The BlueJ Tutorial

Versão 1.4

para BlueJ Versão 1.2.x

Michael Kölling
Mærsk Insitute
University of Southern Denmark

traduzido para o português por João Luiz Barbosa (jluiz_barbosa@yahoo.com.br) em outubro/2002
mestrando no CEFET (www.dppg.cefetmg.br) e professor na FUMEC (www.fumec.br)

1 Prefácio

1.1 Sobre o BlueJ

Este tutorial é uma introdução para o uso do ambiente de programação do BlueJ. BlueJ é um ambiente de desenvolvimento java projetado especificamente para o ensino em um nível introdutório. BlueJ foi projetado e implementado pelas equipes das universidade Monash University, Melbourne, Australia e The University of Southern Denmark, Odense. Maiores informações sobre BlueJ estão disponíveis no site oficial (http://www.bluej.org).

1.2 Escopo e audiência

Este tutorial visa às pessoas que querem familiarizar-se com as potencialidades do ambiente. Ele não explica decisões de projetos usadas na construção do ambiente e nem nas questões associadas a pesquisa desenvolvidas para ele.

Este tutorial não pretende ensinar Java. Os novatos na programação Java devem procurar estudar a linguagem através de livros introdutórios ou cursos da linguagem.

Este não é o manual de referência detalhado do ambiente. Muitos detalhes são deixados de fora – a ênfase está em uma introdução breve e concisa e não na cobertura completa dos recursos.

Cada seção começa com um sumário. Isto permite que os usuários já familiarizados com ambientes de desenvolvimento decidir se querem ler ou saltar cada parte particular. A seção 10 repete apenas estes sumários visando ser uma referência rápida.

1.3 Copyright, licenciamento e distribuição

O software BlueJ e este tutorial estão disponíveis a qualquer um para qualquer tipo de uso. O software e sua documentação podem ser distribuídos livremente.

Nenhuma parte do BlueJ ou sua documentação pode ser vendida ou incluída em pacotes que sejam vendidos sem a autorização explicita dos autores.

Os direitos de propriedade do software BlueJ são propriedades de M. Kölling and J. Rosenberg.

1.4 Feedback

É muito bem vindo os comentários, questões, correções, criticas e qualquer outro tipo de retorno a respeito do software BlueJ ou sobre este material. Por favor, envie suas observações para Michael Kölling (mik@mip.sdu.dk).

2 Instalação

O BlueJ é distribuído em três formatos diferentes: um para o Windows, um para o MacOS, e um para todos os sistemas restantes. A instalação é fácil e completamente automática.

Pré-requisitos

Para usar o BlueJ, você deve ter o J2SE v1.3 (a.k.a. JDK 1.3) instalado ou providenciar a instalação dele no seu sistema operacional após a instalação do BlueJ. Se você não tiver o JDK instalado, você pode fazer o download a partir do site da Sun no endereço http://java.sun.com/j2se /. o MacOS X já possui uma versão do JDK preinstalada – você não necessita instalá-la novamente. Se você encontrar, durante a visita aos sites de download, as versões JRE (ambiente run-time Java) e o SDK (kit de desenvolvimento Java), você deve fazer o download do SDK. O JRE não é suficiente para a atividade de desenvolvimento de software.

2.1 Instalação no Windows

O programa de instalação para o Windows é chamado de *bluejsetup-xxx.exe*, onde xxx é o número da versão. Por exemplo, a versão 1.2.0 do BlueJ é chamada de *bluejsetup-120.exe*. Você pode ter este arquivo fazendo o download a partir do site do BlueJ (http://www.bluej.org). Para instalar, basta executar o programa.

O programa instalador permite você selecionar o diretório para a instalação. Ele oferecerá também a opção de criar um atalho no menu iniciar e no seu desktop.

Após o termino da instalação, você poderá encontrar o programa bluej.exe dentro do diretório escolhido.

Na primeira vez que o BlueJ for acionado, ele procurará o ambiente Java (JDK). Se ele encontrar mais de um ambiente Java (por exemplo, você tem instalado o JDK 1.3.1. e o JDK 1.4), uma janela de diálogo aparecerá para você selecionar qual ambiente será utilizado. Se ele não encontrar nenhum ambiente Java, será solicitado a você que localize o ambiente. Isto pode ocorrer se você tiver o ambiente instalado e as entradas na registry não estiverem corretas ou estiverem ausentes.

O programa de instalação também instala um programa chamado *vmselect.exe*. Usando este programa, você poderá mais tarde mudar a opção do ambiente Java que será utilizado pelo BlueJ. Execute o *vmselect* para iniciar o BlueJ com uma versão diferente do ambiente java.

A escolha do JDK é armazenada para cada versão do BlueJ. Se você tiver versões diferentes do BlueJ instaladas, você pode usar uma versão do BlueJ com o JDK 1.3.1 e outra versão do BlueJ com a versão do JDK 1.4. Entretanto, mudar a versão do ambiente Java para um BlueJ alterará todas as versões das instalações do BlueJ com a mesma versão para o mesmo usuário.

2.2 Instalação no Macintosh

O BlueJ só pode ser executa no no MacOS X.

O programa de instalação para o MacOS é chamado *BlueJ-xxx.sit*, onde *xxx* é o número da versão. Por exemplo, a versão 1.2.0 do BlueJ é chamada de *bluejsetup-120.exe*. Você pode ter este arquivo fazendo o download a partir do site do BlueJ (http://www.bluej.org). Para instalar, basta executar o programa.

Este arquivo pode ser descompactado com o *StuffIt Expander*. Vários browser descompactarão o arquivo. Se isto não ocorrer, de um duplo-click para descompactá-lo.

Após descompactá-lo, você terá uma pasta chamada BlueJ-xxx. Movimente está pasta para a pasta de aplicações (ou para onde você quiser guardar). Nenhuma operação adicional é necessária para finaliza a instalação.

2.3 Instalação no Linux/Unix e em outros sistemas

O arquivo de distribuição do software é um jar executável. Ele é chamado *bluej-xxx.jar*, onde xxx é o número da versão. Por exemplo, a versão 1.2.0 do BlueJ é chamada de *bluejsetup-120.exe*. Você pode ter este arquivo fazendo o download a partir do site do BlueJ (http://www.bluej.org). Para instalar, basta executar o programa

Execute o programa de instalação através do comando abaixo. NOTA: Para este exemplo, eu utilizei o arquivo *bluej-120.jar* – você deve utilizar o nome do arquivo que você baixou do site (com o número correto de versão).

```
<jdk-path>/bin/java -jar bluej-120.jar
```

`<jdk-path>` é o diretório onde o JDK está instalado.

