

# Programmeringsuppgift 4

## Avsikt

Avsikten med programmeringsuppgiften är att du ska träna på att använda en array med objekt och samtidigt skriva ett program med grafiskt användargränssnitt.

Samtliga klasser ska placeras i paketet *p4*.

## Inlämning

Din lösning av uppgiften lämnas in via It's learning *senast kl 09.00 måndagen den 19/11*. Du ska placera *samtliga källkodsfiler i paketet p4* i en zip-fil.

Zip-filen ska du ge namnet AAABBB.zip där AAA är de tre första bokstäverna i ditt efternamn och BBB är de tre första bokstäverna i ditt förnamn. Använd endast tecknen a-z när du namnger filen.

- Om Rolf Axelsson ska lämna in sina lösningar ska filen heta AxeRol.zip.
- Om Örjan Märta ska lämna in sina lösningar ska filen heta MarOrj.zip.
- Är ditt förnamn eller efternamn kortare än tre bokstäver så ta med de bokstäver som är i namnet: Janet Ek lämnar in filen EkJan.zip

## Granskning

Ca *kl 13.00 den 19/11* kommer en kamrats lösning finnas i din inlämning på It's learning. Din uppgift är att granska kamratens lösning avseende:

- funktion – hur väl uppfyller lösningarna kraven? Är körresultatet korrekt? Kommunikerar klasserna korrekt?
- indentering mm, metodnamn, variabelnamn – hur välskriven är källkoden?
- kommentarer – är källkoden väl kommenterad?

Resultatet av din granskning ska du skriva ut och utskriften ska lämnas till labhandledaren efter redovisningen. Så här inleder du din granskning:

Granskning utförd av Einar Bok

Granskad lösning inlämnad av Eva Lind

Funktion/lösningar: (funktion i programmet som helhet, funktion i olika delar och sätt att lösa problem)

....

Indentering, metodnamn, variabelnamn mm: (hur ser det ut i klasserna som skrivits av Eva)

....

Kommentarer: (hur väl kommenterade är metoderna och klasserna)

....

## Redovisning

Redovisning sker *tisdagen den 20/11*. Redovisningstid publiceras på It's learning under *måndagen den 19/11*. Kom väl förberedd till redovisningen. Kom i god tid till redovisningen så du är beredd då det är din tur. Se till att du är inloggad på en dator (eller har egen dator), att eclipse är igång på datorn och att det går att exekvera dina lösningar.

En redovisning sker genom att:

- Studentens lösningar körs.
- Granskaren redogör för sina bedömningar
- Studenten svarar för sina lösningar
- Labhandledaren ställer kompletterande frågor
- De studenter i gruppen som inte redovisar är åhörare.

Godkänd uppgift signeras av läraren på lämpligt papper, t.ex. Redovisade uppgifter (se kurssidan). Du ska spara den signerade utskriften tills kursen är avslutad.

Om labhandledaren anser att det endast krävs *mindre komplettering för att lösningen ska godkännas* kan denna komplettering äga rum direkt efter redovisningen. Labhandledaren granskar kompletterad lösning då tiden medger.

Om labhandledaren anser att det endast krävs *mindre komplettering för att granskningen ska godkännas* kan denna komplettering äga rum direkt efter redovisningen. Labhandledaren granskar kompletterad granskning då tiden medger.

## Uppgift

Du ska skriva ett program som visar världens länder, ländernas befolkning och lite statistik om länderna som visas. Länderna, och deras befolkning, finns lagrad i en fil vilken du kommer att kunna läsa in och använda i ditt program.

Användaren ska kunna välja ett intervall för befolkningen varvid de länder som är i intervallet ska visas. Användaren ska också kunna ändra befolkningen i olika länder. Slutligen ska användaren kunna spara länder-informationen till hårddisken och kunna hämta sparad länder-information från hårddisken.

Till din hjälp har du

- Laboration 19 och 20 i vilka du jobbade med arrayer.
- Klassen **CountryIO** (DA211TP4HT12.zip) vilken ser till att det går att skriva länder-information på hårddisken och hämta länder-information från hårddisken.
- Klassen **CountryUserInput** (se DA211TP4HT12.zip) vilken innehåller designen av fönstret till höger.
- Klassen **Country** (se DA211TP4HT12.zip) vilken lagrar ett lands namn och befolkning.
- Klassen **CountryMain** vilken innehåller *main*-metod.

