Ísis Ardisson Logullo 7577410

Relatório EP1

Para rodar o EP1 apenas use “make” pelo terminal no diretório que se encontram os arquivos do ep. Ele está utilizando o arquivo entrada1.txt como padrão. Para usar outras entradas apenas mude no makefile o norme do arquivo. Ou rode “java Pentamino seuarquivo.txt” após utilizar o “make” pelo menos uma vez.

Sobre a implementação do EP, comecei pela classificação de cada petaminó e como poderia guardá-los para utilização nas inserções. São ao todo 12 peças, mas cada uma pode ter de 1 a 8 posições.
Abaixo estão as relações de cada um:

```
f:      f1 f1      f2          f3       f4          f5 f5          f6          f7              f8
     f1 f1      f2 f2 f2       f3 f3    f4 f4 f4       f5 f5    f6 f6 f6    f7 f7         f8 f8 f8
        f1            f2     f3 f3        f4          f5        f6              f7 f7       f8
```

```
i:     i1       i2 i2 i2 i2 i2
       i1
       i1
       i1
       i1
```

```
l:   l1         l2 l2 l2 l2     l3 l3              l4          l5      l6            l7 l7      l8 l8 l8 l8
     l1         l2                 l3      l4 l4 l4 l4          l5      l6 l6 l6 l6   l7                 l8
     l1                            l3                           l5                    l7
     l1 l1                         l3                        l5 l5                    l7
```

```
n:      n1    n2 n2              n3     n4 n4 n4        n5          n6 n6 n6    n7              n8 n8
        n1       n2 n2 n2     n3 n3          n4 n4     n5       n6 n6          n7 n7     n8 n8 n8
     n1 n1                      n3                     n5 n5                      n7
     n1                         n3                        n5                      n7
```

```
p:   p1 p1    p2 p2 p2       p3      p4 p4        p5 p5        p6 p6     p7          p8 p8 p8
     p1 p1       p2 p2    p3 p3     p4 p4 p4     p5 p5     p6 p6 p6     p7 p7       p8 p8
     p1                   p3 p3                     p5                  p7 p7
```

```
t:   t1 t1 t1            t2          t3          t4
        t1          t2 t2 t2          t3          t4 t4 t4
        t1               t2       t3 t3 t3       t4
```

```
u:   u1    u1      u2 u2      u3 u3 u3      u4 u4
     u1 u1 u1      u2         u3    u3        u4
                   u2 u2                     u4 u4

v:    v1           v2 v2 v2    v3 v3 v3           v4
      v1           v2                 v3          v4
      v1 v1 v1     v2                 v3     v4 v4 v4

w:   w1                 w2 w2      w3 w3               w4
     w1 w1          w2 w2            w3 w3         w4 w4
        w1 w1       w2                    w3      w4 w4

x:       x1
     x1 x1 x1
        x1

y:     y1          y2          y3        y4 y4 y4 y4     y5        y6 y6 y6 y6      y7        y8
   y1 y1   y2 y2 y2 y2        y3            y4           y5 y5          y6          y7    y8 y8 y8 y8
       y1                     y3 y3                      y5                    y7 y7
       y1                     y3                         y5                       y7

z:   z1 z1                z2       z3 z3      z4
        z1          z2 z2 z2       z3         z4 z4 z4
        z1 z1       z2          z3 z3              z4
```

Separei o EP em objetos específicos. Para a manipulação dos dados de entrada, criei uma classe Input que é auxiliar e recebe o arquivo de texto com a matriz que devemos preencher com os pentaminós. Ela também manipula as impressões para a saída. Transformo essa matriz de inteiros em uma de char. Onde tem um espaço vazio (0) é preenchido com “A” e onde tem um espaço não utilizável (1) é preenchido com Z.
Também criei uma classe para implementar a pilha, com suas funções básicas.
A classe principal Pentamino contém as verificações, inserções, e o backtracking dos pentaminós na matriz.

Percorro a matriz inteira ponto a ponto, verificando quais casas estão vazias. Em cada ponto verifico se cabe ali uma das peças. Para essa verificação fiz numa função separada, especificando para cada caso o que é necessário fazer para aquela peça específica caber ali. Aqui verifico todas as possibilidades de rotações das peças também. Uso variáveis globais para a matriz e para as posições usadas de cada peça. Quando é possível inserir, essa peça é incluida na matriz de saida e retorna um ok para a função principal empilhar o pentaminó. Também contabilizo num vetor separado que aquela peça foi usada. A posição fica guardada nas variáveis globais. Então quando o backtracking for necessário, a posição de cada peça usada estará disponível para consulta. Os números de 1 a 8 seguem as posições de cada peça segundo os desenhos anteriores.

Se a peça não couber ali, sigo a ordem do vetor tentado todos pentaminós, um a um.

Garanto que no máximo o vetor dê uma volta completa, tentando usar todas e nenhuma tenha dado certo. Também confiro se todas as peças já foram utilizadas ou não. Então com essas duas informações, parto para o backtracking. Retiro a peça anterior, marco ela como não utilizada e pego a posição inicial dela no tabuleiro. Assim tento um novo encaixe com as que sobraram. Inclusive a mesma peça, porém com as posições que ainda não foram tentadas. Isso é garantido pela variável que guardou a última posição utilizada.

Verifico constantemente se as peças foram todas usadas e se concluí a montagem. Se sim, retorno o tabuleiro preenchido. Se chego no final do tabuleiro e não consegui preencher com as 12 peças retorno que não foi possível preencher aquele tabuleiro.