Laboration 20, Arrayer och objekt

Avsikten med laborationen är att du ska träna på att använda arrayer. Skapa paketet **laboration20** i ditt laborationsprojekt innan du fortsätter med laborationen. Hämta filen **befolkning.txt** från kurssidan och placera filen i katalogen M:\java.

Grundläggande uppgifter

Uppgift 20a

Klassen ***Exercise20a*** är given och din uppgift är att ersätta `// Komplettera med kod` med kod som utför det som beskrivs i deluppgifterna 1-10. När du löst och testat lösningen av en deluppgift så avmarkerar du den för att lösa nästa deluppgift.

I uppgiften används klasserna ***Population*** och ***Populations***. Du finner båda på kurssidan. Placera klasserna i paketet *laboration20* och testkör sedan *Populations.java*. I metoden läses innehållet i filen *befolkning.txt* in i arrayen countries (av typen Population[]). Varje Population-objekt innehåller namnet på ett land och antalet invånare.

Studera klassen *Population* så du vet vilka public-deklarerade metoder som finns i klassen (= som du kan anropa).

```
package laboration20;

public class Exercise20a {
    public void program() {
        // läs in information från befolkning.txt och lagra informationen
        // i en array av typen Population[].
        Population[] countries = Populations.readPopulations(
"M:/java/befolkning.txt" );

        // Lösning till deluppgift 0, avmarkera efter testkörning
        for(int i = 0; i < countries.length; i++ ) {
            System.out.println(countries[ i ].getCountry() + ", " +
                               countries[ i ].getPopulation() );
        }

        // Komplettera med kod
    }

    public static void main( String[] args ) {
        Exercise20a e20a = new Exercise20a();
        e20a.program();
    }
}
```

Deluppgifter

0. Skriv ut samtliga Population-objekt med hjälp av toString-metoden. (se lösning ovan. Testkör den innan du går vidare.)
1. Skriv ut samtliga länder (dock ej ländernas invånare).
   Ledning: Använd *getCountry*-metoden för att erhålla landet, dvs
   `countries[ i ].getCountry();`
2. Skriv ut land + invånare för de länder där antalet invånare överstiger 100 miljoner. (11 st)
   Ledning: Tilldela variabeln *inhabitants* värdet av antalet invånare:
   `long inhabitants = countries[ i ].getPopulation();`

3. Skriv ut de länder som börjar på bokstaven M (18 st).
   Ledning: Tilldela variabeln *country* värdet vid anrop av *getCountry()*-metoden.
   ```
   country = countries[ i ].getCountry();
   ```
   Med metoden *charAt* kan du få det första tecknet:
   ```
   char firstChar = country.charAt( 0 );
   ```
4. Skriv ut alla länder och invånare då länderna har 8-10 miljoner invånare. (12 st)
5. Beräknar och skriver ut antalet länder som har mindre än 1 miljon invånare, t.ex.
   ```
   AA länder har mindre än 1 miljon invånare
   ```
   där AA ersätts med antalet länder.
6. Beräknar och skriver ut antalet länder som börjar med bokstaven ’K’, t.ex.
   ```
   AA länder börjar på bokstaven 'K'
   ```
7. Lagra alla Population-objekt med 10-12 miljoner invånare i en ny Population-array. Skriv sedan ut innehållet i den nya arrayen. Gör så här:
   - Deklarera variablerna *counter*, *index* och *inhabitants*. De två första ska vara av typen *int* och initieras till 0. *inhabitants* ska vara av typen *long*.
   - Skriv en for-loop som beräknar antalet Population-objekt med 10-12 miljoner invånare. Lagra resultatet i variabeln *counter*.
   - Skapa en array med korrekt storlek:
     ```
     Population[] newArray = new Population[ counter ];
     ```
   - Skriv en for-loop som lagrar över aktuella Population-objekt. Variabeln *index* ska hålla reda på positionen i den nya arrayen:
     ```
     for( int i = 0; i < countries.length; i++ ) {
         inhabitants = countries[ i ].getPopulation();
         if( ( inhabitants >= 10000000 ) && ( inhabitants <= 12000000 ) ) {
             newArray[ index ] = countries[ i ];
             index++;
         }
     }
     ```
   - Skriv ut den nya arrayen med en for-loop.
8. Lagra alla Population-objekt där landet börjar på bokstaven ’K’ i en ny Population-array. Skriv sedan ut innehållet i den nya arrayen.
9. Skriv ut samtliga länder (dock ej ländernas invånare). Använd en förenklad for-loop.
10. Beräknar och skriver ut antalet länder som har mindre än 1 miljon invånare, t.ex.
    ```
    AA länder har mindre än 1 miljon invånare
    ```
    där AA ersätts med antalet länder. Använd en förenklad for-loop.

