

Omkontrollskrivning, Programmeringsteknik för D/C

2015–05–08, 8.00–13.00

Anvisningar: Preliminärt ger uppgifterna $8 + 16 + 10 + 6 = 40$ poäng. För godkänt betyg krävs ungefär 25 poäng. Tillåtet hjälpmedel: Java-snabbpreferens.

Betyget på denna skrivning utgör betyget på kontrollskrivningarna, oavsett resultatet på tidigare skrivningar (dock sänks inte ett tidigare godkänt betyg). Resultatlistan anslås på institutionens anslagstavla när rättningen är klar.

1. I schack rör sig pjäserna på olika sätt. En springare ("häst", "knight" på engelska) kan i ett drag flytta sig två rutor i en riktning och därefter plus/minus en ruta i sidled. I nedanstående bild markeras med grått de rutor till vilka en springare på rutan markerad med 1 kan flytta sig.

	1			

Ett schackbräde har 8×8 rutor som vi numrerar från 0,0 (övre vänstra hörnet) till 7,7 (nedre högra hörnet). Om man placerar en springare på en godtycklig ruta kan den i 64 drag besöka brädets samtliga rutor. Två exempel på detta (siffrorna i rutorna anger dragets nummer):

1	16	53	34	3	18	21	36
52	33	2	17	54	35	4	19
15	60	55	48	43	20	37	22
32	51	46	59	56	49	42	5
61	14	57	50	47	44	23	38
28	31	62	45	58	41	6	9
13	64	29	26	11	8	39	24
30	27	12	63	40	25	10	7

24	11	8	33	26	21	6	3
9	34	25	22	7	4	27	20
12	23	10	61	32	29	2	5
35	64	51	30	1	62	19	28
52	13	60	63	56	31	44	41
59	36	55	50	45	42	47	18
14	53	38	57	16	49	40	43
37	58	15	54	39	46	17	48

Du ska skriva ett program som beskriver en springares vandring över ett schackbräde. I programmet ska det finnas två klasser: Board som beskriver schackbrädet och Knight som beskriver springaren. Exempel på användning:

```
public static void main(String[] args) {
    Board board = new Board();           // schackbrädet
    Knight knight = new Knight(board);    // springaren
    knight.walk(0, 0);                    // startposition (som i vänstra bilden)
    board.print();                         // skriv ut resultatet
}
```

Klassen Board har följande specifikation:

```
/** Skapar ett schackbräde med 8x8 rutor */
Board();

/** Markerar alla rutor som obesökta */
void clear();

/** Returnerar true om rutan med index x,y finns på brädet och inte har
    besökts, false annars */
boolean isFree(int x, int y);

/** Markerar att rutan med index x,y besöks i drag nummer moveNbr (rutan
    finns säkert på brädet, behöver ej testas) */
void visit(int x, int y, int moveNbr);

/** Skriver ut schackbrädet med numren på de besökta rutorna */
void print();
```

Implementera klassen. Det har ingen betydelse om x och y är rad- och kolonnindex eller tvärtom.

2. Klassen Knight har följande specifikation:

```
/** Skapar en springare som ska vandra över brädet board */
Knight(Board board);

/** Låt springaren besöka brädets alla rutor med början i xStart, yStart */
void walk(int xStart, int yStart);
```

Implementera klassen. Följande algoritm ska utnyttjas:

1. Placera springaren på startrutan.
2. I de följande dragen (nummer 2 till 64):
 - Undersök alla rutorna som kan nås från den aktuella rutan. Räkna ut antalet möjliga förflyttningar från varje sådan ruta.
 - Gå till den ruta varifrån antalet möjliga förflyttningar är minst. Exempel: när drag nummer 28 ska göras finns tre alternativ (de grå rutorna). Från dessa rutor kan springaren flytta till 4, 5 respektive 3 andra rutor. Rutan med 3 väljs.

24	11	8		26	21	6	3
9		25	22	7	4	27	20
12	23	10				2	5
				1		19	
	13						
							18
14				16			
		15				17	

Anvisningar:

- Skriv två privata metoder i klassen Knight: `nbrFreeFrom(x,y)` som ger antalet möjliga förflyttningar från rutan x,y och `oneMove(moveNbr)` som gör drag nummer moveNbr.
- Definiera följande attribut i klassen:

```
private static final int[] dx = { 2, 2, -1, 1, -2, -2, -1, 1 };
private static final int[] dy = { -1, 1, -2, -2, -1, 1, 2, 2 };
```

Om springarens aktuella position är x,y så är $x+dx[i]$, $y+dy[i]$, $i=0..7$ de rutor som springaren kan flytta till.

3. I en del ordbehandlingsprogram kan man producera ett sakregister (index) till en bok. Man markerar varje ord som ska ingå i registret, och programmet producerar en fil (idx-fil) med ord och numret på sidan där ordet förekommer. En liten del av en idx-fil kan se ut så här:

```
hund 4
katt 4
hamster 4
katt 4
hund 5
katt 6
hamster 6
undulat 6
katt 7
```

Det finns ett program med namnet `makeindex` som läser en idx-fil och producerar en ny fil där alla ord finns med (i bokstavsordning) och där alla sidnummer finns med (också i ordning). Om ett ord förekommer flera gånger på samma sida finns bara sidnumret med en gång. Med ovanstående idx-fil blir resultatet:

```
hamster 4, 6
hund 4, 5
katt 4, 6, 7
undulat 6
```

Du ska skriva en förenklad version av programmet `makeindex`. Ett ord med tillhörande sidnummer beskrivs av klassen `WordElement`:

```
/** Skapar ett ordelement med ordet word och utan några sidnummer */
WordElement(String word);

/** Tar reda på ordet */
String getWord();

/** Lägger till ett sidnummer. Om numret redan finns händer ingenting */
void addPage(int pageNbr);

/** Returnerar ordet och sidnumren, till exempel "katt 4, 6, 7" */
String toString();
```

Alla ordelementen samlas i ett objekt av klassen `WordList`:

```
/** Skapar en tom ordlista */
WordList();

/** Lägger till ett ord och ett sidnummer. Om ordet redan finns läggs
    bara sidnumret till */
void addWord(String word, int pageNbr);

/** Tar reda på antalet ord i listan */
int size();

/** Ger ordet och sidnumren för ord nummer nbr (0..size-1) i den sorterade
    ordningen, till exempel "katt 4, 6, 7" */
String getWordLine(int nbr);
```

- Klassen `WordElement` är färdigskriven. Implementera klassen `WordList`. Du *ska* använda en `ArrayList` för att hålla reda på ordelementen (*inte* en `HashMap` eller något liknande). Listan ska hela tiden vara sorterad.
- Skriv ett testprogram som läser ord och sidnummer från tangentbordet (som i exemplet högst upp på sidan) och skriver resultatet på skärmen (som i det andra exemplet).