Din lösning kommer att bestå av minst sex klasser:

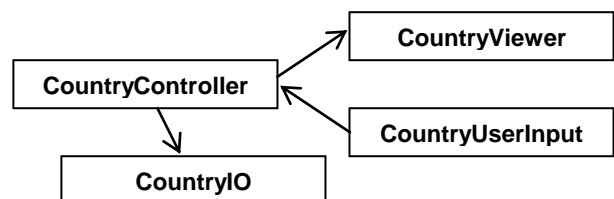
- Klasserna **CountryMain**, **CountryIO** och **Country**. Dessa är färdiga att använda.
- Klassen **CountryUserInput**. Du ska komplettera klassen med händelsehantering.
- Klasserna **CountryViewer** och **CountryController**. Dessa ska du skriva helt och hållet..

I klassen **CountryIO** finns en *main*-metod som visar hur klassen används, dvs hur man hämtar länder-information från hårddisken och hur man skriver länder-information till hårddisken

## Systembeskrivning

Klassdiagrammen till höger visar hur klasserna kommunicerar med varandra.

Nedan ser du förslag till vad klasserna som du ska skriva kan innehålla. Du är fri att utforma klasserna som du vill. Men flödet i programmet måste ungefär stämma med beskrivningen på nästa sida.



<b>CountryController</b>
-viewer : CountryViewer -inOut : CountryIO -countries : Country[]
+CountryController(CountryViewer, CountryIO)  +loadCountries() +saveCountries() +showAllCountries() +showSelection( long min, long max ) +changePopulation( String country, long population)

<b>CountryViewer</b> extends JPanel
+CountryViewer() +showCountries( Country[] ) +showStatistics( String )

<b>CountryUserInput</b> extends JPanel
-controller : CountryController
+CountryUserInput(CountryController) // metod/metoder för händelser eller // inre klass/klasser för händelser

Länder och deras befolkning	
Ungern	9981334
Tchad	9944201
Guinea	9690222
Serbien	9396411
Dominikanska republiken	9183984
Sverige	9122269
Bolivia	8989046
Somalia	8863338
Rwanda	8648248
Haiti	8308504
Österrike	8192880
Burundi	8090068

Statistik	
Antal länder: 12	
Total befolkning: 108410505	
Minsta befolkning: 8090068, Största befolkning: 9981334	

Urval

Minimal befolkning: 8000000    Maximal befolkning: 10000000    Visa alla länder  
Gör urval

Ändra befolkning

Land:    Befolkning:    Ändra

Hämta länder    Spara länder

## Programflöde

När användaren klickar på knappar i *CountryUserInput* så ska metoder i *CountryController* anropas. Sedan utförs nödvändiga instruktioner i *CountryController*-metoden och slutligen visas resultatet i *CountryViewer*-klassen. Förutom när programmet startas så utförs kod endast när användaren klickar på någon av knapparna. Nedanstående punkter beskriver vad som ska utföras vid programstart och vid klick på någon av knapparna.

- **Start** av program  
main-metoden är given i klassen *CountyMain*.

```
import javax.swing.*;


public class CountryMain {
    public static void showFrame( JPanel panel, int x, int y, String title, boolean
resizable ) {
        JFrame frame = new JFrame( title );
        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        frame.setLocation( x, y );
        frame.setResizable( resizable );
        frame.add( panel );
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater( new Runnable() {
            public void run() {
                CountryViewer viewer = new CountryViewer();
                CountryIO io = new CountryIO( "C:/filer/befolkning.txt" );
                CountryController controller = new CountryController( viewer, io );
                CountryUserInput user = new CountryUserInput( controller );

                CountryMain.showFrame( user, 50, 50, "CountryUserInput", false );
                CountryMain.showFrame( viewer, 500, 50, "CountryViewer", true );
            }
        });
    }
}
```

Den viktigaste delen är konstruktorn i *CountryController*-klassen. I den ska:

- parametrarnas värden lagras i instansvariabler
  - länder-data läses in. Detta sker genom anrop till metoden *readCountries()* i klassen *CountryIO*. Till varje land skapas ett *Country*-objekt och dessa lagras i *Country*-arrayen som returneras.
  - metod/metoder i *CountryViewer* anropas så att länder-datan och statistik visas.
- Klick på **Hämta länder**-knappen  
Anropa en metod i *CountryController*. Metoden ska se till att länder-data läses in från hårddisken. Sedan ska den inlästa datan och statistik visas i *CountryViewer*.
  - Klick på **Spara länder**-knappen  
Anropa en metod i *CountryController*. Metoden ska se till att länder-data sparas på hårddisken. Detta sker genom anrop av metoden *saveCountries( Country[] )* i *CountryIO*.
  - Klick på **Visa alla länder**-knappen  
Anropa en metod i *CountryController*. Metoden ska se till att alla länder och statistik om dem visas i *CountryViewer*. (se figur till höger)