Uma janela será exibida, permitindo você escolher o diretório de instalação e a versão do JDK que será utilizada pelo BlueJ. Importante: O caminho de diretório do BlueJ (isto é, nenhum dos diretórios pai) não podem conter espaços em seus nomes.

Click em *Install*. Ao final, BlueJ estará instalado.

2.4 Problemas na instalação

Se você tiver algum tipo de problema durante a instalação, verifique na *Frequently Asked Questions* (FAQ) no web site do BlueJ (http://www.bluej.org/help/faq.html). Leia também a seção *How To Ask For Help* (http://www.bluej.org/help/ask-help.html).

3 Começando do início – editando / compilando / executando

3.1 Iniciando o BlueJ

No Windows e no MacOS, o programa BlueJ foi instalado. Execute-o.

Em sistemas UNIX, o programa de instalação criou um script chamado *bluej* no diretório escolhido. Através da interface gráfica, acione o script com um duplo-click. Você também pode acioná-lo através da linha de comando. Neste caso pode ou não ser passado o nome do projeto como parâmetros.

```
$ bluej
```

ou

```
$ bluej examples/people
```

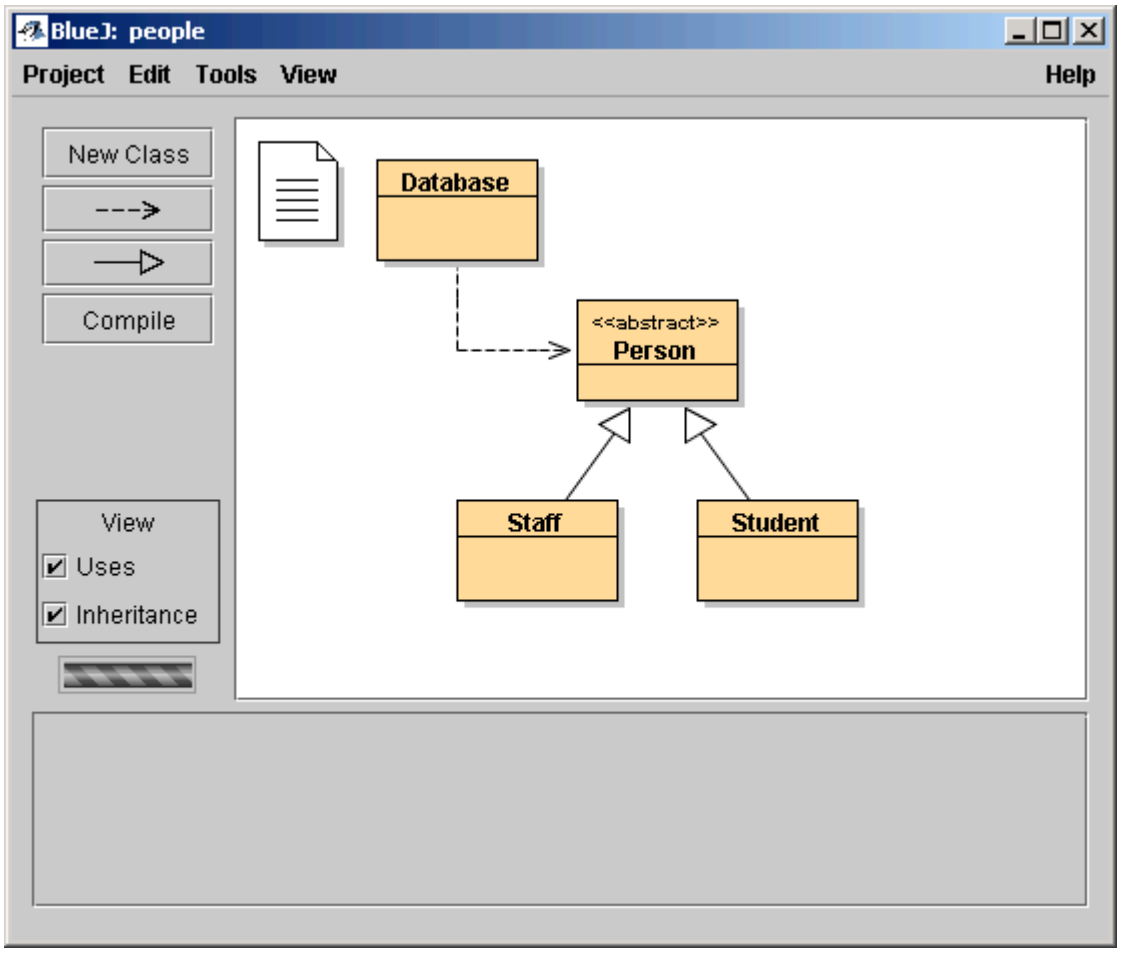

Figura 1: A janela principal do BlueJ

3.2 Abrindo um projeto

Sumário: Para abrir um projeto, selecione Open *no menu* Project.

Projetos usando o BlueJ, como os packages no Java, são diretórios que contem os arquivos que participam do projeto.

Após iniciar o BlueJ, utilize o menu Projet – Open para selecionar e abrir um projeto.

Alguns projetos de exemplo estão incluídos na distribuição padrão do BlueJ dentro do diretório *examples*.

Para esta seção do tutorial, abra o projeto *people*. Você pode encontrá-lo dentro do diretório *examples* que está dentro do diretório home do BlueJ. Após ter aberto o projeto, você deve ver algo similar à janela mostrada na figura 1. A janela pode não estar exatamente com na figura, dependendo do seu ambiente operacional, mas as diferenças devem ser mínimas.

3.3 Criando objetos

Sumário: Para criar um objeto de uma classe, selecione o construtor no menu popup da classe desejada.

Uma das características fundamentais do BlueJ é que você não executa somente a aplicação inteira, mas você pode também interagir diretamente com os objetos de qualquer classe e executar os métodos públicos destes objetos. Uma execução do BlueJ é feita geralmente criando um objeto e acionando um dos seus métodos. Isto é muito útil durante o desenvolvimento de uma aplicação – você pode testar classes individualmente logo após eles terem sido construídos. Não há necessidade de escrever a aplicação completa para executar parte dela.

