Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP1 2° semestre de 2012.

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

Questão 1) (4.0 pontos)

Escreva um programa que receba um inteiro N (N deve ser ÍMPAR) como parâmetro de entrada e imprima uma matriz mágica N por N. Uma matriz é considerada mágica quando:

- contém todos os inteiros entre 1 e N^2, sendo que ESTES NÚMEROS SÓ APARECEM APENAS UMA VEZ; e
- o valor da soma de cada uma das linhas, de cada uma das colunas e da diagonal principal é sempre o mesmo.

Uma forma de gerar uma matriz mágica é atribuir os números inteiros de 1 a N^2 em ordem crescente, começando na parte inferior da matriz, na célula do meio.

Repetidamente, seu algoritmo deve atribuir o próximo número inteiro a posição a direita e abaixo da atual (CUIDADO com os limites da matriz. Se o elemento atual está na última linha, o próximo elemento passa para a primeira linha. Se o elemento atual está na última coluna, o próximo elemento passa para a primeira coluna).

Por fim, se a posição já tiver sido ocupada por outro inteiro menor que o atual, tente usar a célula acima da atual (na mesma coluna).

Seguem alguns exemplos de execução deste código:

```
java Q1_AP1_2012_2 2
ERRO: N deve ser IMPAR
```

```
java Q1_AP1_2012_2 1
1

java Q1_AP1_2012_2 3
4  9  2
3  5  7
8  1  6

java Q1_AP1_2012_1 5
11 18 25  2  9
10 12 19 21  3
4   6 13 20 22
23  5  7 14 16
17 24  1  8 15
```

RESPOSTA:

```
public class Q1_AP1_2012_2 {
  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);
    if (N % 2 == 0) System.out.println("ERRO: N deve ser IMPAR");
    else{
      int[][] magica = new int[N][N];
      int l = N-1;
      int c = N/2;
      magica[l][c] = 1;
      for(int i = 2; i <= N*N; i++) {
        if(magica[(l + 1) % N][(c + 1) % N] == 0) { //achar a posição
          l = (l + 1) % N;
          c = (c + 1) % N;
        }else{
          l = (l - 1 + N) % N; // Posição ocupada: não mudar a coluna
        }
        magica[l][c] = i;
      }
      // imprimir resultados
      for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
          if(magica[i][j] < 10)  System.out.print(" ");  // alinhar
          if(magica[i][j] < 100) System.out.print(" ");  // alinhar
          System.out.print(magica[i][j] + " ");
        }
        System.out.println();
      }
    }
  }
}
```

Questão 2) (3.0 pontos)

Suponha que você tenha que implementar um sistema para controle de aplicações financeiras. Inicialmente foi identificado que uma aplicação contém um saldo inicial, um saldo acumulado e uma data de criação da aplicação. Uma aplicação pode ser um fundo de investimento ou uma poupança. Um fundo de investimento possui um campo que armazena o percentual de recolhimento de Imposto de Renda e outro que contém a taxa de administração cobrada pelo banco para o fundo escolhido. Uma poupança contém um campo que representa a taxa de juros desta aplicação, a qual tem valor único para todas as contas deste tipo. Além disso, é necessário que se possa obter a periodicidade de cada aplicação, ou seja, se o rendimento desta ocorre diariamente (valor do campo igual a 1) ou mensalmente (valor igual a 30). Modele este sistema criando classes e utilizando os conceitos de OO sempre que possível. Quanto a periodicidade, defina-a como um método nas classes criadas, sabendo que poupança tem periodicidade 30 e fundo tem periodicidade 1. Obrigue que qualquer tipo novo de aplicação a ser incorporada ao sistema tenha que definir uma versão deste método.

