

VOP F17 Obligatorisk Aflevering 1

Udleveres under lektion 3, 17. feb. 2017.

Afleveres via Assignment på BlackBoard senest under lektion 5, 10. marts 2017.

Dette opgavesæt indeholder 3 ud af 4 programmeringsopgaver, som blev stillet til den ordinære eksamen i VOP, juni 2016:

- Opgave 1 er opsætning af javaFx projektet. (Opgave 1 vægter 5 %)
- Opgave 2a, b, c løses uafhængigt af brugerfladen og testet med egne main()-metoder.
I opgave 2d integreres med javaFx. (Opgave 2 vægter 25 %)
- Opgave 3a løses uafhængigt af brugerfladen og testet med egen main()-metode.
I opgave 3b og 3c integreres med javaFx. (Opgave 3 vægter 20 %)

Vigtigt: Der kan være problemer med at eksekvere klasser med en main()-metode under et JavaFx projekt. Brug *Shift-F11* eller højre klik på projektet og vælg *Clean and Build*, inden *run* kaldes direkte på en Java main klasse.

Hint: Giv dig tid til at gennemlæse hele opgavesættet, inden du går i gang med løsningerne. Du bør dog starte med at opsætte projektet, som beskrevet i Opgave 1.

Aflevering: Løsningen afleveres på BlackBoard:

- Det er tilladt at samarbejde 2-3 studerende om opgaven; men hver studerende skal aflevere sit eget projekt.
- Hvis flere studerende har arbejdet sammen om væsentlige dele af løsningen, skal brugernavnene på samarbejdspartnerne fremgå af en kommentar til afleveringen på BB.
- Det endelige javaFx projekt skal oprettes med den studerendes SDU-brugernavn, fx *abcde15_VOP_opg1*. Det er altså ikke nok at den afleverede zip-fil hedder sådan, selve projektet skal have dette navn.
- I NetBeans markeres projektet, som indeholder løsningen
- Vælg menuen *File -> Export Projekt -> To ZIP...*
- På den fremkomne pop-up dialog, vælges et passende sted at gemme projektet i feltet *Build ZIP:*
- Sørg for at filen får det rigtige navn. Fx "abcd15 VOP Obl 1.zip"
- Upload filen

God fornøjelse.

Opg.1: Oprettelse af javaFx projekt

(5%)

Formål: At oprette eksamensprojektet som en JavaFX applikation, navngivet korrekt og forberedt til de øvrige eksamensopgaver.

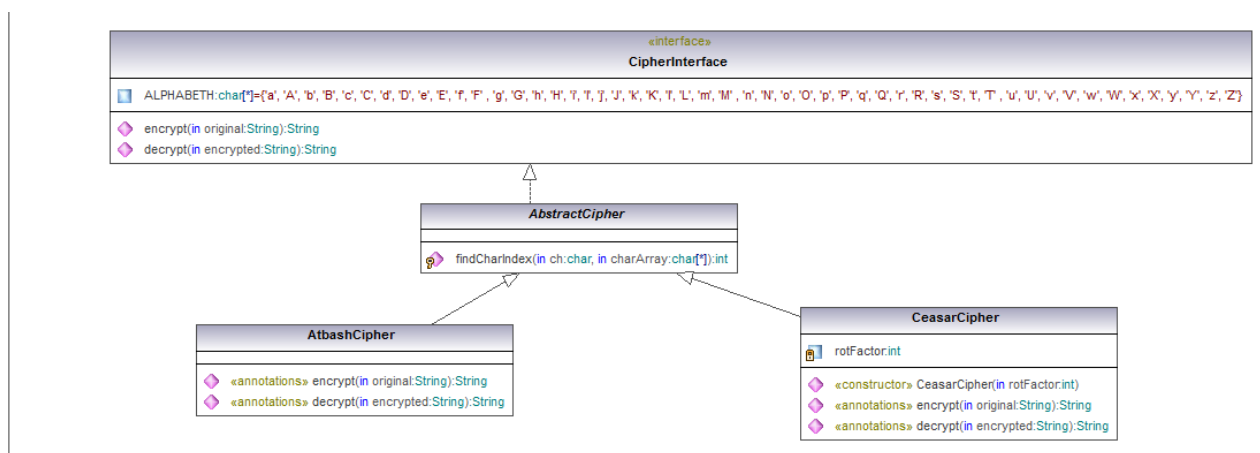
1. Opret et projekt i NetBeans af typen *JavaFX FXML Application*:
 - a. *Project Name* skal begynde med jeres SDU-brugernavn, fx "abcd15 VOP Obl 1".
 - b. Marker checkboksen *Create Application Main Class*.
2. Start *SceneBuilder*en ved at dobbeltklikke på den dannede *FXML*-fil og slet den *Button* og *Label* som er dannet til *Hello World* eksemplet.
3. Det dannede *Anchorpane* kan med fordel gøres større, fx ved at sætte *Pref Width = 640* og *Pref Height = 400* under *Layout*, eller ved at trække i et af hjørnerne med musen.
4. Sæt et *TabPane* på brugerfladen og tilpas dets størrelse til *Fit to Parent*
5. Pak den udleverede zip-fil ud og kopier mappen *ancient_encryption* til *src*-mappen.
6. Kopier de udleverede billedfiler til projektets default-mappe (udenfor *src*-mappen).

