekt erzeugt
    - Versand der Nachricht an alle User in der Session erfolgt wieder über Encoder und Decoder (erneutes Zusammenbauen der JSON-Datei)
- Weitergehende Implementierung: Aufbau einer Historie und Speichern von Objekten
    - Nach Empfang einer Nachricht (kein Chat) speichern des neuerzeugten Objekts in der History (HashMap-Objekt)
    - Sobald die „onOpen"-Methode aufgerufen wird (jemand neues loggt sich ein) → Versand aller in der History befindlichen Objekte
    - Um das Problem eines doppelten Eintrags (im Falle man hat selbst ein Objekt gezeichnet) zu umgehen, wurden Hilfsmethoden eingebaut, welche dies regeln
    - Problem des Löschens bei allen Objekten
        - „einfache" Nachricht, dass alle Objekte aus der History gelöscht werden, ist nicht möglich, da beim „Auslöser" dieses bereits weg war
        - Lösung: Hilfsmethode, wodurch Websocket Nachricht erhalten hat, dass die History gelöscht wird
        - Dadurch separates Löschen bei allen Usern und des Websockets

- Weitergehende Programmierung: Speichern des gezeichneten Bilds und der History als JSON-Datei
    - Implementierung des HTML-Elements Typ: <a>
    - Besitzt das Attribut „Download“, wodurch die angegebenen Objekte heruntergeladen werden
    - Problem, dass die History nicht bei allen Usern richtig heruntergeladen wird
        - Lösung: manuelles Erstellen eines Strings, das alle Objekte der History enthält → Herunterladen dieses Strings