1 - Qual das opções a seguir contém uma declaração de classe válida em linguagem Java?

R: "class Aluno { }".

Comentário: Na declaração de uma classe, o modificador *"public"* é opcional e o único permitido.

2 - Sobre objetos em Java, é correto apenas o que se afirma em:

R: O programador não precisa se preocupar em desalocar a memória de um objeto destruído.

Comentário: A reciclagem de espaço de memória em Java é feita pelo coletor de lixo.

3 - Sobre herança em Java, é correto afirmar apenas que:

R: Um objeto instanciado da subclasse é também um objeto do tipo da superclasse.

Comentário: O mecanismo de herança dá subclasse à mesma estrutura da superclasse.

4 - Em um software Java, uma classe chamada "Painel" tem a classe derivada "LCD", que sobrecarrega um método "acender" de "Painel". O método é protegido em ambas as classes. A única opção que possui uma afirmativa correta é:

R: Trata-se de um caso de polimorfismo.

Comentário: LCD herdará o método "acender" de "Painel", ficando com duas versões, o que caracteriza um tipo de polimorfismo.

5 - Avalie as seguintes afirmações feitas acerca da linguagem Java:

I. Ao utilizar o método "*groupingBy*" da classe "*Collectors*", o programador tem de informar o atributo a ser usado para o agrupamento.

II. Os objetos agrupados são armazenados em um container que é mapeado para a chave de agrupamento.

III. O método "*groupingBy*" só armazena os objetos em coleções do tipo "*List*".

Está correto apenas o que se afirma em:

R: II

Comentário: O retorno do método "*groupingBy*" é um "*Collector*" que cria uma estrutura "*Map*". Essa estrutura mantém um mapeamento entre a chave de agrupamento e o container que contém os objetos agrupados. As demais afirmativas são falsas.

6 - Escolha a única alternativa verdadeira:

R: Uma pilha pode ser implementada com o container "*Deque*".

Comentário: O container "*Deque*" possibilita inserção e remoção em ambas as extremidades, o que permite implementar uma pilha.

7 - A única assinatura de método que cria um ponto de entrada para a execução de um programa Java é:

R: "public static void main (String args[ ])".

Comentário: O método "*main*" em Java não admite sobrecarga. Sua assinatura é fixa, inclusive quanto aos parâmetros.

8 - Para realizar um desenvolvimento em Java, são imprescindíveis todos os itens da alternativa:

R: JDK e Editor de Código.

Comentário: O JDK contém o JRE e a Máquina Virtual Java, mas não possui aplicativo de edição de código, que precisa ser complementado.

9 - Considere duas classes chamadas "Base" e "Derivada", de tal forma que a primeira é superclasse da última. A classe "derivada" atende ao princípio da substituição de Liskov.

Analise as afirmações a seguir e marque a opção em que todas são corretas.

Derivada não pode sobrescrever os métodos protegidos de "Base".

II) Todos os métodos públicos e protegidos de “Derivada” devem ter a mesma assinatura que os de “Base”.

III) Todos os métodos públicos e protegidos de “Base” que forem redefinidos em “Derivada” devem ter a mesma assinatura.

R: III

Comentário: O princípio da substituição de Liskov afirma que a classe “Base” deve poder ser substituída por “Derivada” sem que o programa quebre. Assim, é necessário que os métodos de “Base” redefinidos por “Derivada” tenham a mesma assinatura.

10 - Estruturas de dados são mecanismos fundamentais para a manipulação de dados e possuem impacto na performance de um software. Java fornece implementações de algumas estruturas por meio do Java Collections Framework. Sobre o assunto, marque a única opção correta.

R: Mais de um tipo de coleção permite a implementação de fila (FIFO).

Comentário: O *Java Collections Framework* permite que algumas estruturas de dados sejam implementadas de forma simples e rápidas, um exemplo são as filas que podem ser implementadas por “*Queue*” e “*Deque*”.

