Chapter 7-8: Arrays

I dessa uppgifter kommer du att lära dig om hur man arbetar med arrays, dvs samling av data av samma typ. Arrays är objekt i java och skapas med ordet **new**.

Till exempel en array av heltal(int) skapas på följande sätt:

```
int[] minaTal= new int[50];

String[] klassITF =new String[63];

Item[] basket =new Item[100];
```

Så, tekniken att deklarera och skapa array är desamma, det som skiljer är typen av data som skall finnas i arrayen. Typen står alltid angivet så som du ser i min exempel.

GLÖM INTE!
När du skapar arrayen är den TOM. För att fylla den med värde skall värde tilldelas till arrayens plastser (element).

```
int[0]= 34;  // då har vi lagt 34 på första plastsen i arrayen

klassITF[5]= ”Ida Ek” // då har vi ”Ida Ek” i arrayen på plats 5

basket[3]= new Item( ”skor”, 300); // vi har gjort först ett Item-objekt, sedan har vi tilldelat det till arrayen på plats 3.
```

Viktigt att du läser i boken innan du börjar med uppgifterna.

Skapa och vända på en array

I en klass kallad **Arrays** gör följande:

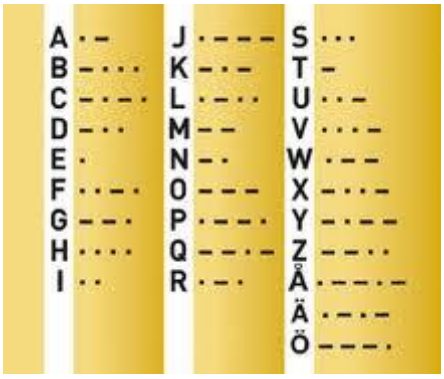
A. Skriv ett program som frågar användaren om ett heltal. Programmet ska skapa en array lika stor som det angivna värdet. Därefter skall användaren bli ombedd om att ange ett heltalsvärde tills arrayen fylls.
Använd en while-loop.

1. Skriv ut alla värden i arrayen i samma ordning som de skrevs in
2. Skriv ut alla värden i omvänd ordning, dvs. sista värdet först
3. Beräkna summan av alla värden i arrayen.

B. Ändra nu så att 1,2,3 är separata metoder som tar som argument en array och gör det som anges. Du bestämmer om metoderna skall returnera eller vara void.
Anropa sedan metoderna i main och visa att programmet fungerar som tidigare.

Morsealfabetet

I maj 1844 skickade Samuel F.B. Morse meddelandet "*What hath God wrought*!" med hjälp av telegraf från Washington till Baltimore och blev början på elektronisk kommunikation.
För att göra det möjligt att kommunicera information enbart med hjälp av närvaro eller frånvaron av en enda ton, utvecklade Morse ett kodningssystem där bokstäver och andra symboler representeras som kodade sekvenser av korta och långa toner, traditionellt kallas prickar och streck.
I Morsealfabetet representeras de 26 bokstäver av följande koder. Endast versaler används i Morse.



Du kan enkelt lagra dessa koder i ett program genom att deklarera en array med 29 element och lagra sekvensen av tecken ”. – ” som motsvarar varje bokstav, kalla den morseArray.

1. Skriv ett program som läser en sträng från användaren och översätter varje bokstav i strängen till dess motsvarighet i Morsekod.
Separera orden med hjälp av System.out.println () men ignorerar alla andra skiljetecken.
Tips! Se nedan några logiska struktur för ett sådant program

```
String morse= ””;
char ch=0;
int counter =0;
while (  counter< message.length())
{

ch=message.charAt( counter);
switch ( ch)
{
  case ‘A’:  morse= morse +  morseArray [0]  // if the first
place correspond to
                    break;

 case ‘B’:  morse=
                 break;

 case ‘C’:

}
 counter++;

}
…..
```

WebShop Simulering

Om man skall simulera en väldigt enkel WebShop kan man tänka sig följande meny:

```
System.out.println("\n  Välj från meny");
System.out.println("   =========");
System.out.println("0: Exit ");
System.out.println("1: Köp" );
System.out.println("2: Ta bort" );
System.out.println("3: Visa korg");
System.out.println("4: Räkna total");
System.out.println("5: Töm, korg");
```