> Nota*: Os métodos estáticos podem ser executados diretamente sem criar a primeira instância de um objeto. Um destes métodos estáticos pode ser o "main", assim nós podemos fazer a mesma coisa que uma aplicação comum em Java – iniciar uma aplicação a partir da execução do método estático main. Nós voltaremos a este tópico mais tarde. Primeiro, faremos outras coisas mais interessantes que não podem ser feitas normalmente em outros ambientes Java.*

Os quadrados que você vê na parte central da janela principal (nomeados *Database*, *Person*, *Staff* e *Student*) são os ícones que representam as classes envolvidas nesta aplicação. Você pode começar obtendo um menu com as operação aplicáveis a uma classe através do click com o botão direito do mouse (Macintosh: ctrl-click[1]) (Figura 2). As operações mostradas são as operações de criação de objetos, uma para cada construtora da classe, seguidas de algumas operações disponibilizadas pelo ambiente.

[1] Sempre que mencionarmos o click com botão direito neste tutorial, usuários do Macintosh devem usar o *ctrl-click*.

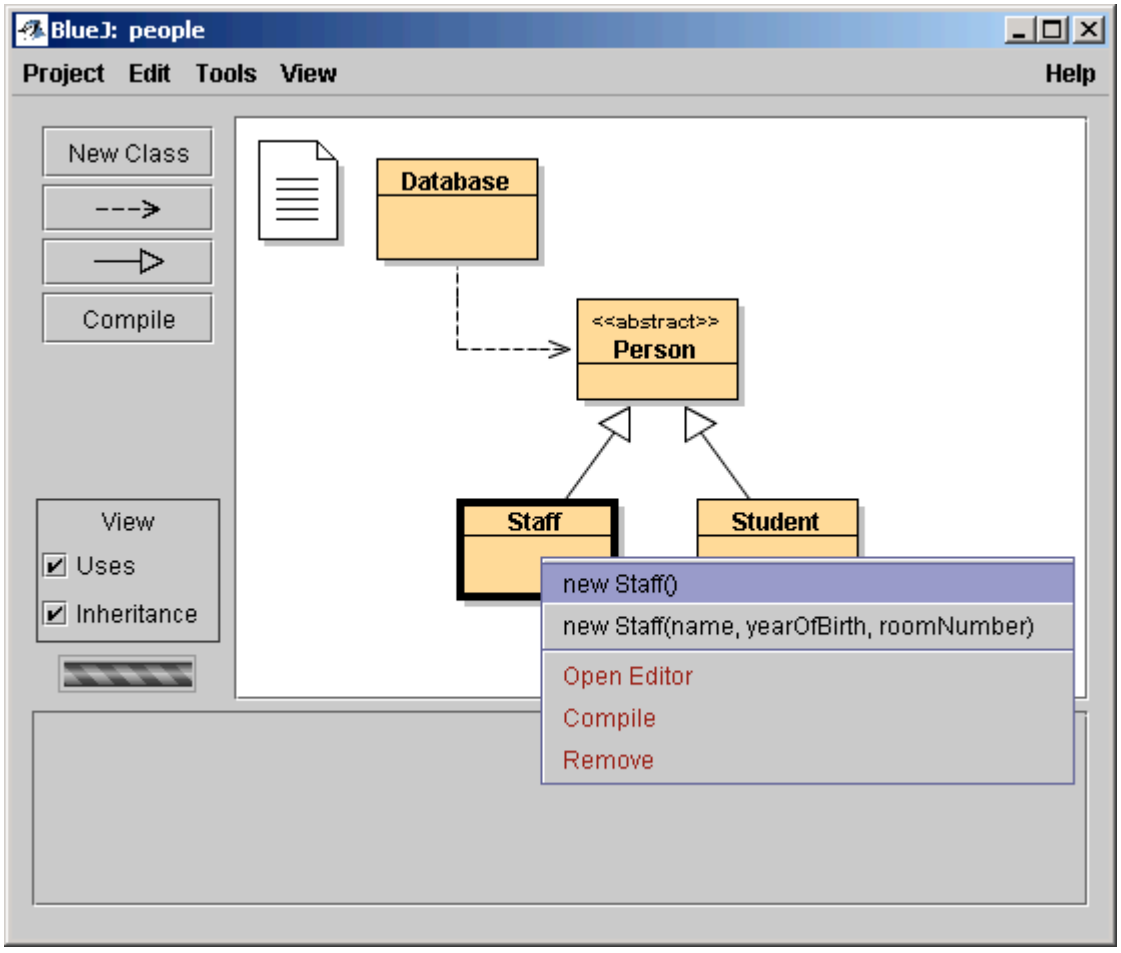

Figura 2: Operações da classe (menu popup)

Nós queremos criar um objeto *Staff*, para isto você deve usar o click com o botão direito no ícone *Staff* (que mostrará um menu como na Figura 2). O menu mostra dois construtores para criar o objeto *Staff*, um com parâmetros e outro sem. Primeiro, selecione o construtor sem parâmetros. Uma janela de diálogo, como na Figura 3, aparecerá.

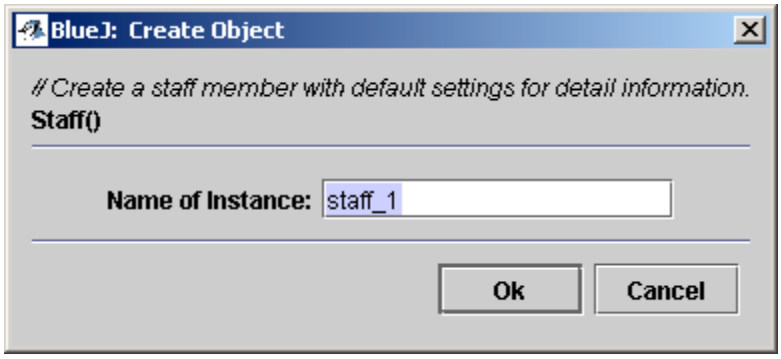

Figura 3: Criação de objetos sem o uso de parâmetros

Este diálogo pede um nome para o objeto a ser criado. Ao mesmo tempo, um nome padrão (*staff_1*) é sugerido. Este nome padrão é bom para este exemplo, portanto apenas click no botão OK. Um objeto *Staff* será criado.

Uma vez que o objeto foi criado, ele será colocado na bancada de objetos (Figura 4). Este é todo o processo para criar um objeto : selecione um construtor no menu popup, execute-o e você terá o objeto criado e visível na bancada de objetos.