11 - Sobre os métodos “equals” e “hashCode” da classe “Objects”, podemos afirmar que:

R: Objetos iguais terem código *hash* distintos viola o contrato geral de “*hashCode*”.

Comentário: Na linguagem Java, o contrato geral do método “*hashCode*” afirma que objetos iguais devem ter códigos *hash* iguais”.

12 - Um dos métodos da classe “Objects” é o “toString”, que retorna uma representação textual do objeto. Sobre esse método, são feitas as seguintes afirmações:

I) São informações obrigatórias, mesmo se ele for redefinido, o nome completamente qualificado do objeto e seu código hash.

II) Ele pode ser invocado em um objeto do tipo String.

III) O código hash informado por “toString” é um identificador único do objeto.

Marque a opção que contém a afirmativa correta:

R: II.

Comentário: Em Java, todos os objetos descendem direta ou indiretamente de “*Objects*”, logo, herdam a implementação padrão de “*toString*”. Isso se dá, inclusive, com um objeto do tipo “*String*”.

13 – Sobre interfaces em Java, é correto afirmar que:

R: Em uma interface não é necessário declarar os métodos como abstratos.

Comentário: A finalidade de uma interface é prover métodos abstratos que formam um contrato a ser seguido e cuja implementação é oculta. Ela é formada por métodos implicitamente abstratos.

14 – Suponha que um programa em Java possua a interface “iContabil”, que é superinterface de “iBalanco”. “iBalanco” é implementada pela classe concreta “Balanco”. Julgue as afirmativas:

I) Podemos usar uma variável do tipo “iContabil” para referenciar um objeto do tipo “Pessoa” e, nesse caso, teremos acesso aos métodos de “iContabil” e “iBalanco”.

II) Não é possível usar uma variável do tipo da superinterface para referenciar um objeto da classe que implementa “iBalanco”.

III) “Balanco” deverá implementar todos os métodos abstratos de “iBalanco” e os de “iContabil” que não forem ocultados por “iBalanco”.

A afirmativa correta é:

R: Somente III.

Comentário: Uma classe que implementa uma interface deve implementar todos os seus métodos abstratos. Os métodos abstratos da superinterface que não são ocultados pela subinterface são herdados por ela e, por isso, também devem ser implementados.

15 – Considere as afirmações a seguir.

I – Um desvio no fluxo principal de um programa é uma exceção.

II – Se uma exceção é lançada pela instrução "throw", então ela é uma exceção explícita.

III – Uma exceção explícita não precisa ser tratada localmente no método no qual é lançada.

É (são) verdadeira(s) apenas a:

R: III

Comentário: Uma exceção é um desvio não previsto no fluxo do programa. Desvios no fluxo principal para fluxos alternativos são condições normais e previstas e não são exceções. Além disso, uma exceção implícita também pode ser lançada pela instrução "throw". Mesmo uma instrução explícita pode ser lançada e propagada, não sendo obrigatório seu tratamento no método onde ocorre.

16 – Sobre exceções implícitas, é correto afirmar-se apenas que:

R: Exceções definidas na classe UnknownError são implícitas.

Comentário: Exceções são implícitas quando definidas nas classes Error e RuntimeException e suas derivadas. A classe UnknownError é uma subclasse de Error.

17 – Um programador criou seu conjunto de exceções por meio da extensão da classe Error, com a finalidade de tratar erros de socket em conexões com bancos de dados. O comando que pode ser empregado para garantir o fechamento correto da conexão, mesmo em caso de ocorrência de exceção, é:

R: finally

Comentário: A ocorrência de exceção causa um desvio não desejado no fluxo do programa, podendo impedir o encerramento da conexão e causando um vazamento de recurso. O comando finally, porém, é executado independentemente do desvio causado pela ocorrência de exceção, o que o torna adequado para a tarefa.

18 – Sobre o comando throw, são feitas as seguintes afirmações:

I – Não pode ser usado com o comando throws.

