INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

AULA:

Serialização e Persistência de Objetos

Programação Orientada a Objetos

Alba Lopes, Profa.

http://docentes.ifrn.edu.br/albalopes
alba.lopes@ifrn.edu.br

Problema

- Nem sempre se deseja ter um objeto que existe somente durante o tempo em que o programa está executando
- É desejado que o objeto persista, mesmo após ser apagado da memória RAM
- Algumas vezes desejamos guardar os objetos por tempo indeterminado
  - Armazenar em disco e recuperar quando necessário
- Ex: Na classe da motocicleta, podemos incluir diversos objetos do tipo Motocicleta na coleção, percorrer a coleção e alterar os valores dos atributos dos objetos.
- Porém, ao fecharmos a aplicação e abrirmos novamente, esses objetos não existirão mais. Os valores dos objetos foram perdidos e novos objetos devem ser criados.
- Questão: Como fazer para que o objeto permaneça existindo mesmo após o fechamento da aplicação?

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Persistência e Serialização

- Persistir objetos requer que esses objetos possam ser serializáveis
  - A serialização possibilitará capturar o estado do objeto ou a estrutura de dados, e transformar em uma cadeia de bytes quando for necessário
- Dessa forma, qualquer forma de dados que esteja sendo trabalhada na aplicação, e puder ser serializada, poderá :
  - ser passada para o meio físico (arquivo de texto, por exemplo), e
  - ser lido do meio físico e recuperada em um objeto

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Serialização em Java

- Serializar um objeto, em Java, só é possível caso sua classe esteja marcada como serializável
- Para tanto, é preciso que a classe implemente a **interface *Serializable*,** do pacote *java.io.Serializable*
- Essa interface não possui métodos para se implementar e nem atributos
- A herança entre classes é naturalmente afetada pela serialização. Já que quando uma classe implementa a interface “Serializable”, toda classe que dela estender também estará implicitamente marcada como serializável

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Serialização em Java

```
public class Motocicleta implements Serializable{
    public String marca;
    public String modelo;
    public int velocidade;

    public void acelerar(int valor){
        velocidade+= valor;
    }
    public void frear(int valor){
        velocidade-= valor;
    }
    public void parar(){
        velocidade = 0;
    }

    /* demais métodos */

}
```


Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Serialização em Java

- A partir do momento em que a classe está “serializada”, é possível utilizar métodos para gravar essa classe em um arquivo de texto, por exemplo, bem como ler os dados do arquivo e atribuí-los a objetos
- O procedimento é bem fácil e pode-se considerar genérico para qualquer classe serializada.
- A classe apresentada a seguir pode ser usada como padrão sempre que desejado gravar um objeto em arquivo e ler os dados para um objeto.

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Serialização e Deserialização em Java

- Algumas classes são fundamentais para esse processo:
  - **ObjectOutputStream**: classe que possui o método mágico (write) de serialização. Permite gravar o objeto em arquivo.
  - **ObjectInputStream**: classe que possui o método mágico (read) de serialização. Permite ler os dados do arquivo transformá-lo em Object.
  - **FileInputStream** e **FileOutputStream** trabalham diretamente com a gravação e leitura de arquivo.

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Classe para gravação e leitura de objetos

```
public class Serializador {

    public void gravar(String caminho, Object objeto)
            throws FileNotFoundException, IOException{
        FileOutputStream outFile = new FileOutputStream(caminho);
        ObjectOutputStream s = new ObjectOutputStream(outFile);
        s.writeObject(objeto);
        s.close();
    }

    public Object ler(String caminho)
            throws FileNotFoundException, IOException, ClassNotFoundException{
        FileInputStream inFile = new FileInputStream(caminho);
        ObjectInputStream s = new ObjectInputStream(inFile);
        Object objeto = s.readObject();
        s.close();
        return objeto;
    }
}
```


Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Classe para gravação e leitura de objetos

```
public class Serializador {

    public static void gravar(String caminho, Object objeto)
            throws FileNotFoundException, IOException{
        FileOutputStream outFile = new FileOutputStream(caminho);
        ObjectOutputStream s = new ObjectOutputStream(outFile);
        s.writeObject(objeto);
        s.close();
    }

    public static Object ler(String camin
            throws FileNotFoundException, IOException, ClassNotFoundException{
        FileInputStream inFile = new FileInputStream(caminho);
        ObjectInputStream s = new ObjectInputStream(inFile);
        Object objeto = s.readObject();
        s.close();
        return objeto;
    }
}
```

FileOutputStream cria um arquivo no caminho passado por parâmetro

ObjectOutputStream recebe como parâmetro o arquivo que irá armazenar objetos serializáveis

writeObject grava os bytes referentes ao objeto serializável no arquivo.