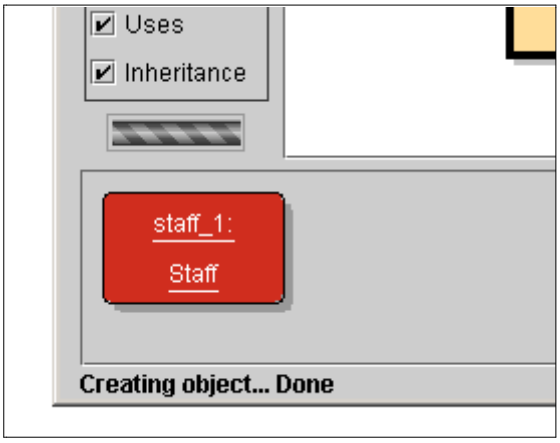

Figura 4: Um objeto na bancada de objetos

Você pode ter observado que a classe *Person* recebeu uma indicação de <<abstract>> (é uma classe abstrata). Você observará (se tentar fazer isto) que você não consegue criar objetos de classes abstratas (A especificação da linguagem Java define que isto não é possível).

3.4 Execução

Sumário: Para executar um método, selecione-o através do menu popup.

Agora que você criou um objeto, você pode executar suas operações públicas (em Java chamamos operações de métodos). Click com o botão direito no objeto e um menu com as operações disponíveis aparecerá (Figura 5). O menu mostra os métodos disponíveis para este objeto e duas operações especiais fornecidas pelo ambiente (*Inspect* e *Remove*). Nós as discutiremos mais tarde. Por agora, vamos focar nos outros métodos.

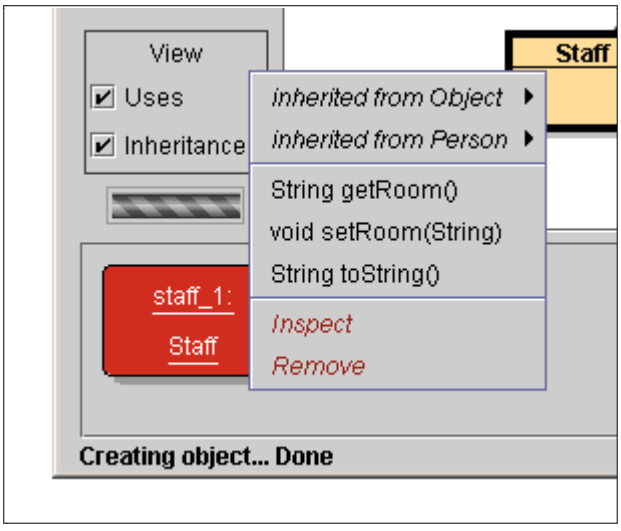

Figura 5: o menu de um objeto

Você pode ver que existem os métodos *getRoom* e *setRoom* que atribuem e retornam o numero da sala do membro do staff. Tente chamar *getRoom*. Selecione-o simplesmente no menu do objeto e ele será executado. Um diálogo aparecerá mostrando a você o resultado da chamada efetuada (Figura 6). Neste caso o nome será "(unknown room)" por que nós não especificamos a sala para esta pessoa.

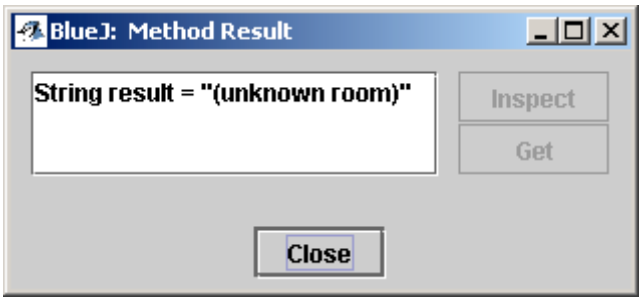

Figura 6: Mostra o resultado da função

Os métodos herdados de uma superclasse também estão disponíveis no submenu. No alto do menu popup do objeto, existe dois submenu, um para os métodos herdados da classe *Object* e outro para os métodos herdados da classe *Person* (Figura 5). Você pode chamar os métodos da classe *Person* (como o *getName*) selecionando ele no submenu. Tente isto. Você observará que a resposta é igualmente vaga: "(unknown name)", porque nós não atribuímos o nome ao objeto.

Vamos especificar o número da sala. Isto mostrará como fazer uma chamada com parâmetros (As chamadas de *getRoom* e *getName* tiveram valor de retorno, mas não tiveram parâmetros). Acione a função *setRoom* através da seleção no menu. Uma janela de diálogo aparecerá para obter o valor do parâmetro (Figura 7).

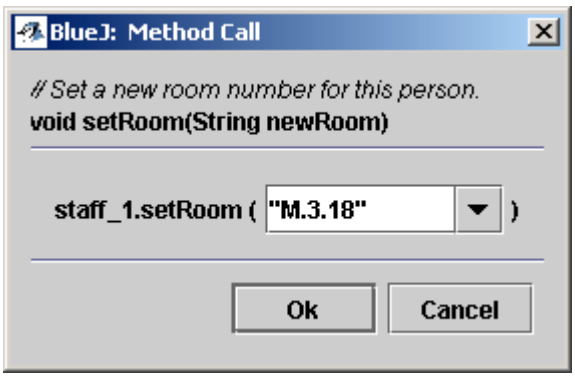

Figura 7: Chamada de função com passagem de parâmetro

No alto, o diálogo mostra a interface do método que foi acionado (incluindo os comentários e a assinatura). Abaixo disto está o campo para a entrada de dados onde você pode entrar com o parâmetro. A assinatura, no topo, nos informa que o parâmetro esperado é do tipo String. Entre com o valor da sala com uma String (incluindo as aspas) no campo de teste e acione o botão OK.

Isto é tudo – desde que o método não retorne nenhum valor, nenhum dialogo de resposta é mostrado. Chame o método *getRoom* novamente e verifique que a sala foi realmente alterada.

Experimente criar novos objetos e chamar alguns de seus métodos. Tente chamar o construtor com argumento e chame também outros métodos até ficar familiarizado com estas operações.

3.5 Editando uma classe

Sumário: Para editar o código fonte de uma classe, de um duplo-click no ícone da classe.