Uppgift 20b

Komplettera klassen ***Exercise20b*** med kod. Metoderna motsvara lösningarna i deluppgifterna 1-4 i Exercise20a.

```
package laboration20;

public class Exercise20b {

    public void printCountries( Population[] array ) {
       // Komplettera med kod
    }

    public void moreThanHundredMillions( Population[] array ) {
       // Komplettera med kod
    }

    public void startsWithM( Population[] array ) {
       // Komplettera med kod
    }

    public void eightToTenMillions( Population[] array ) {
       // Komplettera med kod
    }

    public void program() {
        Population[]countries = Populations.readPopulations(
"M:/java/befolkning.txt" );

        // Aktivera metoderna en i taget, men först när du kompletterat
        // med kod.
//        printCountries( countries );
//        moreThanHundredMillions(countries);
//        startsWithM(countries);
//        eightToTenMillions(countries);
    }

    public static void main( String[] args ) {
        Exercise20b e20b = new Exercise20b();
        e20b.program();
    }
}
```

Uppgift 20c

Komplettera klassen ***Exercise20c*** med kod. Metoderna motsvara lösningarna i deluppgifterna 5-8 i Exercise20a.

```
package laboration20;

public class Exercise20c {

    public int lessThanOneMillion( Population[] array ) {
        // Komplettera med kod
    }

    public int startsWithK( Population[] array ) {
        // Komplettera med kod
    }

    public Population[] getTenToTwelveMillions( Population[] array ) {
        // Komplettera med kod
    }

    public Population[] getStartsWithK( Population[] array ) {
        // Komplettera med kod
    }

    public void program() {
        Population[] countries = Populations.readPopulations( "M:/java/befolkning.txt" );
        Population[] res;

        // Aktivera testerna en i taget, men först när du kompletterat
        // metoderna med kod.

        // test lessThanOneMillion
//        int n = lessThanOneMillion( countries );
//        System.out.println( n + " länder har mindre än 1 miljon invånare");

        // startsWithK
//        n = startsWithK( countries );
//        System.out.println( n + " länder börjar på bokstaven 'K'");

        // test getTenToTwelveMillions
//        res = getTenToTwelveMillions( countries );
//        for( int i = 0; i < res.length; i++ ) {
//            System.out.println( res[ i ].toString() );
//        }

        // test getStartsWithK
//        res = getStartsWithK( countries );
//        for( int i = 0; i < res.length; i++ ) {
//            System.out.println( res[ i ].toString() );
//        }
//    }

    public static void main( String[] args ) {
        Exercise20c e20c = new Exercise20c();
        e20c.program();
    }
}
```

Uppgift 20d

Komplettera klassen ***Population*** med kod så att arrayen countries i nedanstående exempel sorteras växande med avseende på invånarna. Följande ska du göra:

Lägga till *implements Comparable* i klassens deklaration.

```
public class Population implements Comparable {
```

Lägga till metoden *compareTo* i klassen:

```
public int compareTo( Object obj ) {
    Population country = ( Population )obj;
    long inhabitants = country.getPopulation();
    // Nu ska this.population och inhabitants jämföras
    // Om this.population är minst ska värdet -1 returneras
    // Om inhabitants är minst ska värdet 1 returneras
    // Om this.population och inhabitants är lika stora så ska 0 returneras
}
```

Testklass:

```
package laboration20;
import java.util.*;

public class Exercise20d {
    public void program() {
        Population[] countries = Populations.readPopulations(
"M:/java/befolkning.txt" );
        Arrays.sort( countries );
        for( int i = 0; i < countries.length; i++ ) {
            System.out.println(countries[ i ].toString() );
        }
    }

    public static void main( String[] args ) {
        Exercise20d e20d = new Exercise20d();
        e20d.program();
    }
}
```

Uppgift 20e

Du ska skriva klassen ***AlphabeticalOrder***. Klassen ska användas för att sortera Population-arrayer växande med avseende på ländernas namn. Så här ska du göra:

Skapa en klass som heter *AlphabeticalOrder*. Klassen ska *implementera Comparator*. Därför måste filen importera java.util.*;

```
package laboration20;
import java.util.*;

public class AlphabeticalOrder implements Comparator {

}
```

Skriv metoden *compare* i klassen:

```
public int compare( Object obj1, Object obj2 ) {
    Population country1 = ( Population )obj1;
    Population country2 = ( Population )obj2;
    String name1 = country1.getCountry();
    String name2 = country2.getCountry();
    // Här ska du jämföra name1 med name2
    // Är name1 mindre än name2 så ska metoden returnera -1. Denna jämförelse
    // gör du så här:
    // if( name1.compareTo( name2 ) < 0 ) osv
    // Är name1 större än name2 så ska metoden returnera 1
    // Är name1 och name2 lika stora så ska metoden returnera 0

    // Ovanstående jämförelse görs korrektare med hjälp av ett Collator-objekt.
    // Skulle det funnits länder som börjar med Å eller Ä så skulle även dessa
    // ordnats på avsett sätt.
    // * importera java.text.*;
    // * Efter ovanstående fyra rader lägg till
    //   Collator coll = Collator.getInstance();
    // * Jämför sedan med coll.compare( name1, name2 )
}
```

Testklass:

```
package laboration20;
import java.util.*;

public class Exercise20e {
    public void program() {
        Population[] countries = Populations.readPopulations(
"M:/java/befolkning.txt" );
        Arrays.sort(countries, new AlphabeticalOrder() );
        for( int i = 0; i < countries.length; i++ ) {
            System.out.println(countries[ i ].toString() );
        }
    }

    public static void main( String[] args ) {
        Exercise20e e20e = new Exercise20e();
        e20e.program();
    }
}
```

Fördjupande uppgifter

Uppgift 20f

Komplettera klassen ***GradeReport*** med kod. Följande gäller:

* När metoden *setGrades* anropas så ska användaren få mata in ett antal betyg. Vilka betyg som ska matas in styrs av innehållet i fältet ämnen.
* När metoden *statistics* anropas så ska betygsstatistik skrivas ut.
* Du får gärna lägga till egna metoder i klassen.

