VOP F14 Re-Eksamen 20. august 2014

# Eksamensopgaver

Dette opgavesæt indeholder 4 programmeringsopgaver.

Opgaverne er vægtet på følgende måde:

1. 10 %  
   Her oprettes et javaFX projekt, indeholdende en brugerfalde, som (kan) benyttes i de følgende 3 opgaver.
2. 25 %  
   Der er 3 dele i opgaven, som bidrager med hhv. 10 %, 10 % og 5 %\*
3. 35 %  
   Her er der 5 dele, som bidrager med hhv. 7 %, 10 %, 3 %, 10 % og 5 %\*
4. 30 %  
   3 dele, som bidrager med 15 %, 10 % og 5 %\*

\*) Opgaverne 2, 3 og 4 kan løses og testes uafhængigt af javaFX brugerfladen, vha. main()-metoder.  
De sidste 5 % i disse opgaver gives for at integrere funktionaliteten med javaFX brugerfladen fra opgave 1. Dvs. javaFX delen af det samlede opgavesæt udgør i alt 25 %.

**Hint 1:** 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 Opg. 1.

**Hint 2:** Hvis der er problemer med at få eksekvere javaFX brugerfladen, kan det hjælpe at højreklikke på den genererede main()-fil og vælge *run*. Hvis ikke det hjælper må underviseren tilkaldes.

**Hint 3:** 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.

**Aflevering:** Ved afslutningen af eksamen skal løsningen afleveres på BlackBoard:

* 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 ”abcd13 VOPreEksamen.zip”
* Upload filen

God fornøjelse.

# Opg 1. Forberedelse af JavaFx projekt 10 %

1. Opret et projekt i NetBeans af typen *JavaFX FXML Application*:
   1. *Project Name* skal begynde med jeres SDU-brugernavn, fx ”abcd13 VOPreEksamen”.
   2. Marker checkboksen *Create Application Main Class.*
2. Start *SceneBuilderen* ved at dobbeltklikke på den dannede *FXML*-fil og slet den *Button* og *Label* som er dannet til *Hello World* eksemplet.
3. Sæt et *TabPane* på brugerfladen og tilpas dets størrelse til *Fit to Parent.*
4. Sæt 3 Tabs på med titlerne ”*Poly og Strings”*, *”Array Test”* og *”Urban Population”.*
5. Pak den udleverede zip-fil ud og kopier de 3 mapper til src-mappen.
6. Kopier *ByBefolkning.txt* til projektmappen (udenfor src-mappen).

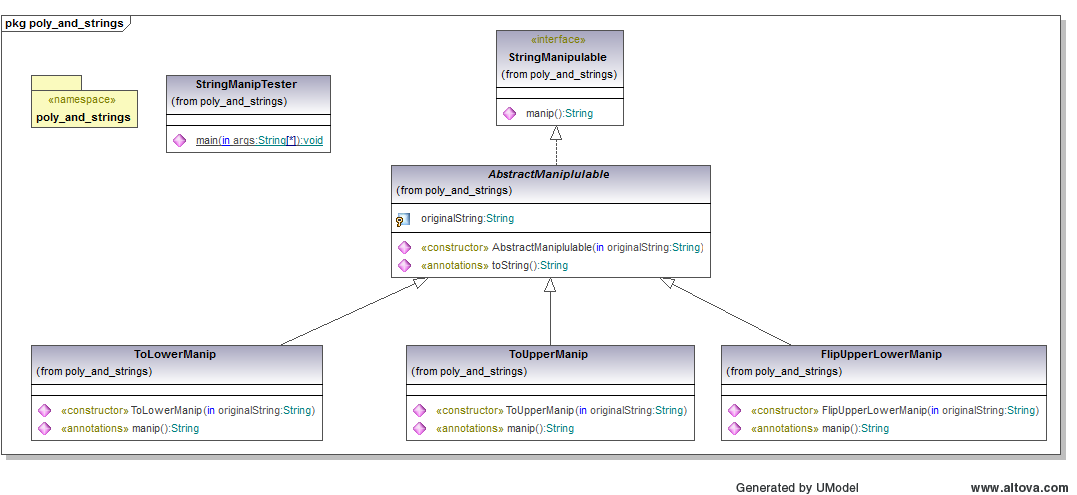
**Opg 2. Polymorphism og operationer på String 25 %**