Até agora, nós utilizamos somente a interface do objeto. Agora veremos o que acontece dentro da classe. Você pode ver a implementação da classe selecionando *Open Editor* selecionando no popup de operações da classe. (Lembre-se: click com o botão direito no ícone da classe mostra as operações da classe). Um duplo-click no ícone da classe é um atalho para a função de edição. O editor não será descrito em detalhes neste tutorial, mas ele é de fácil uso. Os detalhes do editor serão tratados em outro material. Agora abra a implementação da classe *Staff*. Encontre a implementação do método *getRoom*. Ele retorna, como o próprio nome sugere, o número da sala da pessoa membro do staff. Vamos tentar mudar o método. Adicione o prefixo "room" no retorno da função (para que o método retorne "room M.3.18" ao invés de "M.3.18"). Nós podemos fazer isto mudando a linha

```
return room;
```

para

```
return "room " + room;
```

BlueJ possui suporte integral ao Java, portanto não existe nada de especial em como implementar suas classe.

3.6 Compilação

Sumário: Para compilar uma classe, click no botão Compile *no editor de texto. Para compilar um projeto, click no botão* Compile *na janela de visualização do projeto.*

Após inserir o texto (antes que qualquer outra coisa), verifique a janela de projeto (a janela principal). Você observará que o ícone da classe *Staff* foi alterado: Ele está com uma lista agora. Esta lista no ícone sinaliza que a classe ainda não foi compilada desde a ultima alteração. Retorne para o editor de código fonte.

> Nota*: Você pode querer saber porque os ícones das classes não estavam com a lista quando o projeto foi aberto a primeira vez. Isto ocorreu porque as classes já tinham sido compiladas anteriormente. Normalmente os projetos distribuídos com o BlueJ não são previamente compilados, portanto você deve encontrar a partir de agora vários ícones de classes com listas.*

Na toolbar, no topo do editor de código fonte, existem alguns botões que corresponde as funções mais utilizadas. Uma delas é o *Compile*. Esta função permite compilar a classe diretamente. Click no botão *Compile*. Se você não cometeu nenhum erro, uma mensagem será mostrada na área na parte de baixo da janela do editor informando que a classe foi compilada com sucesso. Se você cometeu algum erro de sintaxe, a linha de erro será destacada e uma mensagem com a descrição do erro será mostrada na área de informação. (Se a sua primeira compilação ocorreu com sucesso, tente introduzir algum erro de sintaxe agora – como a ausência de um ponto-e-vírgula – e compile novamente para você como os erros de sintaxe são indicados).

Após você ter compilado a classe com sucesso, feche o editor.

> Nota: *Não é necessário salvar explicitamente o código fonte de sua classe. Os códigos fontes são automaticamente salvos no momento mais apropriado (por exemplo,*

quando o editor é fechado ou quando uma classe é compilada). Você, se quiser, pode explicitamente salvar (existe esta função no menu da janela do editor) seu código, mas isto só é necessário se o seu ambiente estiver instável e você estiver preocupado com perder parte do trabalho.

A toolbar da janela do projeto também possui o botão *Compile*. Este botão aciona a compilação de todas as classes do projeto (na verdade, ele determina quais as classes precisam ser recompiladas e recompila-as na ordem correta). Tente fazer isto mudando duas ou mais classes (de modo a ver duas ou mais classes com as listas no diagrama de classes) e então acione o botão *Compile*. Se algum erro for detectado em uma das classes compiladas, o editor será aberto e a localização do erro será indicada.

Você pode observar que a bancada de objetos está vazia outra vez. Os objetos serão sempre removidos quando a sua classe for recompilada.

3.7 Ajuda nos erros de compilação

Sumário: Para obter ajuda com as mensagens de erro durante a compilação, click no sinal de interrogação que está próximo da mensagem de erro.

Muito freqüentemente, alunos iniciantes têm dificuldade de compreender as mensagens de erro de compilação. O BlueJ tentará prover alguma ajuda. Abra o editor novamente, introduza um erro no código fonte e compile. Uma mensagem de erro deverá aparecerá na área de informação. No lado direito da área de informação aparecerá uma exclamação, se você clicar neste ícone você obterá mais informações sobre o tipo de erro descrito (Figura 8).

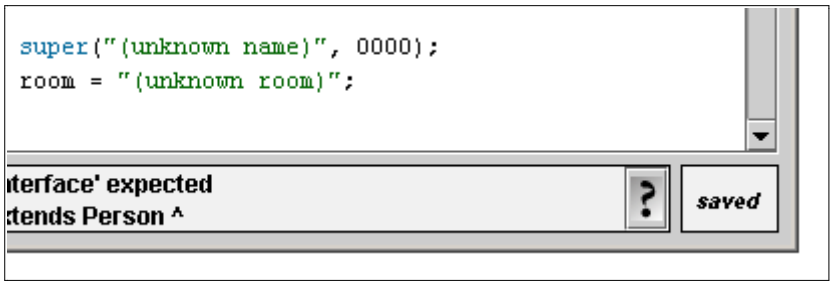

Figura 8: Um erro de compilação e o botão de *Help*

Neste estágio, os textos de ajuda ainda não estão disponíveis para todas as mensagens de erro. Muitos deles ainda devem ser escritos. Apesar disto vale a pena tentar pois muitos já foram cadastrados. Os restantes serão disponibilizados nas versões futuras do BlueJ.

4 Fazendo um pouco mais...

Nesta seção, nós investigaremos mais algumas coisas que você pode fazer com o ambiente. Coisas que não são essenciais, mas são comumente usadas.

4.1 Inspecionar

Sumário: inspecionar objetos permite um nível de depuração através da verificação dos estados internos do objeto.

Quando você executou métodos de um objeto, você pode ter observado que a operação *Inspect* estava disponível junto com os métodos definidos pelo usuário(Figura 5). Esta operação permite verificar os estado das variáveis de instância ("fields") de um objeto. Tente criar um objeto com alguns valores definidos pelo usuário (por exemplo, um objeto *Staff* com parâmetros para o método construtor). Selecione então o *Inspect* no menu do objeto. Um diálogo mostra os campos do objeto, seus tipos e seus valores (Figura 9).