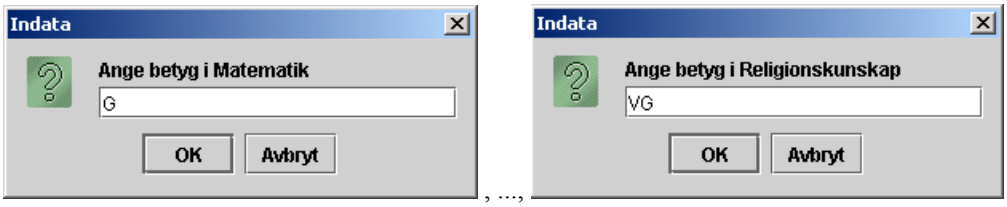
```
public class GradeReport {
  private String[] subjects = {"Matematik","Svenska","Engelska","Idrott",
           "Bild","Fysik","Biologi","Kemi","Historia","Geografi",
           "Samhällskunskap","Religionskunskap"};
  private String[] grades = new String[subjects.length]; // Lika många element som ämnen

  public void setGrades() {
    // Låt användaren mata in betyget i olika ämnen
    // Betygen ska vara "IG", "G", "VG" eller "MVG" och lagras i grades
    // Den ambitiöse ser till att varje betyg är ett tillåtet betyg
  }

  public void statistics() {
    // Beräkna antalet "IG", "G", "VG" resp "MVG". Lagra t.ex.
    // beräkningarna  i 4 olika räknare
    // Beräkna betygspoäng
    // Skriv ut betygsstatistik
  }

  public static void main(String[] args) {
    Betyg prog = new Betyg();
    prog.setGrades();
    prog.statistics();
  }
}
```

Exempel på programkörning



Uppgift 20g

Skriv ett program vilket

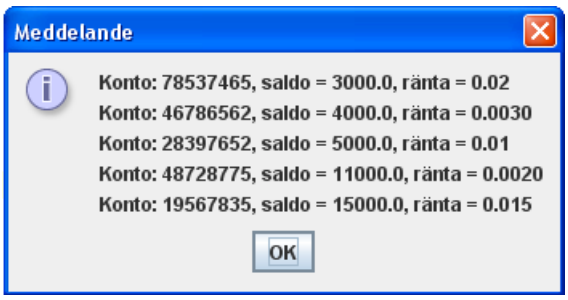
1. Skapar en int-array med 10 element
2. Tilldelar elementen i arrayen slumpvärden i intervallet 100-200.
3. Skriver ut elementen i arrayen med start på det första (skriv en metod som gör detta)
4. Sorterar arrayen
5. Skriver ut elementen i arrayen med start på det första
6. Låter elementen i arrayen byta plats så att det första elementet kommer sist och det sista elementet kommer först. (skriv en metod som gör detta)
7. Skriver ut elementen i arrayen med start på det första

Ett körresultat av programmet kan vara så här:

```
193 167 134 135 187 165 101 162 152 107
101 107 134 135 152 162 165 167 187 193
193 187 167 165 162 152 135 134 107 101
```

Uppgift 20h

I Laboration 12 arbetade du med klassen ***BankAccount***. Kopiera *BankAccount*, från paketet *laboration12,* till paketet *laboration20*. Låt klassen implementera interfacet ***Comparable*** på så sätt att *BankAccount*-objekt sorteras växande med avseende på balance.

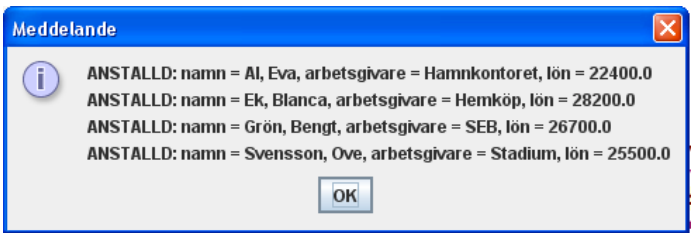


Testprogram (glöm ej importera java.util.* och javax.swing.*)

```
BankAccount[] accounts = new BankAccount[ 5 ];
String res = "";
accounts[ 0 ] = new BankAccount( "28397652", 5000, 0.01 );
accounts[ 1 ] = new BankAccount( "78537465", 3000, 0.02 );
accounts[ 2 ] = new BankAccount( "19567835", 15000, 0.015 );
accounts[ 3 ] = new BankAccount( "48728775", 11000, 0.002 );
accounts[ 4 ] = new BankAccount( "46786562", 4000, 0.003 );
Arrays.sort( accounts );
for( int i = 0; i < accounts.length; i++ ) {
    res += "Konto: " + accounts[ i ].getAccountNbr() +
           ", saldo = " + accounts[ i ].getBalance() +
           ", ränta = " + accounts[ i ].getInterestRate() + "\n";
}
JOptionPane.showMessageDialog( null, res );
```

Uppgift 20i

I Laboration 12 arbetade du med klassen ***Employee***. Kopiera *Employee*, från paketet *laboration12*, till paketet *laboration20*. Låt klassen implementera interfacet ***Comparable*** på så sätt att *Employee* -objekt sorteras växande med avseende på namnet.

