Manipulação de arquivos e serialização

Problema 1

- Escreva uma aplicação em Java que gerencie um arquivo de texto simples. Para isso, crie uma classe que permita gravar strings nesse arquivo e também recuperar todo o conteúdo gravado.

Problema 2

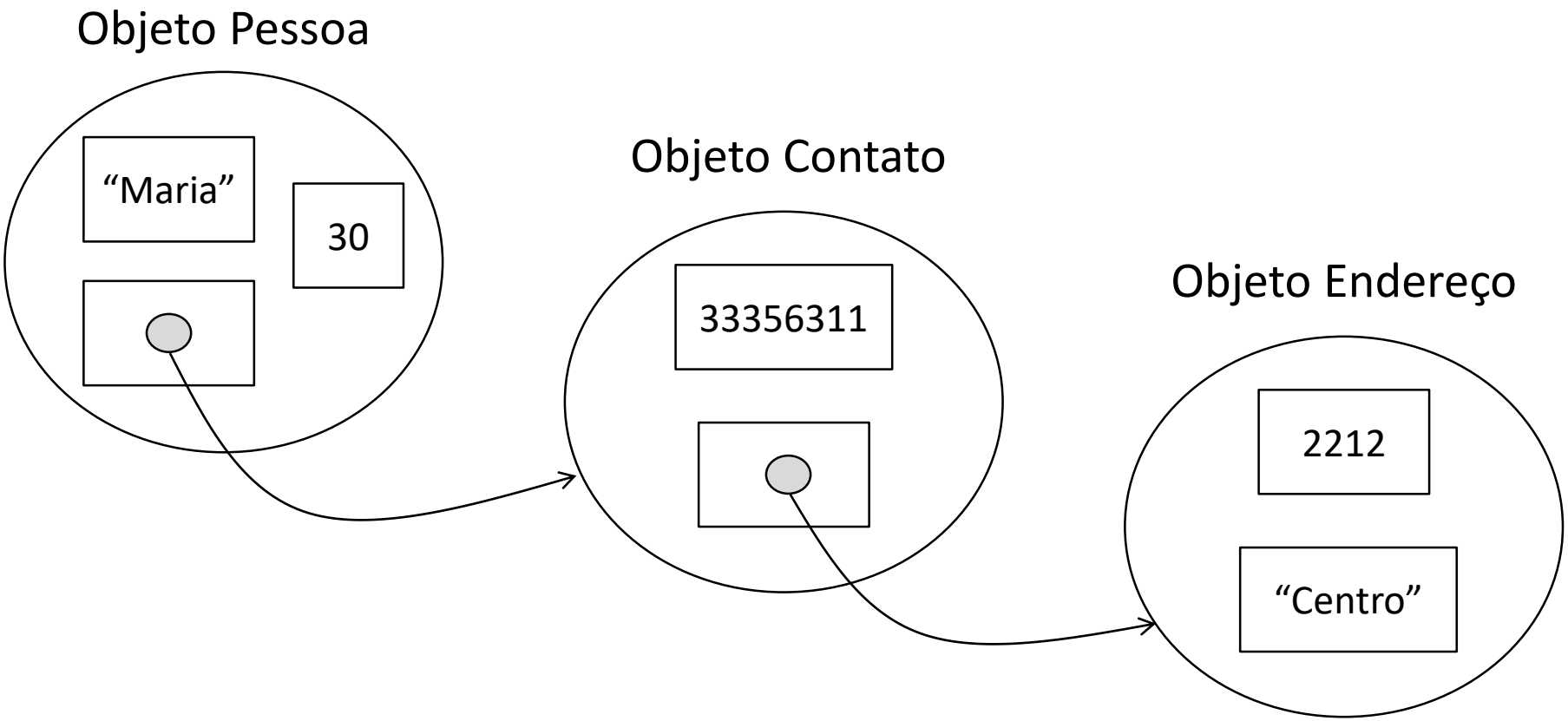
- Elabore uma aplicação em Java que permita gravar em um arquivo os dados de pessoas. Após gravar os dados, o programa deverá recuperá-los na forma de objetos e exibi-los na tela. Sabe-se que cada pessoa possui nome, idade e contato, que é composto por telefone e endereço. Cada endereço, por sua vez, possui rua e bairro. Crie uma classe que instancie uma pessoa com base em dados quaisquer, grave-as em um arquivo e, a seguir, remova suas referências. Após a remoção, a classe deve restaurar os objetos, com base nos dados gravados no arquivo.

Hipóteses

- Hipótese 1: Gravar todos os dados em um arquivo de texto de acordo com os métodos já utilizados.
- Limitações?
  - Definição de um formato para gravação e leitura.
  - Necessidade de gravação e recuperação dos dados de um objeto e dados de todos os objetos que são, de alguma forma, apontados por ele.