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Classe para gravação e leitura de objetos

```
public class Serializador {

    public static void gravar(String caminho, Object objeto)
            throws FileNotFoundException, IOException{
        FileOutputStream outFile = new FileOutputStream(caminho);
        ObjectOutputStream s = new ObjectOutputStream(outFile);
        s.writeObject(objeto);
        s.close();
    }

    public static Object ler(String caminho)
            throws FileNotFoundException, IOException, ClassNotFoundException{
        FileInputStream inFile = new FileInputStream(caminho);
        ObjectInputStream s = new ObjectInputStream(inFile);
        Object objeto = s.readObject();
        s.close();
        return objeto;
    }
}
```

FileInputStream abre o arquivo que está no caminho passado por parâmetro

ObjectInputStream recebe como parâmetro o arquivo que irá armazenar objetos serializáveis

readObject lê os bytes referentes ao objeto serializável e retorna o objeto.

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Utilizando a classe Serializador

- Gravando o objeto no arquivo “teste.dat”

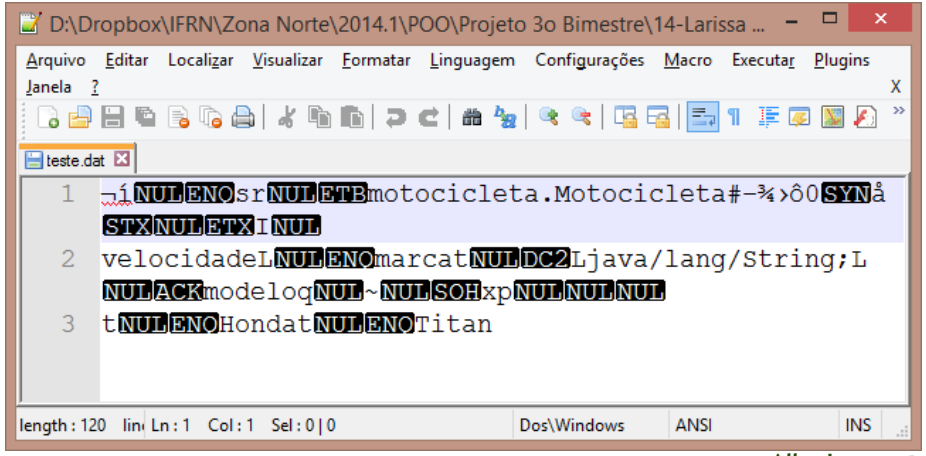
```
public class TestarGravacaoObj {
    public static void main(String[] args) throws IOException {
        Motocicleta m = new Motocicleta();
        m.setMarca("Honda");
        m.setModelo("Titan");
        m.setVelocidade(10);
        Serializador.gravar("teste.dat", m);
    }
}
```

Como o método foi definido como estático, pode-se chamar o método sem precisar instanciar a classe.

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Comprovando a gravação dos dados no arquivo

- Procure na pasta do seu projeto o arquivo teste.dat criado.
- Abra o arquivo e verifique os dados contidos nele. Os bytes representando o objeto foram armazenados.
- Não é possível identificar tudo o que está armazenado, porém o utilizar o método ler da classe Serializador será possível recuperar o conteúdo em um objeto



Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Utilizando a classe Serializador

- Lendo o objeto no arquivo “teste.dat”

```
public class TestarLeituraObj {
    public static void main(String[] args)
            throws IOException, FileNotFoundException, ClassNotFoundException {

        Motocicleta m = (Motocicleta) Serializador.ler("teste.dat");
        System.out.println(m.getModelo());
    }
}
```

É preciso fazer um *cast* para o tipo do objeto que está sendo recuperado.

Como o método foi definido como estático, pode-se chamar o método sem precisar instanciar a classe.