Programmet nedan är skallet på en sådant program. Spara den i din arbetsmap. Som du ser nedan, ”korgen” är en array av typen Item. Klassen Item finns nedan också. Den är i princip komplett. Kopiera och spara i samma map med WebShop klassen. Det du skall göra är att komplettera klassen med de olika case som finns beskrivna i meny.

```
import java.util.Scanner;

    public class WebShop
   {
      static Item [] basket= new Item[20];
      static Scanner scan=new Scanner(System.in);
      static int index=0;

       //-------------------------------------------------------
--
       public static void main(String[] args)
      {
        System.out.println( " I den här shopen finns allt du
vill, bara lägg produkten i din korg");
        System.out.println( " Priset bestämmer du själv
också...");

         printMenu();
         int choice = scan.nextInt();

      // programmet stanar kvar i loopen tills användaren
      //matar in 0
         while (choice != 0)
         {
            dispatch(choice);
            printMenu();
            choice = scan.nextInt();
         }
```

```
        } // main slutar här


       // Skriver ut användar möjligheter
        public static void printMenu()
       {
          System.out.println("\n  Välj från meny");
          System.out.println("   =========");
          System.out.println("0: Exit ");
          System.out.println("1: Köp" );
          System.out.println("2: Ta bort" );
          System.out.println("3: Visa korg");
          System.out.println("4: Räkna total");
          System.out.println("5: Töm, korg");

         System.out.print("\nMata in dit val: ");
       }

// metoden dispach behandlar och gör beräkningar beroende på
//val
        public static void dispatch(int choice)
       {

         switch(choice)
          {
             case 0:
                    System.exit(0);
                    break;
             case 1:
                    // läs in från användare namn och pris
                    // skapa Item- objekt, lägg den i korg på
                    // plats index.
                    // variabeln index, uppdateras.
                   break;

             case 2:
                    // för att kunna implementera ta bort, måste
                    //du först söka efter
                    // en produkt som har ett visst namn
                    // detta visar jag på föreläsning

                   break;

             case 3:
                    // loopa genom arrayen och anropa metoden
//toString() för varje objekt i arrayen. Glöm inte att dina
//objekt heter nu basket[0], basket[1]...
```

```
                break;
            case 4:
               // ha en variabel som uppdaterar det totala
korgpriset

            break;
            case 5:

       // nullställ alla positioner i arrayen



            default:
               System.out.println("Sorry, fel val");
         }
      }


 }
```

```
import java.text.*;

public class Item {
     String name;
     double price;

    // ----------------------------------------------------
    //  Create a new item with the given attributes.
    // ----------------------------------------------------

   public Item (String itemName, double itemPrice)
   {
   name = itemName;
   price = itemPrice;


    }


    // ----------------------------------------------------
    //   Return a string with the information about the item
    // ----------------------------------------------------
    public String toString ()
    {
   NumberFormat fmt = NumberFormat.getCurrencyInstance();
```

```
return (name + "\t" + fmt.format(price));

 }

 // -----------------------------------------------
 //   Returns the unit price of the item
 // -----------------------------------------------
 public double getPrice()
 {
return price;
 }

 // -----------------------------------------------
 //   Returns the name of the item
 // -----------------------------------------------
 public String getName()
 {
   return name;
 }

}
```

Cesar kryptering

Caesars kod eller Caesar skift, är en av de enklaste och mest kända krypteringsteknik. Det är en typ av chiffer där varje bokstav i klartexten ersätts av en bokstav något fast antal positioner ner i alfabetet. Till exempel, med en förskjutning av 3, skulle en A ersättas med D, B skulle bli E, och så vidare. Metoden är uppkallad efter Julius Caesar, som använde det för att kommunicera med sina generaler.

Att avkoda meddelandet innebär att varje bokstav skiftas samma antal tecken bakåt.

En förbättring kan göras till denna kodningsteknik om vi använder en upprepande nyckel. Istället för att flytta varje tecken med ett konstant värde, vi kan skifta varje karaktär med olika värde genom att använda en array av nyckel värde.

Till exempel om arrayen med nyckel innehåller värdena 3 1 7 4 2 5, då första tecknet förskjuts med tre, det andra tecknet med ett, den tredje av sju, etc..

n	o	v	a	n	j	g	H	l		m	u		u	r	x	l	v
3	1	7	4	2	5	3	1	7		4	2		5	3	1	7	4
k	n	o	w	l	e	d	G	e		i	s		p	o	w	e	r

Den första raden är det kodade meddelandet, den andra är nycklarna, den tredje är det avkodade meddelandet.

Skriv metoder för att kryptera och dekryptera meddelande. Endast printbara tecken kommer att krypteras och dekrypteras (inte blanksteg).

1. Använd ASCII-tabellen för tecknet och nyckeln för att kryptera / dekryptera.

Börja med att skriva följande metoder i en klass som heter Caesar. Metoderna tar som argument en tecken och en nyckel och returnerar en annan tecken.

public static char encrypt_character (char tkn, int key) {….your code… }

public static char decrypt_character (char tkn, int key) {…your code…}

2. Implementera nu metoderna **encrypt_message () och decrypt_message() .** För detta skall du använda de tidigare skrivna metoderna **encrypt_character, and decrypt_character.**

Tips! Du har en array där du sparar dina nycklar. Du skall också ha en variabel som håller koll på vilken nyckel som står på tur att användas, kalla variabeln **currentKey.**

När du har kommit till slutet av arrayen med currentKey, skall variabeln pecka tillbaka till början.

3. Använd metoderna i ett program. Om du kan skall du inte ”hårdkoda” arrayen med nycklar, användaren skall kunna mata hur många nycklar och vilka nyckelvärde programmet skall använda.

Extra:

Hitta en lösning så att din kryptering producerar enbart alfabetiska tecken, dvs. inga # € &, mm samt siffror. Du behöver inte ha med de svenska bokstäverna ”ä,ö,å ” , men gör det gärna om du hinner.