1. O que é serialização?

Resposta:
É o processo de converter o estado de um objeto em uma representação binária ou textual para armazenamento.

---

2. O que é desserialização?

Resposta:
É o processo inverso: reconstruir um objeto a partir de sua representação binária armazenada.

---

3. Qual interface deve ser implementada para permitir a serialização de objetos?

Resposta:
A interface `Serializable`.

---

4. Que tipo de interface é Serializable?

Resposta:
Uma interface de marcação (sem métodos).

---

5. Qual método principal da classe ObjectOutputStream grava objetos?

Resposta:
`writeObject(Object obj)`.

---

6. O que é necessário para um objeto ser serializado?

Resposta:
A classe do objeto deve implementar Serializable.

---

7. O que acontece com variáveis estáticas durante a serialização?

Resposta:
Elas **não** são serializadas.

---

8. Para que serve o modificador transient?

Resposta:
Para indicar que um atributo não deve ser serializado.

---

9. É possível gravar vários objetos em um mesmo stream?

Resposta:
Sim, ObjectOutputStream permite múltiplos objetos.

---

10. O que ocorre se tentarmos serializar uma classe que não implementa Serializable?

Resposta:
É lançada a exceção `NotSerializableException`.

---

11. Qual exceção ocorre se tentarmos desserializar uma classe desconhecida no sistema?

Resposta:
`ClassNotFoundException`.

---

12. Por que a ordem de leitura deve ser igual à ordem de escrita?

Resposta:
Porque o ObjectInputStream lê os objetos sequencialmente conforme gravados.

---

13. O que é serialVersionUID?

Resposta:
Um identificador único que representa uma versão da classe serializável.

---

14. O que acontece se um objeto é desserializado com uma versão de classe diferente?

Resposta:
É lançada a exceção `InvalidClassException`.

---

15. O serialVersionUID é obrigatório?

Resposta:
Não, mas é altamente recomendado que seja declarado.

---

16. O que acontece se serialVersionUID não for declarado?

Resposta:
O Java gera automaticamente um número baseado na classe.

---

17. Para que serve ObjectInputStream?

Resposta:
Para desserializar objetos previamente gravados.

---

18. Qual método da ObjectInputStream lê objetos do stream?

Resposta:
`readObject()`.

---

19. O que é StreamCorruptedException?

Resposta:
Exceção lançada ao tentar ler um arquivo cujo cabeçalho do ObjectOutputStream está corrompido.

---

20. Quando StreamCorruptedException geralmente ocorre?

Resposta:
Quando se grava objetos usando append com FileOutputStream, adicionando vários cabeçalhos.

---

21. O cabeçalho do ObjectOutputStream tem quantos bytes?

Resposta:
4 bytes.

---

22. Pode-se serializar objetos de qualquer classe?

Resposta:
Não, apenas de classes que implementam Serializable.

---

23. Para que serve a classe ObjectOutputStream além da gravação de objetos?

Resposta:
Ela também grava o cabeçalho necessário para interpretar o arquivo de objetos.

---

24. A serialização de objetos permite armazenamento em quais mídias?

Resposta:
Qualquer armazenamento secundário (arquivos, rede, etc.).

---

25. É possível serializar Strings?

Resposta:
Sim, Strings são serializáveis.

---

26. O que ocorre com atributos transient após desserialização?

Resposta:
Eles perdem seu valor original e assumem valores padrão (null, 0, false).

---

27. O que ocorre se a classe mudou sua estrutura mas mantém o mesmo serialVersionUID?

Resposta:
O objeto pode ser desserializado sem erro, mas pode gerar inconsistências lógicas.

---

28. Como garantir que duas versões de uma classe são compatíveis na serialização?

Resposta:
Mantendo o mesmo serialVersionUID entre versões compatíveis.

---

29. Por que a serialização é útil?

Resposta:
Permite persistir objetos completos, incluindo seus estados.

---

30. Quais tipos de classes não devem ser serializadas?

Resposta:
Classes que representem recursos não persistíveis, como conexões, threads ou streams abertos. (Good practice geral, coerente com o PDF.)

1 – Sobre serialização de objetos:

a) ( ) Serialização converte objetos em texto legível.
b) ( ) Serialização pode gerar dados binários.
c) ( ) Objetos precisam implementar Serializable.
d) ( ) Variáveis estáticas são serializadas.