*Udleveret: StringManipualble.java*

*AbstractManipulable.java*

*StringManipTester.java*

Der skal implementeres tre klasser ud fra dette klassediagram:



1. **Manipulation af hele strenge 10%**

De to klasses, ToLowerManip.java og ToUpperManip.java, skal arve AbstractManipulable og indeholde en constructor, som tager en String som input. Denne String skal sættes som værdi i variablen originalString via et kald til constructoren i AbstractManipulable.

Metoden public String manip() skal returnere strengen som hhv. udelukkende små og udelukkende store bogstaver.

1. **Manipulation af enkelte karakterer 10%**

Klassen FlipUpperLowerManip.java, skal også arve AbstractManipulable og benytter constructoren herfra.

I metoden public String manip(), skal strengen gennemløbes så alle små bogstaver ændres til store og alle store bogstaver ændres til små. Øvrige tegn forbliver uændret.

Test med StringManipTester.java bør give et resultat der ligner dette:

poly\_and\_strings.ToLowerManip:

Test AF PolyMORPHISM! -> test af polymorphism!

poly\_and\_strings.ToUpperManip:

Test AF PolyMORPHISM! -> TEST AF POLYMORPHISM!

poly\_and\_strings.FlipUpperLowerManip:

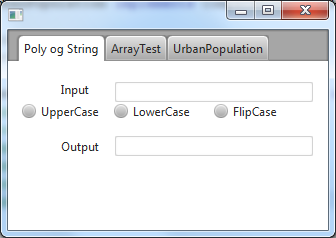
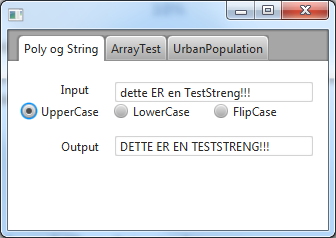
Test AF PolyMORPHISM! -> tEST af pOLYmorphism!

1. **Integrering med javaFX 5%**

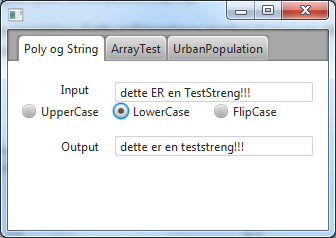
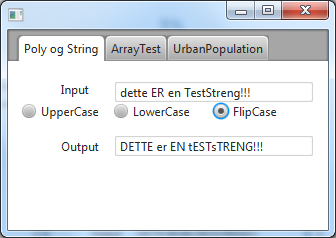
Fanebladet *Poly og String* skal indeholde en Label og et TextField til skrivning af input-strengen, tre RadioButtons til valg af Upper-, Lower- eller FlipCase (husk at tilknytte dem til en ToggleGroup), samt en Label og et TextField til den manipulerede output-streng. Der skal tilknyttes en ActionHandler (i FxController klassen) til de tre knapper. Benyt polymorphi på følgende måde i handleren:

* Opret en lokal variabel af interface typen StringManipulable.
* Undersøg hvilken radioknap der er klikket på og instantier variablen til en instance af den klasse der skal benyttes
* skriv resultatet af kald til manip() i output vinduet.

Herunder ses et muligt resultat i de fire situationer:

*Oprettelse:* *UpperCase:*

*LowerCase*: *FlipCase:*



**Opg 3. Operationer på arrays 35%**

*Udleveret kodeskelet:*   
ArrayTester.java: Klassen indeholder en main()-metode med udkommenterede test-kald. Fjern udkommenteringerne efterhånden som metoderne implementeres. Desuden er der en udkommenteret swap()*-*metode, som kan benyttes til implementering af opgave d.

