Lista de exercícios 5

Programação Orientada à Objetos

Todos exercícios deverão ser entregues no Moodle, em um único arquivo comprimido (.rar de preferência). Os códigos deverão seguir um estilo de código padrão e deverão estar comentados.

1) Modifique o exercício da lista 2 (já modificado na lista 3), o exercício da lista 4 (o primeiro) e a prova, considerando interfaces, classes e métodos abstratos. Reflita quando usar interface e quando usar classes e métodos abstratos. Veja se existe um caso de método final e implemente.

2) No exercício da lista 4, adicione o tipo do cartão (normal, internacional, ouro, black e platinum). Além disso, indique o tipo do cliente (padrão ou vip). Para isso use enumerações.

3) Leia toda a questão antes de começar a fazê-la. Considere um jogo de simulação (estilo SIMS) com objetos sobre um mapa bidimensional. O mapa deverá ser um array bidimensional de 100 posições por 100 posições. Cada posição do mapa deverá indicar se existe terra ou água (simulando por exemplo rios, mares ou oceanos). O mapa deverá ser da seguinte forma:

onde todos os pontos são terra, exceto por uma faixa (simulando um rio) que vai de y = 45 até y = 55 e se extende por todos pontos x (como mostrado na figura).

a) Crie uma classe Posição que contém apenas um inteiro x e um inteiro y indicando a posição de um objeto.

b) Crie classes para os seguintes ‘objetos’: Animal, Peixe, Tigre, Pássaro, Pessoa, Veículo, Avião, Carro, Barco e Pedra. Organize essas classes usando herança da forma mais coerente. A posição dos objetos não poderá ser visível fora da classe, a não ser em classes que herdem dela e também não devem ser criados setters para as posições. Não deverá ser possível criar objetos da classe Animal, nem da classe Veículo.

c) Crie 4 interfaces: Móvel, Aéreo, Aquático, Terrestre. Móvel não precisa ter nenhum método. Aéreo tem o método:

void voa(Posição posDestino)

que muda a posição do objeto e que escreve na tela:

“Avião decolando em <posição origem>, voando e aterrisando em <posição destino>”

“Pássaro voando de <posição origem> para <posição destino>”

Aquático tem o método:

void nada(Posição posDestino)

que muda a posição do objeto e escreve na tela:

“<nome da classe> nadando de <posição origem> para <posição destino>”,

no entanto, se a distância da posição origem para a posição destino for:

maior que 5 para a classe Pessoa, deve-se imprimir na tela “Pessoa se afogou”.

Maior que 50 para a classe Barco, deve-se imprimir na tela “Acabou o combustível do Barco”. Essas informações de distância máxima deverão ser armazenadas de tal forma que sejam definidas na classe, e sejam constantes durante toda execução.

Terrestre tem o método:

void anda(Posição posDestino)

que muda a posição do objeto e imprime

“<nome da classe> andando de <posição origem> para <posição destino>”.

d) Crie uma classe concreta InterfaceGráfica. Essa classe deverá ter um método

void moveObjeto(Movel movel, Posição posDestino)

que movimenta o objeto movel para a posição posDestino. Esse método deverá fazer o objeto se mover em linha reta até o destino, se for possível, ou até a máxima distância em linha reta que o objeto pode chegar considerando a natureza do movimento do objeto. Alguns objetos não se movem (logo não podem ser passados como parâmetro para esse método, outros andam (podem se movimentar na terra), outros nadam (podem se movimentar na água) e outros voam (podem se movimentar por tudo, mas não podem pousar na água. Esse método não pode fazer

e) Crie esse mapa no construtor da classe InterfaceGráfica. O tipo do mapa deverá ser uma enumeração que indica se naquela posição existe terra ou água (de acordo com a especificação acima).

f) Os veículos deverão ter um registro de todos os trajetos que já fizeram. Para isso crie um método registra(Posição origem, Posição destino) que registra essa informação em um array. Esse método deverá pertencer à classe Veículo e não poderá ser sobrescrito por classes filhas.

g) Armazene em algum lugar a quantidade de animais existentes, por tipo. Não armazene essa informação na main, e não crie nenhuma classe nova para armazenar isso. Essa informação deverá estar atualizada e considerar todas unidades criadas menos as pessoas que morreram afogadas.