Länder och deras befolkning	
Tonga	114009
Mikronesiens federerade stater	108004
Kiribati	105432
Grenada	89703
Seychellerna	81541
Andorra	71201
Antigua och Barbuda	69108
Dominica	68910
Marshallöarna	60422
Saint Christopher och Nevis	39139
Liechtenstein	33987
Monaco	32543
San Marino	29251
Palau	20579
Nauru	13287
Tuvalu	11810
Vatikanstaten	932

Antal länder: 194	
Total befolkning: 6506954533	
Minsta befolkning: 932, Största befolkning: 1313973713	

- Klick på **Gör urval**-knappen

Anropa en metod i *CountryController*. Metoden ska skapa en *Country*-array som innehåller alla länder med en befolkning i det intervall som användaren angett (inmatningsrutorna Minimal befolkning och Maximal befolkning i *CountryUserInput*). Sedan ska länder-datan och statistik om länder-datan visas i *CountryViewer*.

**CountryUserInput**

Urval

Minimal befolkning: 150000    Maximal befolkning: 250000    **Visa alla länder**    **Gör urval**

Ändra befolkning

Land:    Befolkning:    **Ändra**

**Hämta länder**    **Spara länder**

**CountryViewer**

**Länder och deras befolkning**

Vanuatu	208869
São Tomé och Príncipe	193413
Samoa	176908
Saint Lucia	168458

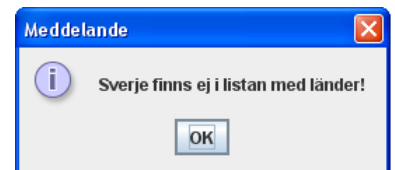
**Statistik**

Antal länder: 4  
Total befolkning: 747648  
Minsta befolkning: 168458, Största befolkning: 208869

- Klick på **Ändra**-knappen

Anropa en metod i *CountryController*. Metoden ska utföra följande:

- Söka efter positionen till ett *Country*-objekt med angivet landsnamn, dvs det landsnamn som användaren matat in under rubriken Land i *CountryUserInput*-fönstret.
  - Om ett sådant finns så
    - skapa ett nytt *Country*-objekt med angivet namn och ny befolkning.
    - lagra det nya objektet i den funna positionen i *Country*-arrayen.
    - visa alla länder i *CountryViewer*
- Annars (det sökta landet finns inte)
- visa en dialog med texten "LAND finns ej i listan med länder"



## Information om *CountryViewer*

Länder-datan ska visas i en *JTextArea*-komponent. Vissa inställningar behövs för *JTextArea*-komponenten:

- Ändra *font* till Monospaced, storlek 12.
- Anropa metoden *setEditable* med argumentet *false*.
- JTextArea*-komponenten ska placeras i en *JScrollPane*-komponent.

```
private JTextArea taCountry = new JTextArea();  
private JScrollPane scroll = new JScrollPane( taCountry );
```

*JScrollPane*-komponent ska i sin tur placeras i containern (t.ex. ärvd *JPanel*).

```
add( scroll );
```

Det är *JScrollPane*-komponenten som du ska ge önskvärd storlek:

```
scroll.setPreferredSize( ... );
```

För att fylla textarean med länder-data så ska du

- tömma *JTextArea*-komponenten på text. Använd *setText*-metoden med en tom sträng som argument.
- iterera genom hela *Country*-arrayen. För varje *Country*-objekt ska du anropa *append*-metoden

med *String*-argumentet: `countries[i].toString() + "\n"`. På så sätt skapas en lista med alla länder-objekt i *Country*-arrayen.

Om listan blir så lång så det behövs en scroll-bar till höger så lägger komponenten till denna automatiskt.

Statistiken ska visas i en *JTextArea*. Denna har en border av typen *TitledBorder*.

## Frivilliga utökningar

När användaren ändrar befolkningen i ett land så bör *Country*-arrayen sorteras med avseende på ländernas befolkning. Den ska sorteras avtagande.

Man kan låta användaren göra fler typer av urval, t.ex. första bokstav i namnet så att alla länder som börjar på en viss bokstav visas.

*Hämta länder* respektive *Spara länder* kan låta användaren välja fil med en fildialog. Titta i API-dokumentationen (<http://docs.oracle.com/javase/7/docs/api/index.html>) på klassen ***JFileChooser***. I klassen finns speciella metoder för att visa öppna-dialog respektive spara-dialog.