Inlämningsuppgift 1

Uppgiften handlar om arrayer samt ArrayList.

Det är viktigt att du kan visa vad du kan programmera när du söker arbete eller praktik.

Därför skall du skapa dig en portfölj med alla dina lösningar. Du sätter ihop dina lösningar (källkoden) tillsammans med svaret på frågorna i en digital mapp. Du skall ha en mapp för varje inlämningsuppgift.

För varje uppgift: Tänk på att du skall strukturera koden genom att dela den i så många metoden det behövs för att den skall vara tydligt och lättläst. Variabel och metodnamnen (identifierare) skall vara relevanta. Kommentera koden väll.

Uppgift 1. Arrayer och java.util.Arrays

1.1 Skriv metoden **public static int [] generateArray**(int size, int min, int max) {}. Metoden skall skapa och returnera en array av storlek size, fylld med ***unika*** värden i intervallet min till max.

1.2 Skriv metoden **public static boolean passwordCheck (char [] password)** som tar som argument en ”password” i form av en array av char. Metoden skall returnera true om passworden är korrekt enligt följande:

- mellan 8 -12 tecken långt
- innehåller minst en siffra 0-9,
- innehåller minst en stor bokstav,
- innehåller minst en liten bokstav,

Skriv ett program som läser in en lösenord från tangentbordet och undersöker om lösenordet är korrekt eller inte. **OBS!** Inputen från tangentbordet är av typen String som skall göras om till char[] innan metoden anropas.

1.3 Skriv en metod som **public static char[] generatePassword(){}** som skapar och returnerar en lösenord enligt den beskrivningen du fick innan.

Gör om programmet (som du skapade i uppgift 1.2) så att den genererar 1000 lösenord med *generatePassword()* och testar dessa med metoden *passwordCheck()*. Hur många av lösenorden klarar testet? Få programmet skriva ut detta i %.

1.4 Skriv en static metod som givet ett argument av typen int [], ett heltal samt ett index placerar in heltalet på den position som ges av indexet samt flyttar alla element i fältet med samma eller högre index ett steg ”uppåt". Det sista elementet i fältet försvinner ur fältet om antalet tal i fältet skulle bli för stort. För att kunna testa metoden skriv ett enkelt main () där du anropar metod ett antal gånger. Skriv ut arrayen med **Arrays.toString(int[ ])** från i util-paketet.

1.5 Skriv ett program som kan läsa en text från en fil och göra frekvensanalys på de bokstäver som förekommer (ignorera ä,ö,å). För att lagra information (frekvensen) skall du använda en array. Utdata skall ges i form av en tabell:

boks	**antal**	**frekv**
a	**112**	**9.33%**
b	**15**	**1.25%**

. ...
och så vidare

Frekvensen anges i % och endast bokstäver som förekommer minst 1 gång tas med.

1.6 Skriv en metod
public static int [ ] mergeArrays(int [ ] a , int [] b) { }
som givet två arrayer flätar samma dem. Anta att arrayerna är redan sorterade. Resultat arrayen skall också vara sorterad. Den nya arrayen har storleken av a+b.
Metoden skall först kolla om någon av arrayera a och b är null. I sådant fall skall metoden kasta ett lämpligt Exception objekt. Tänk på att arrayerna inte är lika stora. Metoden skall returnera den nya arrayen.

Testa metoden i ett program som först genererar arrayer med metoden **generateArray().** Sorterar dessa arrayer med **javas Arrays.sort().** Bevisa också med hjälp av kod att resultat är en sorterad array!

Uppgift 2. Använd javas färdiga datastrukturer från java.util. ArrayList

Gå till java API och undersök klassen ArrayList.

Uppgift 2.1

a) Skriv ett program som läser en fil innehållande namn (se bifogad fil) och lägger alla namn i ett ArrayList-objekt kallad klasslista. Läsa fil gör du enklast med javas ***Scanner scan = new Scanner (new File (”filnamn”)).***

b) Skriv en metoden public ***static boolean addToList (ArrayList list, String name)*** som tar som argument en ArrayList-objekt ***list*** som innehåller en lista med namn i alfabetisk ordning och ett namn. Metoden skall undersöka om namnet finns i listan. Om den inte redan finns skall namnet läggas till listan på rätt plats (dvs.listan måste förbli sorterad) och metoden skall returnera true. Om namnet redan finns skall metoden returnera false.

c) Komplettera programmet från punkt a så att den anropar metoden addToList några gånger, både med namn som redan finns och med namn som inte finns.
Hur kan du bevisa att programmet fungerar korrekt. Dvs. att lista alltid är sorterad och inte innehåller dubbletter? Skriv ett program som kan testa detta och bevisa det är rätt.

Uppgift 2.2 Lottospel

Ladda ner följande klass **Game** och **Player**.

-Undersök och förstå Player och Game objekten, gör en skiss på papper. Vilken relation finns mellan dessa objekt?
-Undersök och exekvera programmet Game. Försök förstå hur programmet är uppbyggt.

Som du ser är lotteriet är en ArrayList av Player. I sin tur en Player är en ArrayList av arrayer. Varje array är en lottorad.

Nedan en beskrivning om hur ungefär spelet skall fungera

Programmet skall göra följande:

1. Fråga användaren efter sitt namn och hur många lottorader som användaren önskar. Därefter skall så många rader skapats och fyllas med lottonummer. Lottnummer är i intervalet 1-35. Dessa nummer skall slumpas men så klart får inte en rad innehålla samma nummer 2 gånger. En lottorad ska vara 7 nummer. En lottorad skall vara en array men alla lottorader skall sparas i en ArrayList (se klassen Player). **Extra** (ej obligatoriskt): Om du vill kan du läsa lottoraderna från en fil. Filen med lottorader kan du skapa själv eller kan generera av ett annat program.

2. Skapa lottodragningen. En lottodragning är en rad med 10 nummer i intervalet 1-35.
3. Efter dragningen skall programmet söka och hitta de lottorader som har 7 rätt sedan med 6 rätt och 5 rätt resten är ointressant. Dessa rader kan hållas sorterade efter antal rätt.
4. Anta att man betalar 20 kr/tippad rad. Programmet skall dela ut vinsten t.ex enligt följande:
   Fördela 70% av pengarna till de rader som har 7 rätt. Fördela 20% av pengarna till de rader som har 6 rätt. Fördela 10% av pengarna till de rader som har 5 rätt

   Om det inte finns någon med 7 rätt skall en del av pengarna tilldelas till de med 6 rätt (20%) och till 5 rätt 20%. De resterande 30% går till lotteriets kassan. Om det inte finns någon med 6 rätt går 20% av pengarna till 5 rätt. Eller liknande, bestäm det själv..

e) Frågor att besvara

1. Vilka metoder från klassen ArrayList använde du?
2. Vad är skillnaden mellan en grundläggande array och en ArrayList datastruktur
3. I vilka situationer skulle du välja en array framför datastrukturen ArrayList.
4. Kan dina testprogram bevisa att koden är korrekt? Varför? Varför inte?