1. R: Uma classe abstrata é uma classe que não pode ser instanciada diretamente e serve como um modelo para outras classes.
Ela pode ser útil quando possuímos múltiplas classes que tem caraterísticas parecidas, mas necessitam implementações diferentes.

2. R: Herança é um princípio que permite que uma classe (**subclasse** ou **classe derivada**) herde atributos e métodos de outra classe (**superclasse** ou **classe base**).
É útil quando diferentes classes possuem características parecidas.

3. R: **Superclasse (Classe Base ou Pai):** É a classe mais genérica que define atributos e comportamentos comuns.
Subclasse (Classe Derivada ou Filha): É a classe que **herda** da superclasse e pode adicionar novos atributos, métodos ou até sobrescrever comportamentos.

4. R: Via acesso direto, pode acessa-los se os atributos forem public ou protected, com o uso de super e com assessores como: getters e setters, quando possuírem o estado private.

5. R: Não, pois o uso do extends traz todos os atributos e métodos que não estejam no estado "private".

6. R: Não podem ser instanciadas, podem ter métodos abstratos e concretos, podem ter atributos (inclusive com inicialização), construtores em classes abstratas, * herança: podem estender outras classes e implementar interfaces, obrigatoriedade de implementação, permite modificadores de acesso variados.

7. R: Se a classe possuir a declaração de abstract, sim, do contrário deverá implementar todos os métodos.

8. R: Em Java, todas as classes definidas pelo usuário herdam, direta ou indiretamente, da classe Object, que é a superclasse raiz da hierarquia de classes. Isso significa que * qualquer classe criada em Java é, por padrão, uma subclasse de Object, mesmo que você não use explicitamente a palavra-chave extends.

9. R: A classe Object em Java é a superclasse de todas as classes, ou seja, qualquer classe criada automaticamente herda seus métodos. Esses métodos fornecem funcionalidades básicas * que podem ser utilizadas ou sobrescritas de acordo com a necessidade da aplicação.

Método	Funçao	Comum sobrescrever?
toString()	Representação em texto do objeto	Sim
equals()	Compara objetos para igualdade	Sim
hashCode()	Código hash para coleções	Sim (se sobrescrever `equals`)
getClass()	Retorna a classe do objeto	Não
clone()	Cria uma cópia do objeto	Às vezes
finalize()	Executado antes da coleta de lixo	Não recomendado
notify()/wait()	Controle de threads	Avançado

10. R:

Sobrescrita de Métodos (Override):

A sobrescrita de métodos ocorre quando uma subclasse fornece uma nova implementação para um método que já existe na sua superclasse.

Requisitos da Sobrescrita: A assinatura do método na subclasse deve ser idêntica à da superclasse. O tipo de retorno do método sobrescrito deve ser o mesmo (ou um tipo mais específico).

O modificador de acesso do método sobrescrito na subclasse pode ser mais permissivo, mas não mais restritivo.

Sobrecarga de Métodos (Overload):

A sobrecarga de métodos ocorre quando há métodos com o mesmo nome, mas com assinaturas diferentes (diferente número, tipo ou ordem dos parâmetros) dentro da mesma classe.

Objetivo: Criar múltiplos métodos que realizam ações semelhantes, mas que podem ser usados com diferentes tipos ou números de parâmetros.

Exemplo de uso: Oferecer a possibilidade de um método funcionar com diferentes tipos de dados de entrada.

Requisitos da Sobrecarga: O nome do método deve ser o mesmo. A assinatura do método (número e tipo de parâmetros) deve ser diferente. O tipo de retorno pode ser diferente, mas não é suficiente para a sobrecarga.