UNIVERSIDADE FEDERAL DO MARANHÃO
DISCIPLINA: LINGUAGEM DE PROGRAMAÇÃO II
PROFESSOR: TIAGO BONINI BORCHARTT

2ª Prova

1. Quando se diz que uma classe “Cachorro” estende a classe “Animal”, em Programação Orientada a Objetos, estamos afirmando o mesmo que: (1.0 ponto)

☐ as classes são ditas como “irmãs”.

☐ a classe “Animal” é subclasse de “Cachorro”.

☐ a classe “Cachorro” deriva da classe “Animal”.

☐ a classe “Animal” é derivada de “Cachorro”.

☐ as classes são consideradas "primas".

2. Considerando os conceitos de Herança e Polimorfismo, presentes na linguagem orientada a objetos Java, é correto afirmar que: (1.0 ponto)

☐ uma desvantagem da herança analisando o reuso de código é que ela cria dependências entre classes em uma hierarquia.

☐ a herança não oferece uma solução para o problema de modificação oriundo do reuso de tipos abstratos de dados.

☐ o polimorfismo permite a existência de métodos na classe pai que não sejam visíveis na subclasse.

☐ os métodos de classe podem realizar operações somente na classe pai.

☐ as subclasses podem realizar a sobrecarga (overload) dos métodos da classe pai, alterando os parâmetros recebidos e/ou o retorno do método.

3. Uma grande contribuição da orientação a objetos para o projeto e desenvolvimento de sistemas é o polimorfismo. O polimorfismo está associado a dois conceitos fundamentais: a sobrecarga de métodos e a reescrita de métodos. Assim, associe estes conceitos, com as afirmações a seguir: (2.0 pontos)

	Sobrecarga de métodos	Reescrita de métodos
É a possibilidade de existirem vários métodos com o mesmo nome em uma mesma classe.	☐	☐
Permite implementar um método em uma subclasse que tenha o comportamento diferente do método na sua superclasse.	☐	☐

4. Em orientação a objetos, uma classe abstrata é: (1.0 ponto)

☐ um conceito, abstração ou coisa com limites e significados bem definidos para a aplicação em questão.

☐ um grupo de objetos com propriedades (atributos) similares, comportamento (operações) similares, relacionamentos comuns com outros objetos e uma semântica comum.

☐ uma forma de esconder os detalhes da implementação de um objeto.

☐ uma classe que não possui instâncias diretas, mas cujas classes descendentes, sim.

☐ um objeto usado para agrupar e gerenciar objetos relacionados.

5. Assinale apenas as 4 (quatro) alternativas corretas: (2.0 pontos)

☐ Para que a interface pública de uma classe seja considerada coesa, é necessário que todos os recursos dessa interface estejam relacionados ao conceito que a classe representa.

☐ A herança em programação orientada a objetos é um relacionamento pelo qual uma classe, chamada de subclasse, herda todos os comportamentos e estados possíveis de outra classe, chamada de superclasse ou classe base.

☐ Se o construtor da subclasse não chamar explicitamente um construtor da superclasse, então a superclasse usa seu construtor default, isto é, sem argumentos. Se a superclasse não tiver construtor default e o construtor da subclasse não chamar explicitamente nenhum construtor, o próprio Java se encarrega de gerar, em tempo de execução, um construtor default da superclasse.

☐ Acerca do conceito de tratamento de exceções, o bloco try sempre será executado por completo e o bloco catch só será executado se houver o disparo de uma exceção.

☐ Se uma subclasse herdar características de duas ou mais superclasses, ocorrerá uma herança múltipla.

☐ Um objeto é uma instância de uma classe.

☐ A sobrecarga de atributos e operações permite que uma subclasse herde funcionalidades da superclasse, mas modifique os atributos e operações herdados de modo a adaptá-los a necessidades específicas da subclasse.

☐ A visibilidade de um membro de uma classe pode ser privada, pública ou protegida. Um atributo privado só pode ser acessado por métodos privados. Um atributo público só pode ser acessado por métodos públicos. Um atributo protegido só pode ser acessado por métodos protegidos.

☐ Uma classe que implementa uma interface compromete-se a prover o comportamento publicado por aquela interface.

☐ A estrutura try...catch...finally da linguagem Java tem o objetivo de controlar o fluxo de execução do programa, permitindo de maneira robusta que o programador trate possíveis situações de erro, que poderiam interromper a execução da aplicação.