I denne opgave skal der benyttes forskellige udgaver af løkker og betingelser til manipulation af et int-array.

1. **Oprettelse af klasse og implementering af constructor 7%**

Der skal erklæres følgende private felter:

* En statisk variabel, generator = new Random(), af typen java.util.Random.
* En konstant, MAX = 100, af typen int.
* En instance variabel, intArray, af typen int[].

Erklær constructoren så den tager en parameter, size, af typen int. I constructoren skal intArray initialiseres, så intArray får længden size. Benyt herefter generatoren til at udfylde intArray med tilfældige tal i intervallet [0, MAX[.

1. **Implementering af toString()-metoden 10%**

Implementer metoden public String toString() (fra klassen java.lang.Object),

så der returneres en String indeholdende elementerne i intArray, efter disse regler:

* Elementerne skal adskilles af et *komma* efterfulgt af et *mellemrum* (”, ”)
* Der skal indsættes et linjeskifte ("\n") som første tegn og derefter for hvert 8. element
* værdier, som er mindre end 10 skal indsættes med et foranstillet '0'
* Til sidst (udenfor loopet) skal der tilføjes "\n---------------------\n" eller lignende.

constructor:

45, 10, 21, 13, 34, 16, 63, 21

10, 76, 01, 45, 21, 80, 82, 89

74, 27, 25, 86, 50, 44, 53, 98

83, 19, 60, 27, 63, 33

---------------------

Når udkommenteringen fjernes for de første 3 linjer i main()-metoden skal resultatet ligne dette:

1. **Sortering 3%**

Implementer public void sort(), så sortering af arrayet sker med en af standard metoderne fra java.util.Arrays.

sorted:

01, 10, 10, 13, 16, 19, 21, 21

21, 25, 27, 27, 33, 34, 44, 45

45, 50, 53, 60, 63, 63, 74, 76

80, 82, 83, 86, 89, 98

---------------------

Test med main() metoden bør ligne dette:

1. **Implementering af reverse()-metode 10%**

Implementer metoden public void reverse(), så elementernes rækkefølge i intArray bliver omvendt. Når de sidste linjer i main() aktiveres, bør resultatet ligne:

reverse:

98, 89, 86, 83, 82, 80, 76, 74

63, 63, 60, 53, 50, 45, 45, 44

34, 33, 27, 27, 25, 21, 21, 21

19, 16, 13, 10, 10, 01

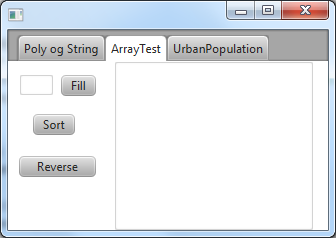
---------------------

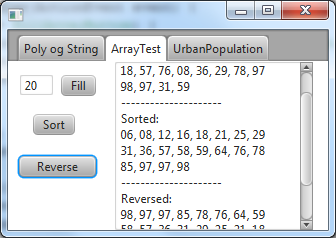
*Hint: Fjern udkommenteringen af swap()-metoden og benyt den i et loop med to indeks-variable.*

1. **Integrering med javaFx 5%**

Det er vigtigt at erklære et felt, arrayTester, af typen ArrayTester i FXLMControlleren.

Fanabladet *ArrayTest* skal indeholde følgende komponenter:

* TextArea til udskrift.
* TextField til input af ønskede størrelse på arrayet
* Fill-button. ActionHandleren på denne skal oprette en ny instance af ArrayTester, ved at kalde constructoren med værdien af inputfeltet konverteret til int. Herefter skal toString() kaldes og udskrives i TextArea’et
* Sort-button og Reverse-button:
  + Hvis ikke arrayTester er initialiseret, skal der skrives en fejl-meddelelse i TextArea’et
  + Ellers kaldes sort()eller reverse() efterfulgt af toString() og resultatet tilføjes (append’es) til TextArea’et