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Gravando Arrays de Objetos

```
public class TestarGravacaoArray {
    public static void main(String[] args) throws IOException {
        ArrayList<Motocicleta> minhasMotos = new ArrayList<>();
        Motocicleta m1 = new Motocicleta();
        Motocicleta m2 = new Motocicleta();
        Motocicleta m3 = new Motocicleta();
        minhasMotos.add(m1);
        minhasMotos.add(m2);
        minhasMotos.add(m3);
        Serializador.gravar("testeArray.dat", minhasMotos);
    }
}
```

O método gravar da classe Serializador recebe como parâmetro um Object. Qualquer objeto pode ser passado por parâmetro, inclusive um array de objetos.

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Lendo Arrays de Objetos

```
public class TesteLeituraArray {
    public static void main(String[] args) throws IOException,
            FileNotFoundException, ClassNotFoundException {

        ArrayList<Motocicleta> minhasMotos;
        minhasMotos = (ArrayList<Motocicleta>) Serializador.ler("teste.dat");
        for(Motocicleta m : minhasMotos){
            System.out.println(m.getModelo());
        }
    }
}
```

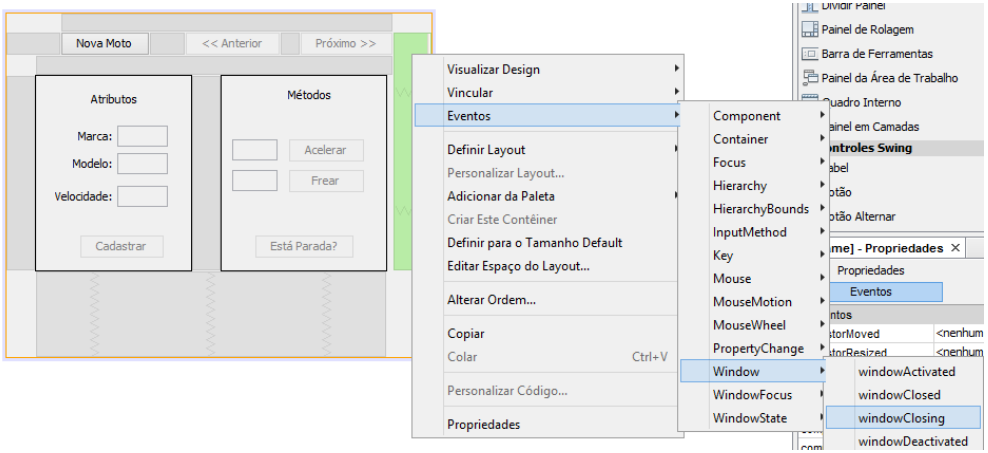
Variação prática do laço for para percorrer coleções de objetos

O método ler da classe Serializador pode retornar qualquer objeto, inclusive um array de objetos. É preciso, entretanto, fazer um *cast* para o tipo de objeto específico

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Exemplo Prático - Motocicleta

- Voltando ao nosso exemplo da classe InterfaceMotocicleta
  - Vamos fazer com que nossa coleção de motos seja persistida no momento em que fechamos a janela da aplicação.
  - Para tanto, adicione um evento para monitorar o fechamento da janela. Clique em qualquer parte vazia do seu JFrame com botão direito **Eventos → Window → windowClosing**



Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Exemplo Prático - Motocicleta

- No código do método **formWindowClosing** adicione a seguinte linha:

```
private void formWindowClosing(java.awt.event.WindowEvent evt) {
    Serializador.gravar("minhasmotos.dat", minhasMotos);
}
```

minhasmotos.dat é o nome do arquivo que você deseja que os dados sejam gravados

minhasMotos é a coleção de objetos do tipo Motocicleta que criamos anteriormente

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Exemplo Prático - Motocicleta

- No código do método **formWindowClosing** adicione a seguinte linha:
  - Circunde o bloco com a instrução try-catch, conforme solicitado