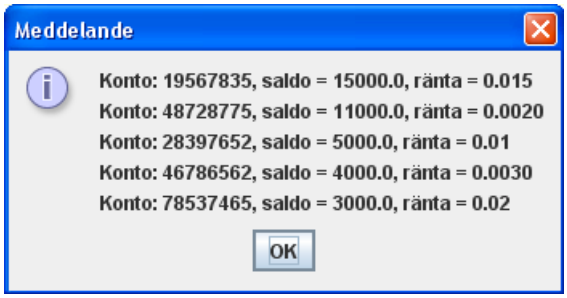


Testprogram

```
Employee[] manpower = new Employee[ 4 ];
String res = "";
manpower[ 0 ] = new Employee( "Grön, Bengt", "SEB", 26700 );
manpower[ 1 ] = new Employee( "Al, Eva", "Hamnkontoret", 22400 );
manpower[ 2 ] = new Employee( "Ek, Blanca", "Hemköp", 28200 );
manpower[ 3 ] = new Employee( "Svensson, Ove", "Stadium", 25500 );
Arrays.sort( manpower );
for( int i = 0; i < manpower.length; i++ ) {
    res += manpower[ i ].toString() + "\n";
}
JOptionPane.showMessageDialog( null, res );
```

Uppgift 20j

Skriv klassen ***BalanceDescendinge*** vilken ska implementera gränssnittet ***Comparator***. *BalanceDescendinge*-klassen ska användas vid sortering av *BankAccount*-objekt och se till att *BankAccount* -objekten sorteras så att konton med störst saldo ordnas först.



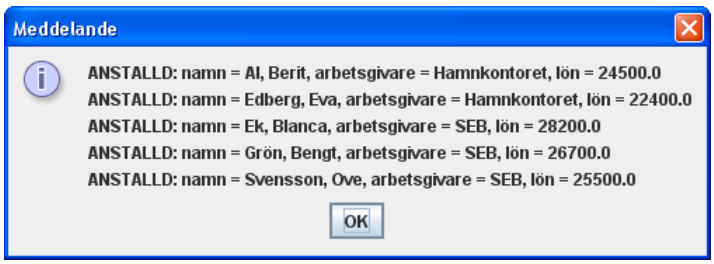
Testprogram

```
BankAccount[] accounts = new BankAccount[ 5 ];
String res = "";
accounts[ 0 ] = new BankAccount( "28397652", 5000, 0.01 );
accounts[ 1 ] = new BankAccount( "78537465", 3000, 0.02 );
accounts[ 2 ] = new BankAccount( "19567835", 15000, 0.015 );
accounts[ 3 ] = new BankAccount( "48728775", 11000, 0.002 );
accounts[ 4 ] = new BankAccount( "46786562", 4000, 0.003 );
Arrays.sort( accounts, new BalanceDescending() );
for( int i = 0; i < accounts.length; i++ ) {
    res += "Konto: " + accounts[ i ].getBankAccountNbr() +
           ", saldo = " + accounts[ i ].getBalance() +
           ", ränta = " + accounts[ i ].getInterestRate() + "\n";
}
JOptionPane.showMessageDialog( null, res );
```

Extrauppgifter

Uppgift 20k

Skriv klassen ***EmployeeSort*** vilken ska implementera gränssnittet ***Comparator***. *EmployeeSort*-klassen ska användas vid sortering av *Employee*-objekt och se till att *Employee*-objekten sorteras avseende på arbetsgivare. Om flera anställda har samma arbetsgivare ska de sorteras växande avseende namn.

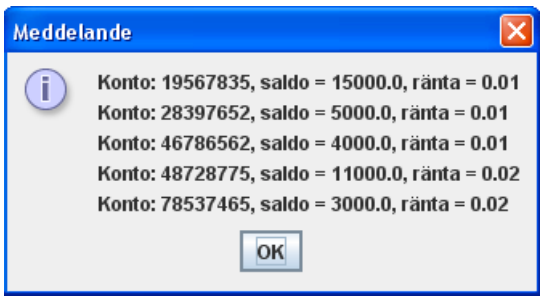


Testprogram

```
String res = "";
Employee[] manpower = new Employee[ 5 ];
manpower[ 0 ] = new Employee( "Grön, Bengt", "SEB", 26700 );
manpower[ 1 ] = new Employee( "Edberg, Eva", "Hamnkontoret", 22400 );
manpower[ 2 ] = new Employee( "Ek, Blanca", "SEB", 28200 );
manpower[ 3 ] = new Employee( "Svensson, Ove", "SEB", 25500 );
manpower[ 4 ] = new Employee( "Al, Berit", "Hamnkontoret", 24500 );
Arrays.sort( manpower, new EmployeeSort() );
for( int i = 0; i < manpower.length; i++ ) {
    res += manpower[ i ].toString() + "\n";
}
JOptionPane.showMessageDialog( null, res );
```

Uppgift 20l

Skriv klassen ***BankAccountSort*** vilken ska implementera gränssnittet ***Comparator***. *BankAccountSort*-klassen ska användas vid sortering av *BankAccount*-objekt och se till att *BankAccount*-objekten sorteras med lägst ränta först. Har två konton samma ränta ska konton med högst saldo sorteras först.



Testprogram

```
String res = "";
BankAccount[] accounts = new BankAccount[ 5 ];
accounts[ 0 ] = new BankAccount( "28397652", 5000, 0.01 );
accounts[ 1 ] = new BankAccount( "78537465", 3000, 0.02 );
accounts[ 2 ] = new BankAccount( "19567835", 15000, 0.01 );
accounts[ 3 ] = new BankAccount( "48728775", 11000, 0.02 );
accounts[ 4 ] = new BankAccount( "46786562", 4000, 0.01 );
Arrays.sort( accounts, new BankAccountSort() );
for( int i = 0; i < accounts.length; i++ ) {
    res += "Konto: " + accounts[ i ].getAccountNbr() +
           ", saldo = " + accounts[ i ].getBalance() +
           ", ränta = " + accounts[ i ].getInterestRate() + "\n";
}
JOptionPane.showMessageDialog( null, res );
```

Uppgift 20m

Uppgift 20m förutsätter att du löst Uppgift 19k och Uppgift 19l på Laboration 19.