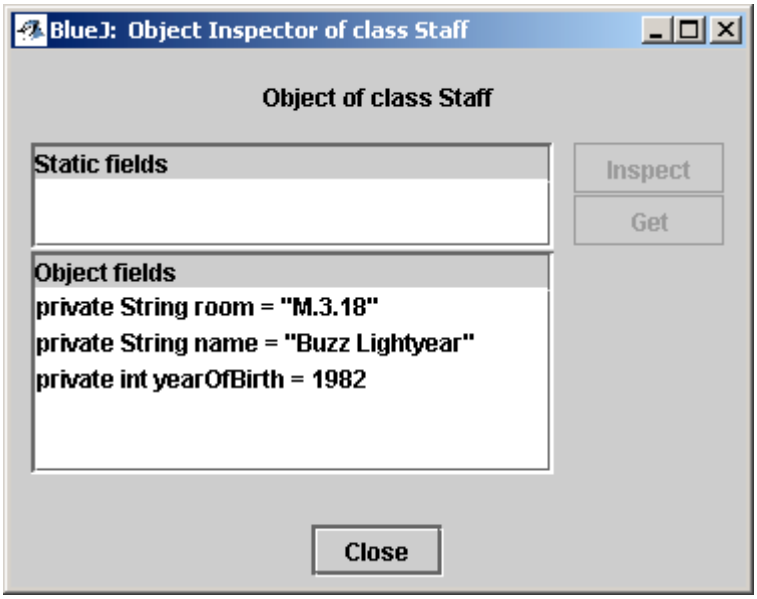

Figura 9: Dialogo de inspeção

Inspeção é útil por ser uma forma rápida de verificar se uma "mutator operation" (uma operação de mudança de estado do objeto) foi executada corretamente. Assim, a inspeção é uma ferramenta de verificar e eliminar erros simples.

No exemplo do objeto *Staff*, todos os atributos são de tipos simples (não são objetos e nem Strings). O valor destes tipos pode ser mostrado diretamente. Você pode imediatamente ver se o construtor fez ou não as atribuições corretamente.

Nos casos mais complexos, os valores dos campos podem ser referencias para objetos definidos pelo usuário. Para ver um exemplo, nós vamos usar um outro projeto. Abra o projeto chamado *people2*, que está incluído na distribuição padrão do BlueJ. O *people2* pode ser visto na Figura 10. Como você pode ver, este segundo exemplo possui uma

classe *Address* que não existia no projeto anterior. Um dos campos da classe *Person* é um atributo do tipo *Address*.

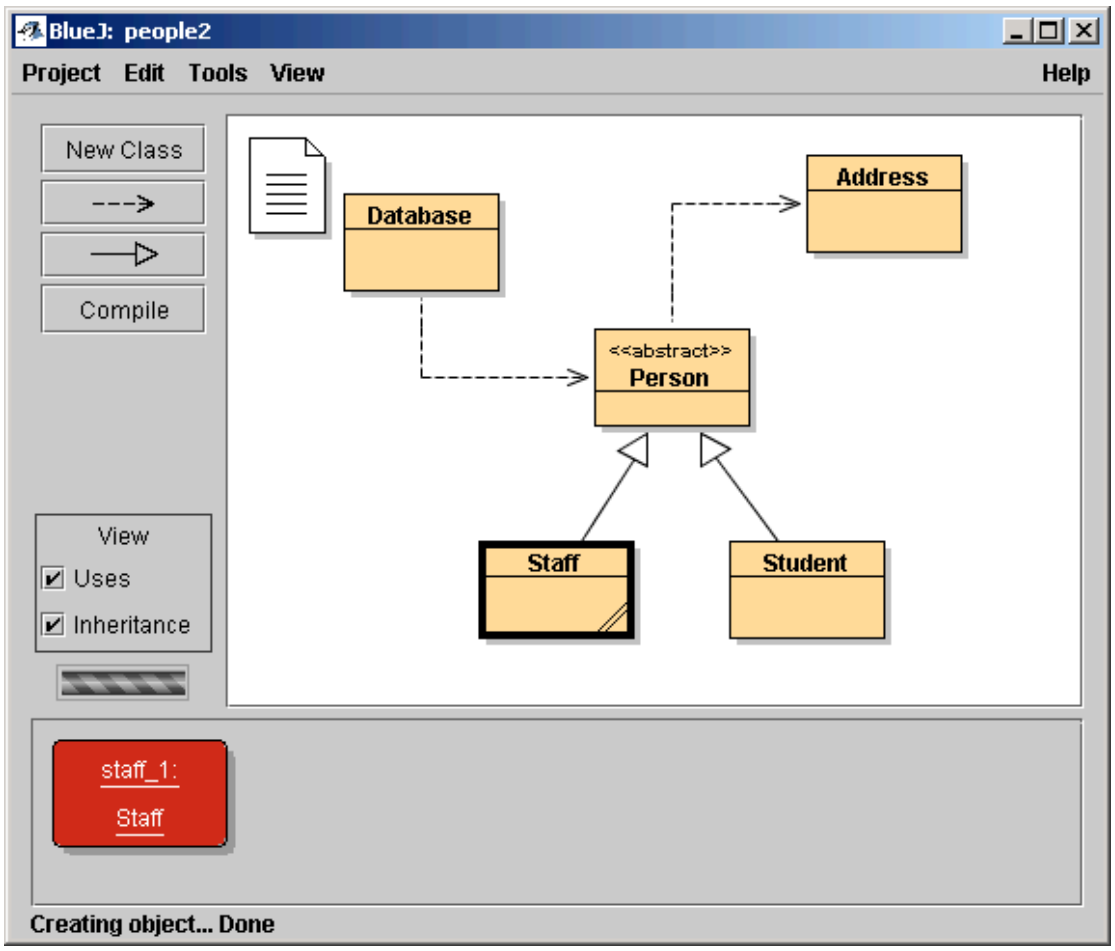

Figura 10: A janela do projeto *people2*

Para o nosso próximo teste – inspecionar o valor de um objeto que é atributo de outro – crie um objeto *Staff* e então acione o método *setAddress* deste objeto (você encontrará o método no menu popup do objeto *Person*). Entre com o endereço. Internamente o código do objeto *Staff* cria um objeto da classe *Address* e armazena-o no atributo *address*.

Agora, inspecione o objeto *Staff*. O resultado desta inspeção está na in Figura 11. Nos atributos do objeto *Staff* temos agora o *address*. Como você pode ver, o valor dele é mostrado como <*object reference*> pois ele é um objeto complexo e definido pelo usuário e seu valor não pode ser mostrado diretamente na lista. Para examinar o endereço, selecione o campo *address* na lista e click o botão *Inspect* na janela de diálogo. (Você também pode dar um duplo-click no campo *address*) Uma outra janela de inspeção será aberta e ela deverá mostrar em detalhes o objeto *Address* (Figura 12).