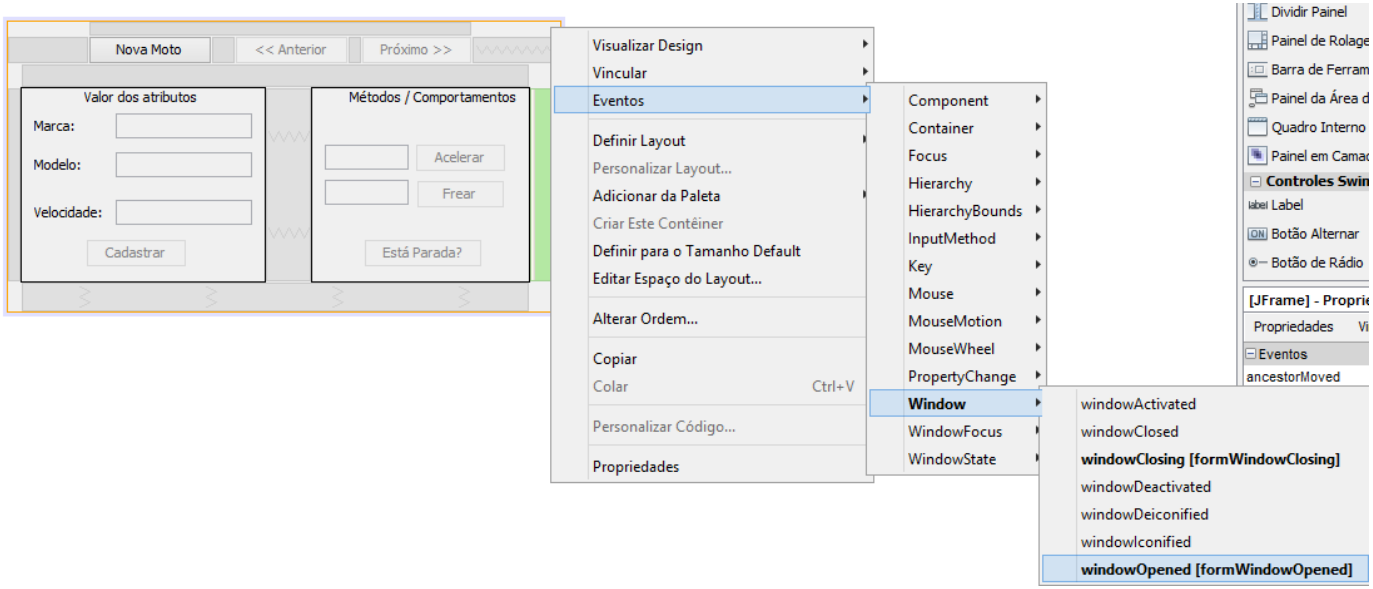
```
351     private void formWindowClosing(java.awt.event.WindowEvent evt) {
            Serializador.gravar("minhasmotos.dat", minhasMotos);
353     Circundar Instrução com try-catch
```

```
351     private void formWindowClosing(java.awt.event.WindowEvent evt) {
352         try {
353             Serializador.gravar("minhasmotos.dat", minhasMotos);
354         } catch (IOException ex) {
355             Logger.getLogger(InterfaceMotocicleta.class.getName()).log(Level.SEVERE, null, ex);
356         }
357     }
```


Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Exemplo Prático - Motocicleta

- Agora vamos recuperar os dados sempre que a janela da aplicação for aberta.
- Para tanto, clique em qualquer parte vazia do Jframe com o botão direito. Selecione: **Eventos → Window → windowOpened**



Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Exemplo Prático - Motocicleta

- Adicione o seguinte trecho de código ao método formWindowOpened

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {

        minhasMotos = (ArrayList<Motocicleta>) Serializador.ler("minhasmotos.dat");
        habilitarProximoAnterior();

}
```


Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Exemplo Prático - Motocicleta

- Circunde o bloco com try-catch de modo a capturar as exceções quando ocorrerem:
  - A primeira exceção (IOException) será disparada sempre que a aplicação for aberta pela primeira vez e nenhum catálogo de motos tiver sido criado. Nesse caso, exibiremos uma mensagem indicando que um novo catálogo será criado (essa mensagem é opcional!)

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    try{
        minhasMotos = (ArrayList<Motocicleta>) Serializador.ler("minhasmotos.dat");
        habilitarProximoAnterior();
    }catch(IOException io){
        JOptionPane.showMessageDialog(this, "Iniciando novo catálogo de motos.");
    }catch( ClassNotFoundException e){
        e.printStackTrace();
    }
}
```


Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Teste sua aplicação!

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Exercício

- Torne os objetos da sua aplicação da ContaCorrente persistentes, de forma a recuperar os dados já gravados sempre que a aplicação for aberta.

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br

Referências

- http://www.linhadecodigo.com.br/artigo/3401/serializacao-solucao-para-persistencia-de-objetos.aspx

Alba Lopes, Profa.
alba.lopes@ifrn.edu.br