Simulado 1 Linguagem Orientada a Objetos

Nome:

Questão 1

Um desenvolvedor precisa criar um sistema de cadastro de animais para um pet shop. Ele inicia definindo a classe `Animal` com atributos `nome` e `idade`. Entretanto, o gerente pede que cada animal tenha também a capacidade de emitir um som específico. Qual seria a melhor solução em POO?

a) Adicionar o som como atributo fixo da classe `Animal`.

b) Criar um método `emitirSom()` dentro da classe `Animal`.

c) Criar uma variável global `som` fora da classe e acessá-la.

d) Utilizar herança múltipla para cada animal.

e) Criar uma classe `Som` sem relação com `Animal`.

Questão 2

Durante a execução de um projeto no JDoodle, um estudante percebe que cada vez que roda o programa precisa reconfigurar atributos manualmente. Como aplicar melhor o conceito de encapsulamento para evitar esse problema?

a) Declarar todos os atributos como `public`.

b) Definir os atributos como `private` e criar métodos `get` e `set`.

c) Inicializar variáveis no método `main()`.

d) Criar um construtor vazio sem parâmetros.

e) Usar apenas variáveis locais em vez de atributos.

Questão 3

Uma equipe está desenvolvendo um sistema acadêmico e deseja representar a relação entre um `Aluno` e seu `ProfessorOrientador`. Qual conceito de POO é mais adequado para representar essa relação?

a) Herança.

b) Polimorfismo.

c) Associação (composição ou agregação).

d) Encapsulamento.

e) Sobrecarga de métodos.

Questão 4

Durante testes, o método `emitirSom()` da classe `Animal` imprime sempre o mesmo texto, independentemente do tipo de animal. O professor orienta que cada subclasse (`Cachorro`, `Gato`, etc.) deve sobrescrever esse método. Esse princípio é chamado de:

a) Herança simples.

b) Abstração.
c) Polimorfismo.
d) Modularização.
e) Overload de atributos.

Questão 5
Um aluno está iniciando no JDoodle e escolhe a opção de “projeto vazio”. Entretanto, esquece de selecionar a versão correta do JDK e enfrenta incompatibilidades com novos recursos da linguagem. Qual seria a configuração recomendada?
a) Usar JDK 1.5 para máxima compatibilidade.
b) Escolher sempre a versão mais antiga.
c) Selecionar JDK 21.0.0, estável e atualizado.
d) Criar o código sem dependência de versão.
e) Alterar para linguagem C++ no compilador.

Questão 6
Um professor deseja ensinar laços de repetição usando Alice. Ele cria uma cena em que um personagem deve pular 5 vezes. Qual procedimento representa corretamente o conceito de loop?
a) Incluir manualmente 5 instruções de pulo.
b) Utilizar um `if` para verificar condição de pulo.
c) Arrastar a estrutura de repetição com contador de 5.
d) Criar 5 objetos iguais para simular o movimento.
e) Usar uma função `return` em vez de loop.

Questão 7
Um estudante cria um cenário no Alice com vários objetos, mas todos surgem sobrepostos. Para corrigir o problema, ele deve:
a) Aumentar o tamanho do cenário.
b) Alterar a cor de fundo.
c) Reconfigurar as coordenadas X, Y, Z de cada objeto.
d) Executar o programa e reposicionar manualmente.
e) Alterar a aba de Funções Lógicas.

Questão 8
Um avatar humano em Alice deve cumprimentar o usuário e convidá-lo para calcular o fatorial de um número. Para garantir clareza da mensagem, o estudante deve configurar:
a) A função `return`.
b) Caixa de diálogo com tempo de exibição definido.
c) O atributo de cor do cenário.

d) Apenas o áudio do personagem.
e) A posição inicial do objeto humano.

Questão 9
Durante uma animação, um estudante deseja que o personagem se mova até a estante e pegue um livro. Para isso, a melhor abordagem é:
a) Utilizar variáveis globais.
b) Criar um novo avatar.
c) Definir um procedimento que combine movimentação e interação.
d) Usar apenas a aba “Funções Lógicas”.
e) Alterar o código Java diretamente.