Uppgiften går ut på att skriva göra mindre ändringar i ***ChoiceInput*** respektive ***ChoiceViewer***.

ChoiceInput

Ta en kopia av *ChoiceInput* i eclipse (i paketet labquiz). Nu ska du göra följande förändringar i *ChoiceInput*:

Ändra så att du lagrar referenserna till svarsknapparna i en array:

```
private JButton[] choiceButtons;
:
choiceButtons = new JButton[5];
```

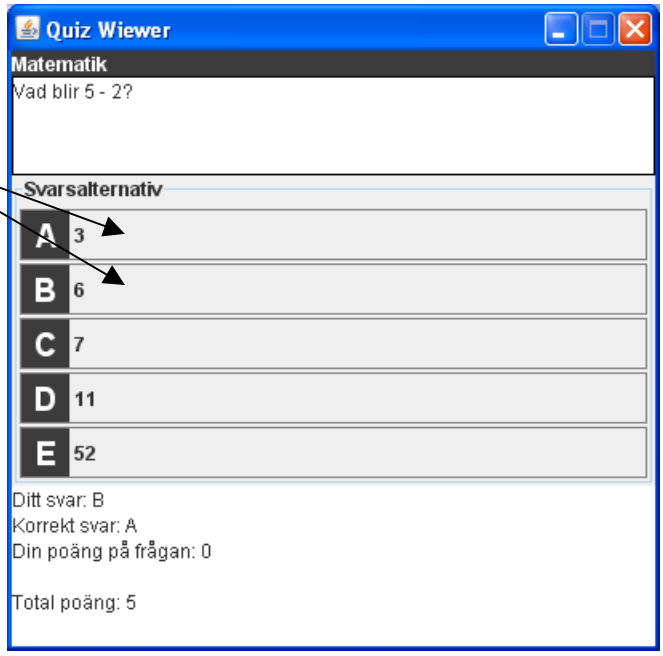
Gör sedan nödvändiga ändringar i klassen *ChoiceInput*. Kontrollera att *ChoiceInput* fungerar på samma sätt som den förra versionen.

ChoiceViewer

Ta en kopia av *ChoiceViewer* i eclipse. Nu ska du göra följande förändringar i *ChoiceViewer*:

Ändra så att svarsalternativen visas med hjälp av *JLabel*-komponenter. Eftersom svarstexterna ska kunna ändras så måste dessa *JLabel*-komponenter vara instansvariabler i klassen:

```
private JLabel[] lblChoices;
```



I fönstret till höger har jag ändrat följande:

Ytan där frågan skrivs ut är inte så hög som tidigare.

Varje svarsalternativ är en panel med två *JLabel*-komponenter. Dessa fem paneler är placerad i en panel med *TitledBorder*.

Ytan för frågan och för svarsalternativen är i en panel.

Förslag till lösningar

Uppgift 20a

```
Delupppgift 1
for(int i = 0; i < countries.length; i++ ) {
    System.out.println( countries[ i ].getCountry() );
}

Deluppgift 2
for(int i = 0; i < countries.length; i++ ) {
    inhabitants = countries[ i ].getPopulation();
    if( inhabitants > 100000000 ) {
        System.out.println( countries[ i ].toString() );
    }
}

Deluppgift 3
for(int i = 0; i < countries.length; i++ ) {
    country = countries[ i ].getCountry();
    if( country.charAt(0) == 'M' ) {
        System.out.println( countries[ i ].toString() );
    }
}

Deluppgift 4
for(int i = 0; i < countries.length; i++ ) {
    inhabitants = countries[ i ].getPopulation();
    if( ( inhabitants > 8000000 ) && ( inhabitants < 10000000 ) ) {
        System.out.println( countries[ i ].toString() );
    }
}

Deluppgift 5
counter = 0;
for(int i = 0; i < countries.length; i++ ) {
    inhabitants = countries[ i ].getPopulation();
    if( inhabitants < 1000000 ) {
        counter++;
    }
}
System.out.println( counter + " länder har mindre än 1 miljon invånare");

Deluppgift 6
counter = 0;
for(int i = 0; i < countries.length; i++ ) {
    country = countries[ i ].getCountry();
    if( country.charAt( 0 ) == 'K' ) {
        counter++;
    }
}
System.out.println( counter + " länder börjar på bokstaven 'K'");

Deluppgift 7
counter = 0;
for(int i = 0; i < countries.length; i++ ) {
    inhabitants = countries[ i ].getPopulation();
    if( ( inhabitants >= 10000000 ) && ( inhabitants <= 12000000 ) ) {
        counter ++;
    }
}
Population[] newArray = new Population[ counter ];
for( int i = 0; i < countries.length; i++ ) {
    inhabitants = countries[ i ].getPopulation();
    if( ( inhabitants >= 10000000 ) && ( inhabitants <= 12000000 ) ) {
        newArray[ index ] = countries[ i ];
        index++;
    }
```

```
}
for( int i = 0; i < newArray.length; i++ ) {
    System.out.println( newArray[ i ].toString() );
}

Deluppgift 8
counter = 0;
for(int i = 0; i < countries.length; i++ ) {
    country = countries[ i ].getCountry();
    if( country.charAt( 0 ) == 'K' ) {
        counter++;
    }
}
Population[] newArray = new Population[ counter ];
for( int i = 0; i < countries.length; i++ ) {
    country = countries[ i ].getCountry();
    if( country.charAt( 0 ) == 'K' ) {
        newArray[ index ] = countries[ i ];
        index++;
    }
}
for( int i = 0; i < newArray.length; i++ ) {
    System.out.println( newArray[ i ].toString() );
}

Deluppgift 9
for(Population pop : countries) {
    System.out.println( pop.getCountry() );
}

Deluppgift 10
counter = 0;
for(Population pop : countries) {
    inhabitants = pop.getPopulation();
    if( inhabitants < 1000000 ) {
        counter++;
    }
}
System.out.println( counter + " länder har mindre än 1 miljon invånare");
```