Opg. 2: Implementering af hierarki til oldtids-kryptering af tekster

Baggrund: Behovet for at sende meddelelser, som kun kan læses af den modtager meddelelsen er stilet til, har eksisteret siden oldtiden. I denne opgave skal der programmeres to sådanne historiske krypteringsalgoritmer (på engelsk *ciphers*):

- *AtbashCipher*: Stammer fra ca. år 500 f.kr, hvor det blev benyttet med det hebraiske alfabet. Ideen er at udskifte 1. tegn i alfabetet med det sidste, 2. tegn med det næstsidste, 3. tegn med det 3.-sidste osv.
- *CesarCipher*: Blev opfundet under Julius Cæsar (100 f.kr – 44 f.Kr.). Fremgangsmåden er at alle bogstaver i den originale meddelelse udskiftes med et bogstav, som ligger et givet antal pladser (nøglen) længere nede i alfabetet modulus alfabetets længde.

Klassediagram:



Udleveret kode:

- `CipherInterface.java`: Konstanten `ALPHABETH` er et array af `char`, indeholdende både små og store bogstaver fra det engelske alfabet. Arrayet skal opfattes som værende *cirkulært*, dvs. efter sekvensen `... , 'y', 'Y', 'z', 'Z'` følger `'a', 'A', 'b', 'B', ...`. Derudover erklæres metoder til *kryptering* og *dekryptering* af en `String`.
- `CipherDriver.java`: Driverklasse til test uden brugerflade (opgave a, b og c)

De 3 øvrige klasser fra diagrammet, skal implementeres ud fra disse krav:

Opg. 2a AbstractCipher:

(5 %)

Opret en *abstrakt klasse*, som implementerer CipherInterface. Klassen skal *ikke* implementere de to metoder fra interfaceet, men indeholde en metode, som de konkrete klasser kan benytte til at finde hvilket index et givet bogstav ligger på i ALPHABETH. Metoden skal have signaturen

```
<synlighed> int findCharIndex(char ch){...}
```

, hvor *<synlighed>* skal sikre at metoden kun kan kaldes fra sub-klasser til AbstractCipher.

Metoden skal returnere det index, som ch findes på i arrayet, eller -1, hvis ch ikke er et bogstav i alfabetet.

Opg. 2b AtbashCipher:

(6 %)

Denne klasse skal arve AbstractCipher. Implementer de to metoder fra interfaceet, så:

- `public String encrypt(String message)` returnerer message krypteret, således at alle *alfabetiske* tegn i message er skiftet ud ('a' -> 'Z', 'A' -> 'z', 'b' -> 'Y', osv.). *Ikke-alfabetiske* tegn, skal ikke ændres.
- `public String decrypt(String encrypted)` genskaber den originale message.
Hint: det kan gøres utroligt simpelt med Atbash algoritmen.

Opg. 2c CaesarCipher:

(8%)

Skal også arve AbstractCipher.

- Klassen skal have en heltals-variabel (`int rotFactor`), indeholdende det antal pladser i ALPHABETH, som et givet bogstav skal roteres. Variablen skal tildeles sin værdi via en parameter til en *constructor* og det skal sikres, at `rotFactor` ikke bliver negativ og ikke er større end antallet af tegn i alfabetet.
- `public String encrypt(String message)` returnerer message krypteret, således at alle *alfabetiske* tegn i message er skiftet ud med det tegn, som findes `rotFactor` længere fremme i alfabetet. Husk at opfatte alfabetet som cirkulært og at *Ikke-alfabetiske* tegn, ikke skal ændres.
- `public String decrypt(String encrypted)` skal genskabe den originale message.
Der skal altså roteres den anden vej.