Gabarito: F, V, V, F

---

2 – Sobre desserialização:

a) ( ) readObject() reconstrói objetos gravados.
b) ( ) É obrigatório ler na mesma ordem em que se gravou.
c) ( ) Desserialização ignora tipos de dados.
d) ( ) Pode lançar ClassNotFoundException.

Gabarito: V, V, F, V

---

3 – Sobre a interface Serializable:

a) ( ) Contém métodos abstratos obrigatórios.
b) ( ) É uma interface de marcação.
c) ( ) Sem implementá-la não é possível serializar o objeto.
d) ( ) É da API padrão Java.

Gabarito: F, V, V, V

---

4 – Sobre ObjectOutputStream:

a) ( ) Sempre grava um cabeçalho no início do arquivo.
b) ( ) writeObject() grava objetos.
c) ( ) Pode gravar vários objetos seguidos.
d) ( ) É necessário usar append ao gravar vários objetos.

Gabarito: V, V, V, F

---

5 – Sobre ObjectInputStream:

a) ( ) Lê objetos na ordem gravada.
b) ( ) readObject() pode lançar exceções.
c) ( ) Precisa de FileInputStream.
d) ( ) Consegue ler dados de texto puro.

Gabarito: V, V, V, F

---

6 – Sobre atributos serializados:

a) ( ) Atributos transient não são serializados.
b) ( ) Atributos estáticos são serializados.
c) ( ) Apenas variáveis de instância são serializadas.
d) ( ) Strings são serializáveis.

Gabarito: V, F, V, V

---

7 – Sobre serialVersionUID:

a) ( ) Serve para versionar classes.
b) ( ) É opcional, mas recomendado.
c) ( ) Se não declarado, o Java cria um automaticamente.
d) ( ) Impede desserialização de versões diferentes.

Gabarito: V, V, V, V

---

8 – Sobre InvalidClassException:

a) ( ) Ocorre quando serialVersionUID é diferente.
b) ( ) É lançada durante escrita de objetos.
c) ( ) Impede desserialização inválida.
d) ( ) Está relacionada ao versionamento.

Gabarito: V, F, V, V

---

9 – Sobre StreamCorruptedException:

a) ( ) É causada por cabeçalhos duplicados.
b) ( ) Ocorre quando se usa append em arquivos de objetos.

c) ( ) Pode ser evitada criando apenas um ObjectOutputStream.
d) ( ) O cabeçalho possui 4 bytes.

Gabarito: V, V, V, V

---

10 – Sobre exemplos de serialização:

a) ( ) Strings podem ser serializadas.
b) ( ) Objetos de classe sem Serializable geram erro.
c) ( ) writeObject aceita qualquer tipo de objeto.
d) ( ) Objetos são armazenados na ordem escrita.

Gabarito: V, V, F, V

---

11 – Sobre classes persistidas:

a) ( ) Podem conter atributos transient.
b) ( ) Não podem ter comportamento diferente ao desserializar.
c) ( ) Podem ter vários objetos gravados em sequência.
d) ( ) Não precisam ter construtor padrão.

Gabarito: V, F, V, V

---

12 – Sobre métodos envolvidos:

a) ( ) writeObject() grava estado completo do objeto.
b) ( ) readObject() retorna Object.
c) ( ) readObject() precisa de casting.
d) ( ) writeObject() ignora atributos transient.

Gabarito: V, V, V, V

---

13 – Sobre gravar arquivos com append:

a) ( ) Pode causar StreamCorruptedException.
b) ( ) ObjectOutputStream grava novos cabeçalhos sempre que criado.
c) ( ) Não é recomendado usar append com serialização.
d) ( ) append funciona normalmente com DataOutputStream.

Gabarito: V, V, V, V

---

14 – Sobre erros comuns:

a) ( ) Falta de Serializable gera NotSerializableException.
b) ( ) Falta de serialVersionUID gera erro sempre.
c) ( ) Versões diferentes geram InvalidClassException.
d) ( ) Arquivos corrompidos podem gerar StreamCorruptedException.

Gabarito: V, F, V, V

---

15 – Sobre processo de serialização:

a) ( ) Objetos podem ser armazenados em arquivos.
b) ( ) Objetos podem ser enviados pela rede após serialização.
c) ( ) Serialização preserva métodos da classe.
d) ( ) Apenas o estado é persistido.

Gabarito: V, V, F, V