Uppgift 20b

```
package laboration20;

public class Exercise20b {

    public void printCountries( Population[] array ) {
        for(int i = 0; i < array.length; i++ ) {
            System.out.println( array[ i ].getCountry() );
        }
    }

    public void moreThanHundredMillions( Population[] array ) {
        long inhabitants;
        for(int i = 0; i < array.length; i++ ) {
            inhabitants = array[ i ].getPopulation();
            if( inhabitants > 100000000 ) {
                System.out.println( array[ i ].toString() );
            }
        }
    }

    public void startsWithM( Population[] array ) {
        String country;
        for(int i = 0; i < array.length; i++ ) {
            country = array[ i ].getCountry();
            if( country.charAt(0) == 'M' ) {
                System.out.println( array[ i ].toString() );
            }
        }
    }

    public void eightToTenMillions( Population[] array ) {
        long inhabitants;
        for(int i = 0; i < array.length; i++ ) {
            inhabitants = array[ i ].getPopulation();
            if( ( inhabitants > 8000000 ) && ( inhabitants < 10000000 ) ) {
                System.out.println( array[ i ].toString() );
            }
        }
    }

    public void program() {
        Population[] countries = Populations.readPopulations( "M:/java/befolkning.txt" );

        printCountries( countries );
        moreThanHundredMillions( countries );
        startsWithM( countries );
        eightToTenMillions( countries );
    }

    public static void main( String[] args ) {
        Exercise20b e20b = new Exercise20b();
        e20b.program();
    }
}
```

Uppgift 20c

```
package laboration20;

public class Exercise20c {

    public int lessThanOneMillion( Population[] array ) {
        long inhabitants;
        int counter = 0;
        for(int i = 0; i < array.length; i++ ) {
            inhabitants = array[ i ].getPopulation();
            if( inhabitants < 1000000 ) {
                counter++;
            }
        }
        return counter;
    }

    public int startsWithK( Population[] array ) {
        String country;
        int counter = 0;
        for(int i = 0; i < array.length; i++ ) {
            country = array[ i ].getCountry();
            if( country.charAt( 0 ) == 'K' ) {
                counter++;
            }
        }
        return counter;
    }

    public Population[] getTenToTwelveMillions( Population[] array ) {
        int counter = 0, index = 0;
        long inhabitants;
        for(int i = 0; i < array.length; i++ ) {
            inhabitants = array[ i ].getPopulation();
            if( ( inhabitants >= 10000000 ) && ( inhabitants <= 12000000 ) ) {
                counter++;
            }
        }
        Population[] newArray = new Population[ counter ];
        for (int i = 0; i < array.length; i++) {
            inhabitants = array[i].getPopulation();
            if ((inhabitants >= 10000000) && (inhabitants <= 12000000)) {
                newArray[index] = array[i];
                index++;
            }
        }
        return newArray;
    }

    public Population[] getStartsWithK( Population[] array ) {
        int counter = 0, index = 0;
        String country;

        counter = startsWithK( array );
        Population[] newArray = new Population[ counter ];
        for (int i = 0; i < array.length; i++) {
            country = array[i].getCountry();
            if (country.charAt(0) == 'K') {
                newArray[index] = array[i];
                index++;
            }
        }
        return newArray;
    }

    public void program() {
        Population[] counties = Populations.readPopulations( "C:/java/befolkning.txt" );
        Population[] res;
```

```
        int count = lessThanOneMillion( counties );
        System.out.println( count + " länder har mindre än 1 miljon invånare");

        count = startsWithK( counties );
        System.out.println( count + " länder börjar på bokstaven 'K'");

        res = getTenToTwelveMillions( counties );
        for( int i = 0; i < res.length; i++ ) {
            System.out.println( res[ i ].toString() );
        }

        res = getStartsWithK( counties );
        for( int i = 0; i < res.length; i++ ) {
            System.out.println( res[ i ].toString() );
        }
    }

    public static void main( String[] args ) {
        Exercise20c e20c = new Exercise20c();
        e20c.program();
    }
}
```

Uppgift 20d

```
package laboration20;

public class Population implements Comparable {
    private String country;
    private long population;

    // Konstruktorer och metoder som tidigare

    public int compareTo( Object obj ) {
        Population country = ( Population )obj;
        long population2 = country.getPopulation();
        if( this.population < population2 ) {
            return -1;
        } else if( this.population > population2 ) {
            return 1;
        } else {
            return 0;
        }
    }
}
```