Efter Fill, Sort og Reverse fås dette resultat:

**Opg 4. I/O og datastrukturer 30 %**

*Udleveret:*

* *UrbanPopulation.java:* Kodeskelet til opg 4a
* *UrbanPopulationStatistics.java:* Kodeskelet til opg 4b
* *ByBefolkning.txt*: Oplysninger om hvor mange % af nogle udvalgte landes befolkning der bor i byer i årene 1980, 1990, 2000 og 2008.   
  Oplysningerne er sakset fra Udenrigsministeriets hjemmeside[[1]](#footnote-1), og lagt i tekstfilen med ét land pr. linje. De enkelte oplysninger er adskilt med ’/’ og rækkefølgen er:

*Land/1980/1990/2000/2008*.  
Fx viser linjen *Tanzania/15/19/22/26* at 15 % af befolkningen i Tanzania var byboere i 1980 og det var steget til 26 % i 2008. I denne opgave skal vi benytte oplysningerne om 1980 og 2008 for hvert land.

1. **Implementering af *UrbanPopulation* 10%**

UrbanPopulation skal implementeres så:

* Interfacet Comparable<E> implementeres
* Der skal være 3 felter:
  + Landets navn (String)
  + bybefolkning i % i 1980 (int)
  + bybefolkning i % i 2008 (int)
* Og 3 metoder:
  + private int getDiff(), som returnerer forskellen på de to bybefolknings-andele
  + public String toString(). Skal udskrive de 3 variable og forskellen på de to bybefolknings andele efterfulgt af ”\n”. (fx *Tanzania 1980: 15 2008: 26 Dif: 11*)
  + public int compareTo(UrbanPopulation o), fra Comparable. Instancer af UrbanPopulation skal kunne sorteres stigende efter differencen. Når 2 lande har samme difference skal der sorteres på landenes navne.

1. **Implementering af *UrbanPopulationStatistics* 15%**

UrbanPopulationStatistics skal implementeres så:

* Variablen private Set<UrbanPopulation> popSet i constructoren initialiseres til et sorteret Set, variablen file initialiseres med fileName og metoden private void readFile() kaldes.
* private void readFile() implementeres så:
  + Filen læses én linje ad gangen
  + Af oplysningerne om *landets navn, %-fordeling i 1980* og *%-fordeling i 2008* dannes instanser af UrbanPopulation, som indsættes i popSet.
  + Exceptions skal fanges inde i metoden og filen lukkes korrekt efter indlæsning.

Eksekvering af main()-metoden bør herefter give dette resultat:

UrbanPopulationStatistics:

Egypten 1980: 44 2008: 43 Dif: -1

Danmark 1980: 84 2008: 87 Dif: 3

Polen 1980: 58 2008: 61 Dif: 3

Nicaragua 1980: 50 2008: 57 Dif: 7

USA 1980: 74 2008: 82 Dif: 8

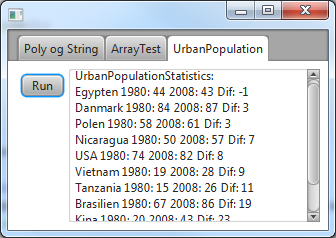
Vietnam 1980: 19 2008: 28 Dif: 9

Tanzania 1980: 15 2008: 26 Dif: 11

Brasilien 1980: 67 2008: 86 Dif: 19

Kina 1980: 20 2008: 43 Dif: 23

1. **Integrering med javaFx 5%**

Vi skal have en meget simpel brugerflade bestående af en Run-knap og et TextArea. Handleren på knappen skal kalde constructoren i UrbanPopulationStatistics og benytte toString()-metoden til at skrive resultatet på Tekstfeltet.

1. <http://udviklingstal.um.dk/da/tematiske-indgange/befolkning-levevilkaar-areal/befolkning/bybefolkning/> [↑](#footnote-ref-1)