Hipóteses

- De acordo com o problema, temos:

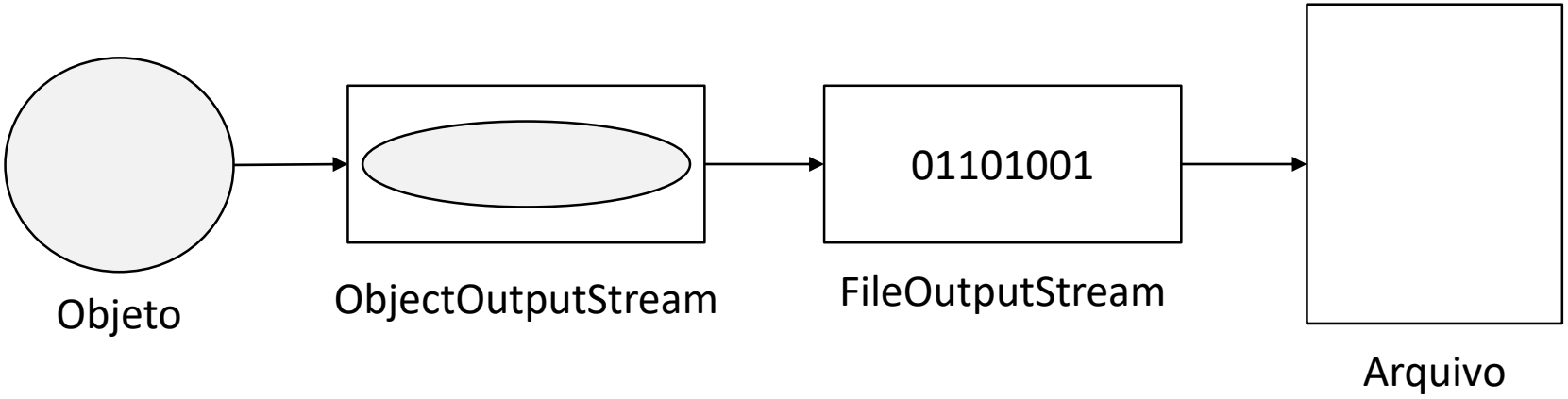


Hipóteses

- **Solução:**
  - Usar métodos para **serialização de objetos**.
- O que é serialização de objetos?
  - Serializar um objeto significa salvar seu estado por meio de uma sequência de bytes.
  - Objetos serializados podem ser reconstituídos em algum momento posterior.

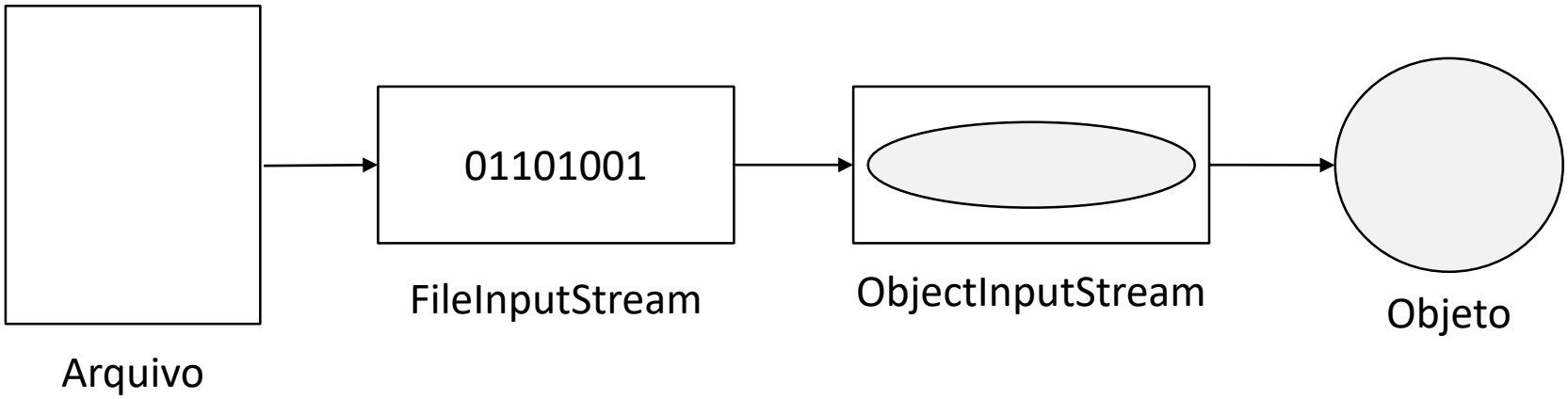
Solução

- Detalhe importante:
  - A serialização salva a ramificação inteira do objeto.
- Quais classes serão utilizadas na serialização?



Solução

- A ilustração anterior mostrou as classes básicas para o processo de serialização.
- E para o processo de desserialização?



Reflexão

- Todos os objetos em Java podem ser serializados?
  - A resposta é não.
- Para que um objeto seja serializado, sua classe deve satisfazer a um contrato.
- Esse contrato exige que uma interface chamada **Serializable** seja implementada.