RESPOSTA:

```
/*
 * Critérios gerais de correção: O objetivo principal deste cabecalho eh apenas
 *delinear os aspectos principais que serao considerados durante a correcao.
 *Entretanto estes nao sao fechados, ou seja, podem haver solucoes alternativas
 *ou erros conceituais nao previstos neste gabarito
 *
 * - Questão vale 3 pontos
 * - Cada classe vale 1 ponto
 * - Nesta questão foi exigido que o aluno soubesse utilizar interfaces ou classes
abstratas. Como isso ainda não foi apresentado, este item será desconsiderado.
 * - Erro na herança (- 0.5 pontos)
 * - Erro na definição de atributo estático da classe Poupança (- 0.2 pontos)
 * - Criação redundante de campos (- 0.5 pontos)
 * - Não definição dos construtores (-0.5 pontos)
 */

import java.util.GregorianCalendar;

abstract class Aplicacao {
        float saldoInicial;
        float saldoAcumulado;
        GregorianCalendar inicio;

        public Aplicacao(float saldoInicial, float saldoAcumulado,
GregorianCalendar inicio) {
                this.saldoInicial = saldoInicial;
                this.saldoAcumulado = saldoAcumulado;
                this.inicio = inicio;
        }
```

```
        abstract int periodicidade();
}

class FundoInvestimento extends Aplicacao {
        float percentualIR;
        float taxaAdministracao;

        public FundoInvestimento(float saldoInicial, float saldoAcumulado,
                        float percentualIR, float taxaAdministracao,
GregorianCalendar inicio) {
                super(saldoInicial, saldoAcumulado, inicio);
                this.percentualIR = percentualIR;
                this.taxaAdministracao = taxaAdministracao;
        }

        int periodicidade() {
                return 1;
        }
}

class Poupanca extends Aplicacao {
        static float juros;

        public Poupanca(float saldoInicial, float saldoAcumulado,
GregorianCalendar inicio) {
                super(saldoInicial, saldoAcumulado, inicio);
        }

        int periodicidade() {
                return 30;
        }
}
```

Questão 3) (3.0 pontos)

```
class Veiculo {
        private int numRodas;
        protected String marca;
        protected int velocidade;

        protected void aumentarVelocidade(int vel) { velocidade = velocidade + vel; };
        protected void diminuirVelocidade(int vel) { velocidade = velocidade - vel; };
        public String getMarca() { return marca; };
        public int getVelocidade() { return velocidade; };
        public void acelerar() { aumentarVelocidade(5); };
        public void frear() { diminuirVelocidade(5); };
}
```

a) Quais membros (métodos e atributos) são acessados por uma subclasse de Carro?
b) Defina um construtor da classe que receba a marca e o número de rodas do carro.
c) Defina a subclasse Carro, cujos objetos possuem uma quantidade de lugares, sua

velocidade é no máximo 150 km/h e sempre possui 4 rodas. Redefina os métodos e construtores necessários e suas implementações.

d) Defina um programa principal que cria um carro da marca Fiat, com 6 rodas e um carro de passeio da Ford.

RESPOSTA:

```
/*
 * Critérios gerais de correção: O objetivo principal deste cabecalho eh apenas
 *delinear os aspectos principais que serao considerados durante a correcao.
 *Entretanto estes nao sao fechados, ou seja, podem haver solucoes alternativas
 *ou erros conceituais nao previstos neste gabarito
 *
 * - Questão vale 3 pontos
 * - Como divulgado pela plataforma, o item a) desta questão foi anulado. Com
isso, cada item desta passa a valer 1 ponto.
*/

class Veiculo {
        private int numRodas;
        protected String marca;
        protected int velocidade;

        protected void aumentarVelocidade(int vel) { velocidade = velocidade + vel;
};
        protected void diminuirVelocidade(int vel) { velocidade = velocidade - vel; };
        public String getMarca() { return marca; };
        public int getVelocidade() { return velocidade; };
        public void acelerar() { aumentarVelocidade(5); };
        public void frear() { diminuirVelocidade(5); };

        public Veiculo(String m, int n) {
                marca = m;
                numRodas = n;
        }
}

class Carro extends Veiculo {
        int qtdLugares;

        public Carro (String m) {
                super(m, 4);
        }

        protected void aumentarVelocidade(int vel) {
                if ((velocidade + vel) < 150)
                        super.aumentarVelocidade(vel);
        }
```

```
}

public class AP1_2012_2_Q3 {
	public static void main(String[] args) {
		Veiculo fiat = new Veiculo("FIAT", 6);
		Carro ford = new Carro("Ford");
	}
}
```