II – Só pode ser utilizado dentro do bloco try ou do bloco catch.

III – Pode ser utilizado para lançar exceções definidas nas subclasses de Error.

É correto apenas o que se afirma em:

R: III

Comentário: O comando throw pode ser usado para lançar exceções fora do bloco try-catch e pode ser usado em combinação com throws, cujo objetivo é notificar o chamador de que uma exceção pode ser lançada, mas não será tratada no local. As exceções derivadas de Error e suas subclasses são implícitas, mas throw pode ser usado para lançá-las manualmente.

19 - Sobre o relançamento de exceções em Java, assinale a única alternativa correta.

R: É feito dentro de um bloco catch e usando a referência da exceção capturada.

Comentário: O relançamento utiliza a referência da exceção capturada junto da instrução throw para lançar novamente essa exceção e é feito dentro de um bloco catch.

20 – Um programador criou uma exceção (NotReferencedException) a partir da classe RuntimeException. Considerando seus conhecimentos de tratamento de exceção em Java, marque a única alternativa correta quando

as exceções são empregadas em múltiplas cláusulas catch associadas ao mesmo bloco try para lidar com o lançamento de tipos diferentes de exceção pelo mesmo trecho de código.

R: NotReferencedException deve vir antes de RuntimeException.

Comentário: A classe NotReferencedException é uma subclasse de RuntimeException, logo, se RuntimeException estiver antes de NotReferencedException, a exceção será capturada por este bloco catch e nunca chegará ao bloco destinado a tratar exceções do tipo NotReferencedException.

21 – Threads em Java permitem o paralelismo de tarefas em uma aplicação. Sobre esse tema, são feitas as seguintes afirmações:

I Todas as aplicações cujas atividades podem ser subdivididas podem se beneficiar do uso de threads, reduzindo o tempo de resposta.

II Uma vez que a aplicação Java que faz uso de threads é executada pela MVJ, o desempenho será o mesmo, independentemente de a CPU ter um ou mais núcleos.

III Threads de usuário impedem as threads daemon de serem executadas.Assinale a opção que contém apenas afirmativa(s) correta(s):

R: I

Comentário: Aplicações cujas tarefas podem ser subdivididas permitem que cada subtarefa seja alocada a uma thread e conduzida em paralelo.

22 – Sobre threads em Java, marque a única opção correta:

R: Toda aplicação possui ao menos uma thread.

Comentário: Uma thread é uma linha de execução de programa. Mesmo quando o programador não usa threads, uma thread principal é criada e se torna a única thread em execução, praticamente confundindo-se com o processo da aplicação.

23 – Sobre a programação paralela em Java, marque a única alternativa correta:

R: Um método de uma classe não segura (*non thread safe*) pode ser invocado de forma segura com "synchronized".

Comentário: O modificador "synchronized" pode ser aplicado a um trecho de código. Nesse caso, ele admite como parâmetro a referência para o objeto e, em seu corpo, o método desse objeto que pode ser invocado de forma segura (thread safe).

24 – Sobre semáforos em Java, assinale a opção que contém apenas afirmativa(s) correta(s):

I) Um semáforo é um tipo de mutex.

II) Em um semáforo o número de bloqueios solicitado não precisa ser igual ao número de liberações.

III) Semáforos controlam se o objeto que libera o bloqueio é o mesmo que o solicitou.

R: II

Comentário: O semáforo controla o número de acessos disponíveis a um recurso ou uma região crítica. Uma vez esgotados, novas threads que solicitam acesso são colocadas em espera. Uma thread que tenha obtido acesso não precisa liberar o mesmo número de acessos obtidos. Se liberações parciais permitirem que uma thread em espera ganhe acesso, então ele é concedido.

25 – Considere o objeto "thd" instanciado a partir da classe "MinhaThd", que estende a classe thread. Qual opção mostra uma sequência que não gera erro de compilação?