Questão 10
Uma falha comum em projetos com Alice é a ausência de planejamento prévio da animação, resultando em inconsistências. Qual prática evita esse problema?
a) Criar objetos aleatórios.
b) Usar exclusivamente loops.
c) Definir um roteiro detalhado antes da programação.
d) Aumentar a resolução gráfica.
e) Configurar apenas cores e fundos.

Questão 11
Um aluno implementa um `player` em Greenfoot que deve responder a teclas de seta. Ele esquece de incluir a verificação de evento no método `act()`. O que ocorrerá?
a) O jogo funcionará normalmente.
b) O personagem se moverá sozinho.
c) O personagem não responderá às entradas do teclado.
d) O cenário se reiniciará automaticamente.
e) O compilador impedirá a execução.

Questão 12
Um desenvolvedor deseja criar um `bot` que se mova em zigue-zague. Qual estratégia mais adequada em Greenfoot?
a) Implementar uma herança múltipla.
b) Alterar coordenadas X e Y em ciclos dentro do método `act()`.
c) Criar variáveis globais em `World`.
d) Definir um sprite estático.
e) Usar apenas imagens do OpenGameArt.

Questão 13

Uma equipe precisa criar um cenário que represente uma floresta no Greenfoot. Para isso, deve:

a) Criar objetos dentro do método `main()`.

b) Estender a classe `World` e carregar uma imagem de fundo.

c) Alterar apenas atributos do `Actor`.

d) Definir o cenário dentro de cada `bot`.

e) Usar apenas variáveis locais.

Questão 14

Durante testes, um `Actor` colide com outro mas não gera interação. O erro provável é:

a) Classe `World` não inicializada.

b) Sprite incompatível.

c) Ausência de método de detecção de colisão implementado.

d) Herança incorreta em `Actor`.

e) Uso de imagem PNG transparente.

Questão 15

Um aluno esquece de publicar o jogo final no site do Greenfoot. Qual impacto pedagógico essa falha pode ter?

a) O jogo não rodará localmente.

b) A turma perderá a oportunidade de troca de experiências e feedback.

c) O compilador apresentará erro de sintaxe.

d) A classe `Actor` ficará corrompida.

e) O método `act()` não será executado.

Questão 16

Um jogador controla um personagem que deve coletar moedas no Greenfoot. Entretanto, o código não atualiza a pontuação. Qual seria a correção mais adequada?

a) Alterar a classe `World`.

b) Adicionar uma variável de score e atualizar dentro da colisão.

c) Mover a lógica para o método `main()`.

d) Criar uma nova classe de cenário.

e) Alterar apenas a imagem do objeto.

Questão 17

Um estudante cria um jogo “Pega o Objeto”, mas não limita o número de itens no cenário, causando sobrecarga. Qual prática resolveria o problema?

a) Aumentar o tamanho da janela.

b) Definir condições para geração controlada de objetos.
c) Alterar a imagem de fundo.
d) Desativar colisões.
e) Usar apenas métodos estáticos.

Questão 18
Durante a execução do jogo, o personagem principal não responde ao teclado. Isso ocorre porque:
a) O sprite é incompatível.
b) O método `act()` não contém tratamento de entrada.
c) O método `main()` não foi implementado.
d) O `World` está vazio.
e) As imagens estão corrompidas.

Questão 19
Um aluno deseja que, ao coletar 10 moedas, o jogo exiba uma mensagem de vitória. Qual seria a melhor implementação?
a) Criar um novo objeto “Vitória”.
b) Alterar o sprite do player.
c) Adicionar verificação no score e exibir mensagem no `World`.
d) Reiniciar o método `main()`.
e) Usar variável global em `Actor`.

Questão 20
No desenvolvimento de jogos educacionais, qual a principal vantagem de usar Greenfoot em vez de programação puramente textual?
a) Exige menos memória RAM.
b) Permite aprendizado visual e interativo dos conceitos de POO.
c) Dispensa lógica de programação.
d) Substitui completamente linguagens como Java.
e) Elimina a necessidade de herança.