Hvis den udleverede driverklasse, CipherDriver, eksekveres, skal resultatet svare til dette:

```
run-single:
Original:
Her har vi en Meddelelse, som er hemmelig!
Atbash:
sVI SZI ER VM nVWWVOVOHV, HLN VI SVNNVORT!
Her har vi en Meddelelse, som er hemmelig!
Caesar 13:
oKX NGX BO KT tKJJKRKRYK, YUS KX NKSSKROM!
Her har vi en Meddelelse, som er hemmelig!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Opg. 2d Integrering i javaFX brugerfladen

(6 %)

Definer en brugerflade, som indeholder:
Label og TextField til beskeden.

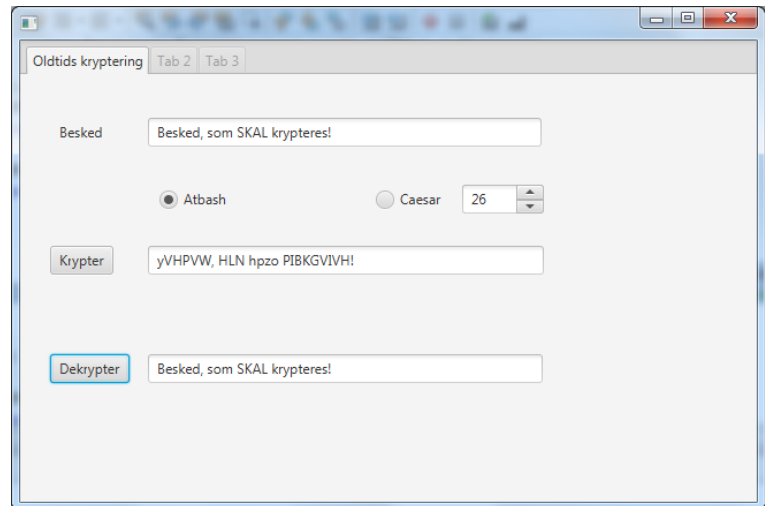
RadioButtons til valg af
krypteringsmetode, og en Spinner*) til
valg af rotFactor for CaesarCipher.

Button til kald af krypteringen og Label
eller TextField til krypteret besked.

Button til kald af dekryptering og Label
eller TextField til dekrypteret besked.

*)Hint: Spinneren kan konfigureres i
controllerens initialize()-metode med

```
ceasarSpinner.setValueFactory(new SpinnerValueFactory.IntegerSpinnerValueFactory(  
    0, CipherInterface.ALPHABETH.length - 1,  
    CipherInterface.ALPHABETH.length / 2));
```






Alternativt kan et lille TextField benyttes. I så fald skal input konverteres til en int.

Der skal være en fælles actionHandler til de to Buttons, som først aflæser hvilken af de to RadioButtons, der er valgt, initialiserer en lokal variabel af typen CipherInterface til enten AtbashCipher eller CaesarCipher (med værdien fra spinneren) og derefter kalder enten encrypt() med beskeden i beskedfeltet, eller decrypt() med den krypterede tekst fra feltet med krypteret besked.

Opg. 3: Sten – Saks – Papir

Baggrund:

Det gamle, kendte spil med håndtegn, Sten – Saks – Papir skal implementeres, så en bruger kan spille mod computeren via tastaturet (3a) og javaFx brugerfladen (3b og 3c). For god orden skyld gives her reglerne:

	Sten vinder over Saks (en saks bliver ubrugelig af nærkontakt med en sten)
	Saks vinder over Papir (en saks kan klippe papiret i små stykker)
	Papir vinder over Sten (stenen kan pakkes ind i papiret)

Opg. 3a RockScissorsPaper

(10 %)

Opret en package rock_scissors_paper, indeholdende en java main-klasse ved navn RockScissorsPaper.java. Klassen skal indeholde:

- et konstant array med de 3 mulige håndtegn som tekster:

```
public final static String[] HANDS = new String[]{"Sten", "Saks", "Papir"};
```


Alternativt kan der defineres en enum med de 3 værdier.

- to private variable af typen `String`, til håndtegnet for hhv. spilleren og computeren.
- `get()`-metoder på de to variable.
- en metode `public void play(String playerHand)` så:
 - computerens håndtegn vælges tilfældigt i `HANDS`.
 - Spillerens håndtegn tildeles parameterens værdi.
- en metode `public String getWinner()`, som returnerer resultatet, fx: "Du vinder!", "Computeren vinder!" eller "Uafgjort!", på baggrund af de håndtegn, som sidst er registreret.
- til test fra keyboard, implementeres `public static void main(String[] arg)` så:
 - der initialiseres en instance af `RockScissorsPaper`.
 - i en løkke eksekveres (afslut løkken, hvis ikke input er 0, 1 eller 2):
 - Brugeren promptes for input. Fx "[0]Sten, [1]Saks or [2]Papir ?"
 - Brug en `Scanner` til at læse input, og find spillerens håndtegn i `HANDS`.
 - Udskriv brugerens og computerens håndtegn, samt vinderen.

Eksempel på udskrift fra NetBeans:

```
run:
[0]Sten, [1]Saks or [2]Papir ?
2
Dig: Papir Computer: Sten
Og vinderen er: Du vinder!!
[0]Sten, [1]Saks or [2]Papir ?
1
Dig: Saks Computer: Saks
Og vinderen er: Uafgjort!!
[0]Sten, [1]Saks or [2]Papir ?
3
Illegal input! Game over!
BUILD SUCCESSFUL (total time: 5 seconds)
```

Opg. 3b Simpel integration med javafx

(5 %)

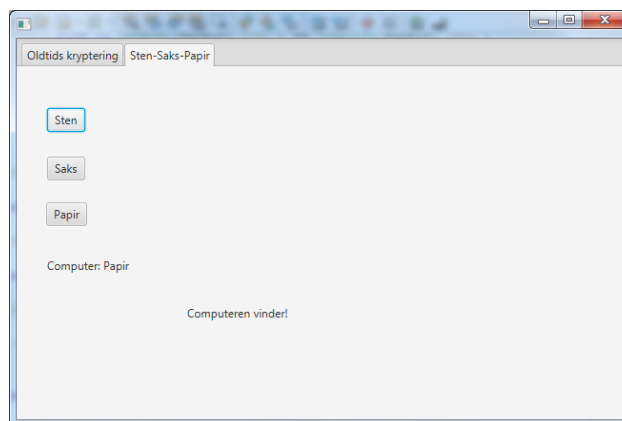
I denne opgave skal `RockScissorsPaper`-klassen fra forrige opgave, benyttes fra javafx-brugerfladen.

Definer følgende komponenter:

3 Buttons til valg af Sten/Saks/Papir

Label til computerens håndtegn

Label til udskrivning af vinderen



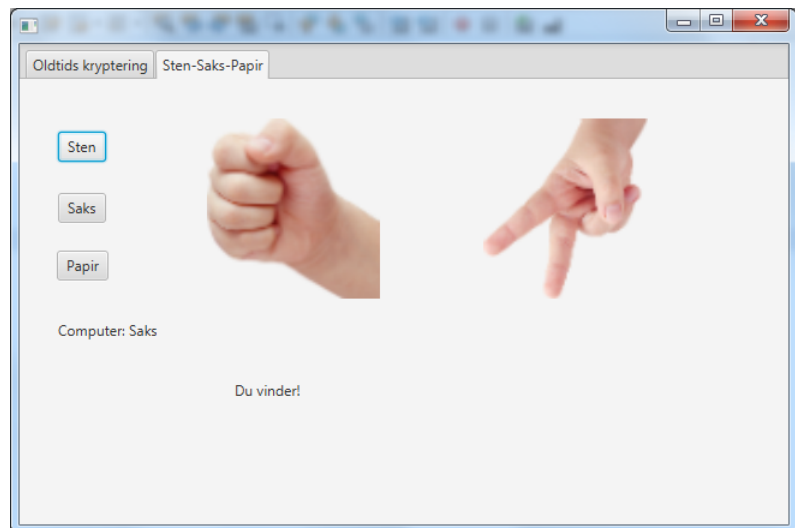
Initialiser en variabel af typen `RockScissorsPaper` i controllerklassens `initialize()`-metode.

Sæt en fælles `actionHandler` på de 3 buttons. Benyt `HANDS` og metoderne i `RockScissorsPaper` til at afvikle spillet med, ved klik på en af de tre knapper.

Opg. 3c: Visning af billeder af håndtegn

(5 %)

Tilføj to ImageView til brugerfladen, til visning af de kastede håndtegn:



Hint: Da dette ikke er gennemgået i detaljer, gives her et par gode råd:

- Opret en variabel `private Map<String, Image> picMap = new HashMap<>();`
- Hvis billederne ligger i projektets *default*-mappe, kan de indsættes med fx

```
picMap.put(RockScissorPaper.HANDS[0],  
           new Image(new File("Rock.png").toURI().toString()));
```

- I `actionHandleren` kan et billede dynamisk sættes på et `ImageView` med fx

```
pcChoise.setImage(picMap.get(rockScissorPaper.getComputer()));  
hvor pcChoise er ImageView'et for computerens håndtegn.
```