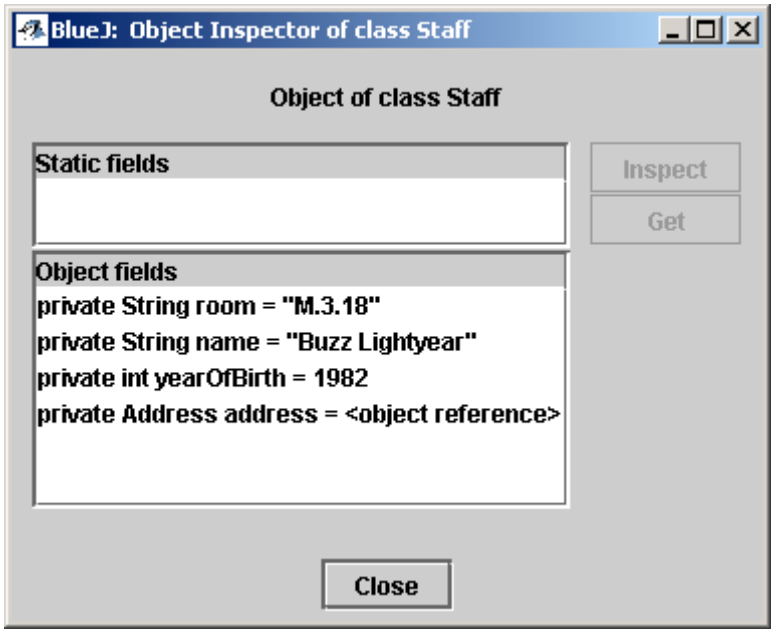

Figura 11: Inspeção com referencia de objeto

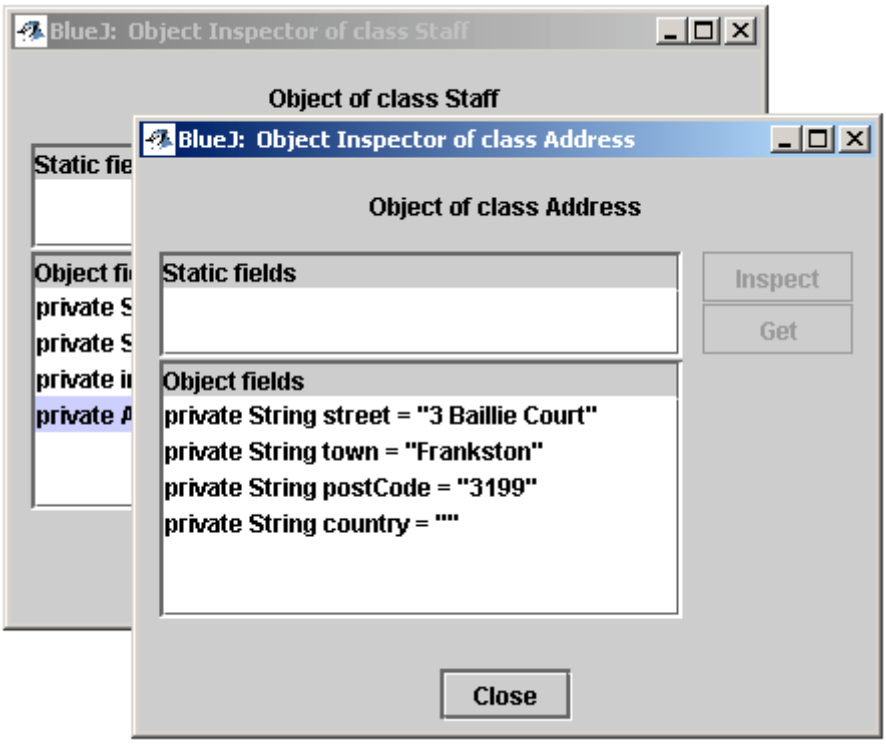

Figura 12: Inspeção em objetos internos

Se o campo selecionado for público, em vez de clicar *Inspect*, você também pode selecionar o campo *address* e clicar no botão *Get*. Está operação coloca o objeto na bancada de objetos. Estando lá você pode examiná-lo e fazer chamadas aos métodos dele.

4.2 Passando objetos como parâmetros

Sumário: Um objeto pode ser passado como um parâmetro para uma chamada de método.

Objetos podem ser passados como parâmetros aos métodos de outros objetos. Vamos tentar um exemplo disto. Crie um objeto da classe *Database* (Você perceberá que a classe tem somente um constructor e sem parâmetros, ou seja é só acionar o construtor). O objeto *Database* tem a habilidade de armazenar uma lista de pessoas. Ele possui operações para adicionar objetos e para mostrar os objetos guardados por ele (Na verdade, chamar este objeto de *Database* é um grande exagero !).

Se você não tiver objetos do tipo *Staff* e *Student* na sua bancada de objeto, crie também um de cada. Para prosseguir, você precisa ter um objeto *Database,* um *Staff* e um *Student* na sua bancada de objetos ao mesmo tempo.

Agora acione o método *addPerson* do objeto *Database*. A assinatura informa que um parâmetro do tipo *Person* é esperado. (Lembre-se: A classe *Person* é abstrata, então não existe nenhum objeto que seja diretamente do tipo *Person*. Mas, por causa da herança, os objetos *Student* e *Staff* podem substituir o objeto *Person*. Desta forma, é correto passar *Student* ou *Staff* onde *Person* é esperado). Para passar um objeto, que você possui na bancada de objeto, como parâmetro na chamada de método, você pode entrar com o nome do objeto no campo para passagem de parâmetro. Também é possível utilizar um atalho, através de um click no objeto na bancada de trabalho, para obter o nome do objeto em questão. Após isto click no botão ok para processar a chamada do método. Como ele não retorna valor, você não percebera o resultado imediatamente. Você deve acionar o método *listAll* no objeto *Database* para verificar se a operação foi realmente efetuada. A operação *listAll* escreve as informações sobre as pessoas na janela de saída padrão. Você verá esta janela aparecer automaticamente com o referido conteúdo.

Tente isto novamente após ter inserido mais algumas pessoas no “database”.

5 Criando um novo projeto

Este capítulo leva você a um rápido passeio na atividade de criar e configurar um novo projeto.

5.1 Criando o diretório do projeto

Sumário: Para criar um projeto, selecione New... *no menu* Project.

Para criar um novo projeto, selecione Project – New... no menu do BlueJ. Um diálogo de seleção de arquivo será aberto para você especificar o nome e a localização do novo projeto. Tente fazer isto. Você pode escolher qualquer nome para o seu projeto. Após você clicar no botão OK, um novo diretório será criado com o nome que você especificou e a janela principal mostrara o novo, e vazio, projeto.