Uppgift 20e

```
package laboration20;
import java.util.*;

public class AlphabeticalOrder implements Comparator {
    public int compare( Object obj1, Object obj2 ) {
        Population country1 = ( Population )obj1;
        Population country2 = ( Population )obj2;
        String name1 = country1.getCountry();
        String name2 = country2.getCountry();

        if( name1.compareTo( name2 ) < 0 ) {
            return -1;
        } else if( name1.compareTo( name2 ) > 0 ) {
            return 1;
        } else {
            return 0;
        }
//        return name1.compareTo( name2 );  // kan ersätta if-else ovan
    }
}
```

Uppgift 20e, alternativ lösning

```
package laboration20;
import java.util.*;
import java.text.*;

public class AlphabeticalOrder implements Comparator {
    public int compare( Object obj1, Object obj2 ) {
        Population country1 = ( Population )obj1;
        Population country2 = ( Population )obj2;
        String name1 = country1.getCountry();
        String name2 = country2.getCountry();
        Collator coll = Collator.getInstance();
        return coll.compare( name1, name2 );
    }
}
```

Uppgift 20f, inklusive inmatningskontroll

```
package laboration20;

import javax.swing.JOptionPane;

public class GradeReport {
    private String[] subjects = { "Matematik", "Svenska", "Engelska", "Idrott",
                      "Bild", "Fysik", "Biologi", "Kemi", "Historia", "Geografi",
                      "Samhällskunskap", "Religionskunskap" };
    private String[] grades = new String[subjects.length]; // Lika många element
                                                           // som ämnen
    private String[] levels = { "IG", "G", "VG", "MVG" };
    private int[] points = { 0, 10, 15, 20 };

    private int indexOf(String txt, String[] texts) {
        for (int i = 0; i < texts.length; i++)
            if (txt.equals(texts[i]))
                return i;
        return -1;
    }

    private String getGrade(String txt) {
        String grade;
        do {
            grade = JOptionPane.showInputDialog(txt);
        } while (indexOf(grade, levels) == -1);
        return grade;
    }

    public void getGrades() {
        for (int i = 0; i < grades.length; i++) {
            grades[i] = getGrade(subjects[i]);
        }
    }

    public void statistics() {
        int[] stat = new int[levels.length];
        String res;
        int index, sum = 0;

        for (int i = 0; i < grades.length; i++) {
            index = indexOf(grades[i], levels);
            stat[index]++; // Går bra - inmatningskontroll!
            sum += points[index];
        }

        res = "BETYGSSTATISTIK\n";
        for (int i = 0; i < levels.length; i++) {
            res += levels[i] + ": " + stat[i] + "\n";
        }
        res += "Betygspoäng: " + sum;
        JOptionPane.showMessageDialog(null, res);
    }

    public static void main(String[] args) {
        GradeReport prog = new GradeReport();
        prog.getGrades();
        prog.statistics();
    }
}
```

Uppgift 20g

```
package laboration20;

public class Exercise20g {

    private void randomArray(int[] array, int min, int max) {
        for(int i=0; i<array.length; i++) {
            array[i] = (int)(Math.random()*(max-min+1)+min);
        }
    }

    private void print(int[] array) {
        for(int i=0; i<array.length; i++) {
            System.out.print(array[i]+" ");
        }
        System.out.println();
    }

    private void reverse(int[] array) {
        int temp, lastIndex = array.length-1;
        for(int i=0; i<array.length/2; i++) {
            temp = array[i];
            array[i] = array[lastIndex-i];
            array[lastIndex-i] = temp;
        }
    }

    public void program() {
        int[] numbers = new int[10];
        randomArray(numbers,100,200);
        print(numbers);
        java.util.Arrays.sort(numbers);
        print(numbers);
        reverse(numbers);
        print(numbers);
    }

    public static void main(String[] args)  {
        Exercise20g e20g = new Exercise20g();
        e20g.program();
    }
}
```

Uppgift 20h

```
package laboration20;
import javax.swing.*;

public class BankAccount implements Comparable {
    private String accountNbr;
    private double balance;
    private double interestRate;

    // konstruktorer och metoder sedan tidigare

    public int compareTo( Object obj ) {
        BankAccount konto = ( BankAccount )obj;
        double saldo = konto.getBalance();
        if( this.balance < balance ) {
            return -1;
        } else if( this.balance > balance ) {
            return 1;
        } else {
            return 0;
        }
    }
}
```

Uppgift 20i

```
package laboration20;

public class Employee implements Comparable {
    private String name;
    private String employer;
    private double wage;

    // konstruktorer och metoder sedan tidigare

    public int compareTo( Object obj ) {
        Employee emp = ( Employee )obj;
        String name = emp.getName();
        return this.name.compareTo( name );
    }
}
```

Uppgift 20j

```
package laboration20;
import java.util.*;

public class BalanceDescending implements Comparator {
    public int compare( Object obj1, Object obj2 ) {
        BankAccount account1 = ( BankAccount )obj1;
        BankAccount account2 = ( BankAccount )obj2;
        double balance1 = account1.getBalance();
        double balance2 = account2.getBalance();
        if( balance1 < balance2 ) {
            return 1;
        } else if( balance1 > balance2 ) {
            return -1;
        } else {
            return 0;
        }
    }
}
```