R: thd.start (); thd.setName ("alfa"); thd.run (); thd.getId ();

Comentário: Como o objeto já foi instanciado, o nome da thread pode ser definido a qualquer momento, devendo ser passada uma “String”. Da mesma forma, o seu identificador pode ser obtido. Assim, a ordem que esses métodos aparecem não faz diferença. O método “run ()” aparece após a execução de “start ()”, que inicia a thread. Assim, todos estão semântica e ordenadamente corretos.

26 – obre a classe thread, é correto afirmar que:

R: “setPriority (Thread.MIN_PRIORITY - 1)” não causa erro de compilação.

Comentário: O método “setPriority” não desrespeita os limites definidos por MIN_PRIORITY e MAX_PRIORITY. Entretanto, a forma mostrada não incorre em erro de sintaxe. A violação do limite mínimo de prioridade só é detectada em tempo de execução. Por isso, essa linha compila e gera uma exceção durante a execução.

27 - Com o advento dos bancos de dados, tornou-se comum a construção de sistemas cadastrais que utilizam esse tipo de repositório. No entanto, há diversos fornecedores para sistemas de gerenciamento de bancos de dados, tornando-se uma boa prática, na construção de um sistema cadastral:

R: Utilizar SQL ANSI.

Comentário: O acesso a uma base de dados independe do tipo de sistema ou de chave primária, e a comunicação entre front-end e back-end deve ser feita por meio de um middleware. Com relação às mensagerias, elas permitem a comunicação assíncrona entre sistemas por meio de mensagens, não tendo relação com a base de dados. A única opção que trata de uma boa prática é a utilização de SQL ANSI, pois permite a troca de fornecedor do back-end sem grandes alterações no front-end, desde que seja utilizado um middleware.

28 – Quando criamos um sistema cadastral, diversos comandos SQL se repetem, com a simples mudança de valores para os campos. Uma estratégia muito interessante, que promove o reuso e simplifica a programação, é o uso de comandos parametrizados, os quais são viabilizados por meio de objetos do tipo:

R: PreparedStatement

Comentário: Os componentes do tipo PreparedStatement viabilizam comandos SQL parametrizados, em que pontos de interrogação definem a posição dos parâmetros em meio ao SQL, os quais devem ser preenchidos, a partir da posição de índice 1, antes da execução da instrução. Além de permitir uma implementação otimizada, torna o sistema mais seguro contra os ataques do tipo injection.

29 – O uso de comandos SQL dispersos, em meio ao código do aplicativo, diminui o reuso e aumenta a dificuldade de manutenção. Com base no padrão de desenvolvimento DAO, temos a concentração dos comandos SQL em uma única classe, em que existem métodos para o retorno de entidades, como obterTodos, que estão relacionados ao comando:

R: SELECT

Comentário: Na construção de uma classe DAO, precisamos minimamente dos métodos obterTodos, incluir, excluir e alterar, que estarão relacionados, respectivamente, aos comandos SELECT, INSERT, DELETE e UPDATE. Com base nesses métodos, temos a possibilidade de listar os registros, acrescentar um novo registro, alterar os dados do registro ou, ainda, remover um registro da base de dados.

30 - A adoção do padrão DAO abriu caminho para a construção de diversos frameworks de persistência, que simplificam muito as operações sobre a base de dados, eliminando a necessidade de utilização de comandos SQL. Entre as diversas opções do mercado, temos uma arquitetura de persistência denominada JPA, em que as entidades devem ser gerenciadas por uma classe do tipo:

R: EntityManager

Comentário: Os componentes do tipo EntityManager gerenciam as operações sobre as entidades do JPA, trazendo métodos como persist, para incluir um registro na base de dados, ou createQuery, para a obtenção de objetos Query, capazes de recuperar as entidades a partir da base. A função de EntityManagerFactory é a de gerar objetos EntityManager, enquanto Persistence faz a relação com as unidades de persistência.