Universidade Federal Fluminense
Instituto de Computação
Departamento de Ciência da Computação
Programação de Computadores II
Professor: Leandro Augusto Frata Fernandes

2º Lista de Exercícios
Arrays e matrizes

1. Escreva uma classe **Estatistica** em Java que contenha métodos estáticos que recebam um *array* de inteiros, juntamente com o número de elementos, e calculem respectivamente:

a) a moda dos elementos no *array* (elemento mais freqüente).

b) a mediana dos elementos no *array* (elemento central).

c) a média.

2. Implemente uma classe em Java denominada **Ordenacao** que tenha dois métodos estáticos que implementam, respectivamente, os algoritmos de ordenação por Inserção Direta e Seleção Direta sobre *arrays* de instâncias de uma interface **Comparavel**.

3. Uma imagem em preto e branco, de tamanho m x n, pode ser representada por uma matriz cujos elementos assumem valores no conjunto {0,1}. Dado um padrão representado por uma matriz 3x3 também assumindo valores em {0,1}, escreva um programa que determine se o padrão existe ou não na imagem.

0	0	1	0	0	0	0	1	0
0	1	1	1	1	1	1	0	0
0	0	1	0	1	1	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0
0	0	0	0	0	1	0	0	0
1	1	1	0	1	0	1	0	0
1	0	1	0	0	0	0	0	0
1	1	1	0	0	1	1	1	0

0	1	0
1	1	1
0	1	0

4. Escreva um programa Java capaz de jogar o jogo da velha e que nunca perca.

5. Sudoku é um quebra-cabeça, cujo objetivo do jogo é preencher os números de 1 a 9 em cada uma das células vazias numa grade de 9×9, constituída por 3×3 subgrades chamadas regiões. Cada coluna, linha e região só pode ter um número de cada um dos números de 1 a 9. Exemplo de um jogo:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Imagem extraída de: Wikipédia

Faça um programa em Java que, dado um jogo Sudoku, representado por uma classe que contem uma matriz 9x9, verifica se o jogo está ou não correto.

6. Um número primo é qualquer inteiro maior que um que é igualmente divisível apenas por si mesmo e 1. O crivo de Eratóstenes é um método de localizar números primos. Ele opera como segue:

a) Crie um array boolean de tipo primitivo com todos os elementos inicializados como true. Os elementos do array com índices primos permanecerão true. Todos os outros elementos do array por fim são configurados como false.

b) Iniciando com o índice de array 2, determine se um dado elemento é true. Se for, faça um loop pelo restante do array e configure como false cada elemento cujo índice é um múltiplo do índice para o elemento com valor true. Então continue o processo com o próximo elemento com valor true. Para o índice de array 2, todos os elementos além do elemento 2 no array que tiverem índices múltiplos de 2 (índices 4,6,8, 10 etc.) serão configurados como false; para o índice de array 3, todos os elementos além do elemento 3 no array que tiverem índices múltiplos de 3 (índices 6, 9, 12, 15 etc.) serão configurados como false; e assim por diante.

Quando esse processo for concluído, os elementos de array que ainda forem true indicam que o índice é um número primo. Esses índices podem ser exibidos.

Escreva um aplicativo que utiliza um array de 1000 elementos para determinar e exibir os números primos entre 2 e 999. Ignore elementos de array 0 e 1.

7. Escreva um aplicativo para simular o lançamento de dois dados. O aplicativo deve utilizar um objeto da classe Random uma vez para lançar o primeiro dado e novamente para lançar o segundo dado. A soma dos dois valores deve então ser calculada. Cada dado pode mostrar um valor de inteiro de 1 a 6, portanto a soma dos valores vai variar de 2 a 12, com 7 sendo a soma mais freqüente e 2 e 12 sendo as somas menos freqüentes. A figura abaixo mostra as 36 possíveis combinações de dois dados. Seu aplicativo deve lançar o dado 36 mil vezes. Utilize um array unidimensional para contar o número de vezes que cada possível soma aparece. Exiba os resultados em formato tabular.

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

8. A linguagem Logo tornou famoso o conceito de gráficos de tartaruga. Imagine uma tartaruga mecânica que caminha no lugar sob o controle de um aplicativo Java. A tartaruga segura uma caneta em uma de duas posições, para cima ou para baixo. Enquanto a caneta está para baixo a tartaruga desenha formas à medida que se move, e enquanto a caneta esta para cima a tartaruga se move quase livremente sem escrever nada. Nesse problema, você simulará a operação da tartaruga e criará um bloco de rascunho computadorizado.

Utilize um array de 20 por 20 denominado **floor** que é inicializado como zeros. Leia comandos a partir de um array que contenha esses comandos (determine um tamanho máximo para o array de entrada). Monitore a posição atual da tartaruga, se a caneta está atualmente para cima ou para baixo, e a direção da tartaruga. O comando "mover" ocorre na direção atual da tartaruga. Suponha que a tartaruga sempre inicia na posição (0,0) do chão com sua caneta para cima e apontando para a direita.

