

# Tópicos Abordados



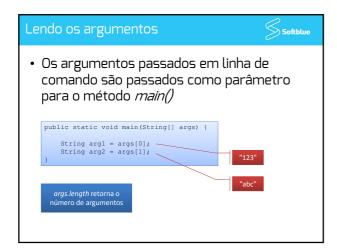
- Argumentos de linha de comando
- A classe *java.util.Properties* 
  - Lendo e escrevendo em arquivos de propriedades
- Serialização de objetos
  - ObjectInputStream e ObjectOutputStream
  - Controlando a serialização
  - Controle de versão

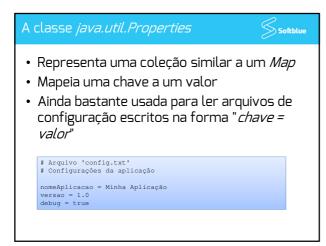
## Argumentos de linha de comando

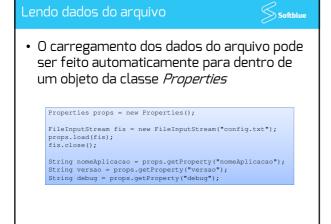


- O Java permite que argumentos de linha de comando sejam fornecidos para a aplicação
- Estes argumentos podem ser lidos pelo programa









# Escrevendo dados no arquivo



• Com o *Properties* é possível também salvar os dados num arquivo de configuração

```
Properties props = new Properties();

props.setProperty("nomeAplicacao", "Minha Aplicação");
props.setProperty("versao", "1.0");
props.setProperty("debug", "true");

FileOutputStream fos = new FileOutputStream("config_save.txt");
props.store(fos, "Gerado pela Aplicação");
fos.close();
```

## Escrevendo dados no arquivo



• Arquivo de saída gerado (config\_save.txt)

#Gerado pela Aplicação versao=1.0 debug=true nomeAplicacao=Minha Aplicação

• Para mais informações sobre a classe *Properties*, consulte o Javadoc

## Serialização de objetos



- Serializar significa transformar em bytes
- Objetos são serializados para que possam ser:
  - transferidos pela rede
  - armazenados no disco rígido

# A interface Serializable Deve ser implementada pelas classes cujos objetos serão serializados public class Mensagem implements Serializable { } Esta interface não possui qualquer método que deva ser implementado pela classe





# Serialização dos atributos



- Quando um objeto serializado possui referências a outros objetos, estes objetos também são serializados
  - Todos os objetos referenciados pela classe devem implementar a interface *Serializable*
- É possível descartar atributos da serialização usando o modificador transient

public class Mensagem {
 private transient String tempMsg;

# Controlando a serialização



- Existem duas formas de ter total controle sobre a serialização
  - Implementando os métodos
    - writeObject(ObjectOutputStream out)
    - readObject(ObjectInputStream in)
  - Implementando a interface *Externalizable* 
    - writeExternal(ObjectOutput out)
    - readExternal(ObjectInput in)
- A JVM chama os métodos caso eles existam

# Controle de versão



- Quando um objeto é serializado, a JVM cria um código numérico para a classe
  - Este código não é randômico
  - É baseado nos nomes dos atributos e assinaturas dos métodos da classe
- Este número identifica a classe, de forma que se algo na classe mudar, este número também muda
- Ao tentarmos ler um objeto serializado depois que a classe for modificada, uma InvalidClassException será lançada

### Controle de versão



- Nem sempre este comportamento é desejado
- Para evitar que isso aconteça, devemos fornecer este número de versionamento manualmente

private static final long serialVersionUID = 1415906L;

• Isto garante que o processo de leitura de um objeto serializado será feito de forma correta