Reflexão

- A interface Serializable possui algum método?
  - A resposta é não.
  - Ela serve apenas como interface marcadora ou interface de *tag*.
- Observação:
  - Todas as classes cujos objetos fazem parte da ramificação de um objeto a ser serializado devem implementar essa interface.

Novo problema

- Observou-se que a serialização salva os valores de todas as variáveis de instância do objeto e dos objetos referenciados por ele.
- Entretanto, o que fazer caso alguma variável de instância não possa ou não deva ser salva como parte do objeto?
  - Nesse caso, a variável de instância deverá ser marcada com a palavra-chave ***transient***.

Novo problema

- Nesse caso, qual será o valor dessa variável após a restauração do objeto?
  - O valor dessa variável será o valor *default*, de acordo com o seu tipo.
- Além de variáveis marcadas com a palavra-chave *transient*, todas as demais poderão ser salvas?
  - Variáveis estáticas também não podem ser salvas.

Mais problema

- Suponha que após a serialização de objetos do tipo Pessoa, a classe que represente pessoas deva ser alterada.
- O que ocorrerá no processo de desserialização?
  - A máquina virtual Java verificará que a classe usada para desserialização é diferente da classe original e lançará uma exceção.

Mais problema

- Entretanto, certas modificações na classe não causarão impacto no processo de desserialização.
- Nesse caso, como garantir que o objeto seja desserializado?
  - Definir explicitamente um número de versão para a classe.
  - Esse número deve ser declarado como uma variável estática *long* chamada **serialVersionUID**.

Mais problema

- Como a máquina virtual Java baseia-se no número de versão para efetuar a verificação, mesmo que a classe evolua, o número de versão permanecerá.

Exemplo

- Escreva uma aplicação em Java que simule um jogo organizado em níveis. A cada jogada, o usuário pode avançar um nível e, a qualquer momento, ele pode parar o jogo e salvar seu estado atual. Dessa forma, posteriormente, ele poderá voltar a jogar e continuar a partir do ponto em que o jogo estava no momento em que ele foi parado.

Exercício 1

- Elabore uma aplicação em Java que gere 10 números inteiros aleatórios entre 20 e 50, e armazene-os em um arquivo texto. Crie uma janela gráfica que possua dois botões: um botão que permita gerar 10 números, e outro botão que permita imprimir na tela os números de maneira inversa à ordem em que foram gerados (Use os componentes JTextArea e JScrollPane). Crie uma classe para cada responsabilidade (gerenciamento do arquivo, janela gráfica e gerenciamento dos números aleatórios). Caso o arquivo contenha alguma linha composta por um valor que não seja um número inteiro, uma exceção chamada “ValorInvalidoException” deverá ser lançada.

Exercício 2

- Crie uma aplicação em Java que simule um jogo de dados simples. Elabore uma janela gráfica que permita ao jogador iniciar um jogo (nesse momento, dois dados serão lançados e a soma de suas faces deverá ser calculada). Após iniciar um jogo, o jogador terá três chances para acertar a soma entre as faces. Caso o jogador deseje, ele poderá salvar o jogo corrente para que seja possível voltar a jogar em outro momento. Crie os seguintes botões: “Novo jogo”, “Jogar”, “Salvar jogo”, “Recuperar último jogo salvo”.

Exercício 3

- Modifique o exercício anterior de tal forma que o usuário possa selecionar o método de gravação que será utilizado. Considere o método de gravação por serialização e o método de gravação em arquivo de texto. Nesse último caso, os dados de um jogo devem ser separados pelo símbolo de ponto e vírgula (;). Dessa forma, a janela deve fornecer botões de radio para permitir a escolha e três botões de ação. Um botão será utilizado para que se possa definir o método escolhido e os outros dois serão utilizados para permitir a gravação e a leitura dos dados. A classe de janela gráfica não deve "saber" qual método será utilizado. Portanto, crie uma classe de controle cujo objeto armazene o método escolhido e, com base nesse método, solicite a operação de gravação e a operação de leitura. O código dessa classe também não deve ser dependente dos métodos específicos de gravação.

Exercício 4

- Crie uma aplicação em Java que exiba um botão cujas algumas propriedades sejam personalizadas pelo usuário. Para isso, a aplicação deve exibir uma janela gráfica que permita ao usuário informar características de fonte (nome, estilo e tamanho) e cor de fundo do botão. O estilo indica se a fonte será em negrito ou não. Após o usuário informar essas características e clicar em um botão chamado “Gravar”, os dados informados deverão ser gravados em um arquivo. Caso o usuário clique em um botão chamado “Abrir nova janela”, uma nova janela deverá ser carregada exibindo o botão de acordo com as propriedades definidas pelo usuário. Caso o usuário solicite a abertura da nova janela e não tenha gravado suas preferências, o botão deverá ser exibido a partir de propriedades *default*.