O conjunto de comandos de tartaruga que seu aplicativo deve processar é mostrado na tabela abaixo:

Comando	Significado
1	Caneta para cima
2	Caneta para baixo
3	Vira para a direita
4	Vira para a esquerda
5 X	Avance X espaços (substitua X pelo valor desejado)
6	Exiba o array 20 por 20
9	Fim dos dados (sentinela)

À medida que a tartaruga se move com a caneta por baixo, configure os elementos apropriados do array floor como 1s. Quando o comando 6 (exibir o array) for dado, onde quer que haja um 1 no array, exiba um asterisco ou o caractere que você escolher. Onde quer que haja um 0, exiba um espaço em branco.

9. Neste problema, você recriará a clássica corrida da tartaruga e da lebre. Você utilizará geração de números aleatórios para desenvolver uma simulação desse memorável evento.

Nossos competidores começam a corrida no quadrado 1 de 70 quadrados. Cada quadrado representa uma possível posição ao longo do percurso da competição. A linha de chegada está no quadrado 70. O primeiro competidor a alcançar ou passar o quadrado 70 é recompensado com um cesto de cenouras frescas e alface. O percurso envolve subir uma montanha escorregadia, então ocasionalmente os competidores perdem terreno.

Um relógio emite um tique por segundo. A cada tique do relógio, seu aplicativo deve ajustar a posição dos animais de acordo com as regras na tabela abaixo. Utilize variáveis para monitorar a posição dos animais (isto é, os números de posição são 1-70). Inicie cada animal na posição 1 (a 'partida'). Se um animal escorregar para a esquerda do quadrado 1, mova-o novamente para o quadrado 1.

Gere as porcentagens na tabela produzindo um inteiro aleatório *i* no intervalo 1 <= *i* <= 10. Para a tartaruga, realize uma 'caminhada rápida' quando 1 <= i <= 5, um 'escorregão' quando 6 <= *i* <= 7 ou uma 'caminhada lenta' quando 8 <= *i* <= 10. Utilize lima técnica semelhante para mover a lebre. Comece a corrida exibindo: BANG !!!!! A CORRIDA COMEÇOU!!!!!

Então, para cada toque do relógio (isto é, para cada repetição de um loop), exiba uma linha de 70 posições mostrando a letra T na posição da tartaruga e a letra L na

posição da lebre. Ocasionalmente, os competidores aterrissarão no mesmo quadrado. Nesse caso, a tartaruga morde a lebre e seu aplicativo deve exibir AI!!! começando nessa posição. Todas as outras posições da saída diferentes de T, L ou AI!!! (no caso de um empate) devem estar em branco. Represente a pista de corrida como um array de caracteres onde cada célula pode assumir os valores T ou L, para representar os animais, AI!!! nas células em seqüência, para representar a presença dos animais na mesma célula, ou espaço vazio no restante da pista.

Depois de cada linha ser exibida, teste se o animal alcançou ou passou o quadrado 70. Se tiver alcançado. exiba o vencedor e termine a simulação. Se a tartaruga ganhar, exiba A TARTARUGA VENCEU!!! EH!!! Se a lebre ganhar, exiba A LEBRE GANHOU. OH! Se as duas ganharem na mesma hora, você pode querer favorecer a tartaruga (a 'coitadinha') ou pode querer exibir OCORREU UM EMPATE. Se nenhum animal ganhar, realize o loop novamente para simular o próximo tique do relógio. Quando você estiver pronto para executar seu aplicativo, monte um grupo de fãs para observar a corrida. Você se surpreendera com a empolgação da sua audiência!

Animal	Tipo de Movimento	Porcentagem do Tempo	Movimento Real
Tartaruga	Caminhada Rápida	50 %	3 quadrados à direita
	Escorregão	20 %	6 quadrados à esquerda
	Caminhada Lenta	30 %	1 quadrado à direita
Lebre	Dormir	20 %	Nenhum movimento
	Salto Grande	20 %	9 quadrados à direita
	Escorregão Grande	10 %	12 quadrados à esquerda
	Salto Pequeno	30 %	1 quadrado à direita
	Escorregão Pequeno	20 %	2 quadrados à esquerda

Dica: Caso queira reduzir a velocidade do seu programa para tornar a corrida mais emocionante, utilize o seguinte trecho de código.

```
try {
        Thread.sleep(1000);
} catch (Exception e) {
        e.printStackTrace();
}
```

10. Utilize uma matriz quadrada de tamanho *n* para construir as *n* primeiras linhas do Triângulo de Pascal. Imprima esta matriz.