Uppgift 20k

```
package laboration20;
import java.util.*;

public class EmployeeSort implements Comparator {
    public int compare( Object obj1, Object obj2 ) {
        Employee emp1 = ( Employee )obj1;
        Employee emp2 = ( Employee )obj2;
        int resultat = emp1.getEmployer().compareTo( emp2.getEmployer() );
        if( resultat == 0 ) {
            resultat = emp1.getName().compareTo( emp2.getName() );
        }
        return resultat;
    }
}
```

Uppgift 20l

```
package laboration20;
import java.util.*;

public class BankAccountSort implements Comparator {
    public int compare( Object obj1, Object obj2 ) {
        BankAccount account1 = ( BankAccount )obj1;
        BankAccount account2 = ( BankAccount )obj2;
        double interestRate1 = account1.getInterestRate();
        double interestRate2 = account2.getInterestRate();
        double balance1 = account1.getBalance();
        double balance2 = account2.getBalance();
        if( interestRate1 < interestRate2 ) {
            return -1;
        } else if( interestRate1 > interestRate2 ) {
            return 1;
        } else if( balance1 < balance2 ) {
            return 1;
        } else if( balance1 > balance2 ) {
            return -1;
        } else {
            return 0;
        }
    }
}
```

Uppgift 20m

```
package labquiz;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ChoiceInput extends JPanel {
    :
    private JButton[] choiceButtons;
    private String labels = "ABCDEFGHIJ";

    public ChoiceInput() {
        :
        add(panelAnswer(), BorderLayout.CENTER);
        addListeners();
    }

    private JPanel panelAnswer() {
            JPanel panel = new JPanel(new GridLayout(1,5));
            panel.setBorder(BorderFactory.createTitledBorder("Svar"));
            choiceButtons = new JButton[5];
            for(int i=0; i<choiceButtons.length; i++) {
                        choiceButtons[i] = new JButton(labels.substring(i,i+1));
                        panel.add(choiceButtons[i]);
            }
            return panel;
    }

    private void addListeners() {
        AnswerListener al = new AnswerListener();
        :
        for(JButton btn : choiceButtons) {
            btn.addActionListener(al);
        }
    }

    public void setController(Controller controller) {...}

    public void enableAnswer(boolean enabled) {
        for(JButton btn : choiceButtons) {
            btn.setEnabled(enabled);
        }
    }

    public void enableStart(boolean enabled) {...}

    public void enableStop(boolean enabled) {...}

    private class AnswerListener implements ActionListener {
        :
    }
}

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

public class ChoiceViewer extends JPanel {
    private JLabel lblTitle = new JLabel(" ");
    private JTextArea taQuestion = new JTextArea();
    private JTextArea taMessage = new JTextArea();
    private JLabel[] lblChoices;
    private Font choiceLabelFont = new Font("SansSerif",Font.BOLD,20);
    private String labels = "ABCDEFGHIJ";

    public ChoiceViewer() {
        JPanel panelQuestion = new JPanel(new BorderLayout());
        setLayout(new BorderLayout());
        taQuestion.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        lblTitle.setBackground(Color.DARK_GRAY);
        lblTitle.setForeground(Color.WHITE);
```

```
        lblTitle.setOpaque(true);
        taQuestion.setPreferredSize(new Dimension(400,60));
        taMessage.setPreferredSize(new Dimension(400,100));
        panelQuestion.add(taQuestion,BorderLayout.CENTER);
        panelQuestion.add(panelChoices(),BorderLayout.SOUTH);
        add(lblTitle,BorderLayout.NORTH);
        add(panelQuestion,BorderLayout.CENTER);
        add(taMessage,BorderLayout.SOUTH);
    }
    private JPanel panelChoice(JLabel lblChoice, String choiceLabel) {
        JPanel panel = new JPanel(new BorderLayout());
        JLabel lbl = new JLabel(choiceLabel);
        lbl.setPreferredSize(new Dimension(30,30));
        lbl.setBackground(Color.DARK_GRAY);
        lbl.setForeground(Color.WHITE);
        lbl.setOpaque(true);
        lbl.setFont(choiceLabelFont);
        lbl.setHorizontalAlignment(SwingConstants.CENTER);
        panel.setBorder(BorderFactory.createLineBorder(Color.GRAY));
        panel.add(lbl,BorderLayout.WEST);
        panel.add(lblChoice,BorderLayout.CENTER);
        return panel;
    }
    private JPanel panelChoices() {
        JPanel panel = new JPanel(new GridLayout(5,1,0,2));
        JPanel[] pnlChoices = new JPanel[5];
        panel.setBorder(BorderFactory.createTitledBorder("Svarsalternativ"));
        lblChoices = new JLabel[5];
        for(int i=0; i<pnlChoices.length; i++) {
            lblChoices[i] = new JLabel(" ");
            pnlChoices[i] = panelChoice(lblChoices[i],labels.substring(i,i+1));
            panel.add(pnlChoices[i]);
        }
        return panel;
    }
    public void clearQuestionArea() {
        lblTitle.setText(" ");
        taQuestion.setText("");
        for(JLabel lbl : lblChoices) {
            lbl.setText(" ");
        }
    }
    public void clearMessageArea() {
        taMessage.setText("");
    }
    public void setQuestion(ChoiceQuestion question) {
        lblTitle.setText(question.getTitle());
        taQuestion.setText(question.getQuestion());
        String[] choices = question.getChoices();
        for(int i=0; choices!=null && i<choices.length; i++) {
            lblChoices[i].setText(" " + choices[i]);
        }
    }
    public void setMessage(String message) {
        taMessage.setText(message);
    }
}
```