5.2 Criando as classes

Sumário: Para criar uma nova classe, click no botão New Class *e especifique o nome da classe.*

Você pode criar suas classes clicando no botão *New Class* na barra de ferramentas. Você deverá preencher o campo com o nome da classe – este nome deve ser um nome válido para uma classe em Java.

Durante a criação da classe, você pode escolher entre quatro tipos de classes: abstrata, interface, applet ou padrão. Esta escolha determina qual o template será utilizado para criar sua classe. Você pode alterar esta definição através da edição do código fonte (por exemplo, adicionando a palavra "abstract" no seu código fonte).

Após criar uma classe, ela passa a ser representada por um ícone no diagrama. Se ela não for uma classe padrão, o tipo (interface, abstract, ou applet) é indicado no ícone. Quando você abre um editor para uma nova classe você observará que um template com a sua classe foi criado – este é um bom inicio para começar o trabalho. O código criado está sintaticamente correto. Ele pode ser compilado (apesar de não fazer nada). Tente criar algumas classes e compilá-las.

5.3 Criando as dependências

Sumário: Para criar uma seta de dependência, click no botão com a seta e araste-a para o diagrama, ou simplesmente codifique o que ela representa no editor de código.

O diagrama de classes mostra as dependências entre as classes na forma de setas. Relações de herança ("extends" ou "implements") são mostrados como setas (em triangulo); relações do tipo "uses" são mostradas como setas simples.

Você pode adicionar as dependências de forma gráfica (diretamente no diagrama) ou de forma textual no código fonte. Se você adicionar a seta no diagrama, o código é automaticamente atualizado; se você adicionar a dependência no código, o diagrama também será atualizado.

Para adicionar uma seta no diagrama, click no botão apropriado (setas duplas para "extends" ou "implements", seta simples para "uses") e arraste a seta de uma classe para a outra classe.

Adicionando uma seta de herança provoca a inserção da definição de "extends" ou "implements" na definição do código fonte da classe (dependendo de qual tipo é o código - classe ou interface).

Adicionar uma seta do tipo "uses" não altera imediatamente o código fonte (a menos que o alvo seja uma classe de um outro pacote. Neste caso é gerado um comando importe, mas nós não veremos isto nos nossos exemplos). Ter uma seta do tipo "uses" no diagrama apontando para uma classe que não possui o código fonte correspondente gerará uma mensagem de aviso indicando que a relação do tipo "uses" foi declarada mas nunca foi usada.

Adicionar a seta no código fonte é muito simples: basta digitar o código que você normalmente escreveria. Assim que a classe for salva, o diagrama será atualizado (lembre-se sempre: fechando o editor sua classe será salva).

5.4 Removendo elementos

Sumário: Para remover uma classe, selecione a função remover no menu popup da própria classe. Para remover uma seta, selecione remove no menu Edit *e click na seta a ser removida..*

Para remover uma classe do diagrama, selecione a classe e escolha *Remove Class* no menu *Edit*. Você também pode selecionar *Remove* no menu popup da classe em questão. Para remover a seta, selecione *Remove Arrow* no menu a então selecione a seta que você deseja remover.

6 Depurando

Esta seção introduz um dos aspectos mais importantes – a funcionalidade de depuração no BlueJ. Nas conversas entre os professores de computação, nós sempre escutamos que usar os depuradores no primeiro ano de ensino da computação seria muito bom, mas normalmente não temos tempo suficiente introduzir o assunto. Os estudantes esforçam-se com o editor, como a compilação e com a execução; Normalmente não temos tempo para introduzir mais uma ferramenta e com complexidade.

Por este motivo decidimos criar um depurador o mais simples possível. A meta foi ter um depurador que pudesse ser explicado em cerca de 15 minuto, e que os estudantes pudessem utilizá-lo sem muitas outras instruções. Vamos ver se tivemos sucesso.

Primeiramente, nos reduzimos as tradicionais funcionalidades dos depuradores em três diferentes tarefas :

- Marcar pontos de parada no código
- Caminhar passo-a-passo no código
- Verificar os valores das variáveis

Desta forma, cada uma das três tarefas são muito simples. Nos vamos testar cada uma delas.

Para iniciar, abra o projeto *debugdemo*, ele está incluído no diretório *examples* da distribuição padrão. Este projeto contem algumas classes com o propósito de demonstramos como é a funcionalidade de depuração – Este exemplo não faz muito sentido em nenhum outro contexto.

6.1 Marcando pontos de parada

Sumário: Para marcar um ponto de parada, click na área de breakpoint, parte esquerda do editor de texto.

Ter um ponto de parada marcado permite a você interromper a execução do programa em uma certa linha de código. Quando a execução é interrompida, você pode investigar o estado de seus objetos. Isto ajuda muito na compreensão do que está acontecendo no seu código fonte.

No editor de texto, no lado esquerdo do texto, existe a área de marcação de breakpoint (Figura 13). Você pode criar um breakpoint através de um simples click nela. Um pequeno sinal de stop aparecerá indicando o breakpoint. Tente fazer isto agora. Abra a classe *Demo*, procure pelo método *loop*, e marce um breakpoint em alguma linha dentro do loop *for*. O sinal deve aparecer no seu editor de código.

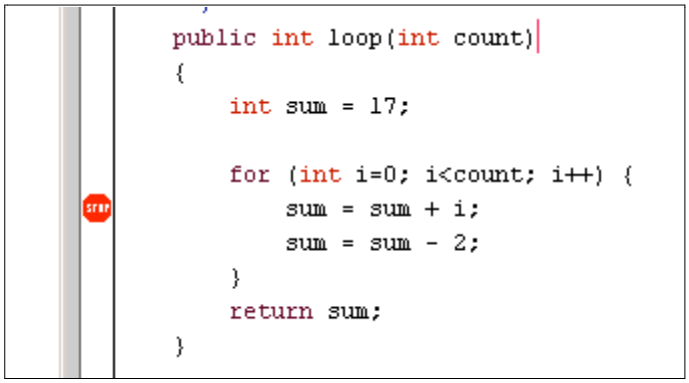

Figura 13: Um ponto de parada - breakpoint

Quando a linha de código associada ao breakpoint for ser executada, a execução será paralisada. Vamos tentar isto agora.

Crie um objeto da classe *Demo* e chame o método *loop* com um parâmetro, por exemplo 10. Quando a linha que contem o breakpoint estiver para ser executada, a janela do editor será mostrada, mostrando a linha de código corrente, e a janela do depurador também será mostrada. Isto será parecido com o que é mostrado na Figura 14.

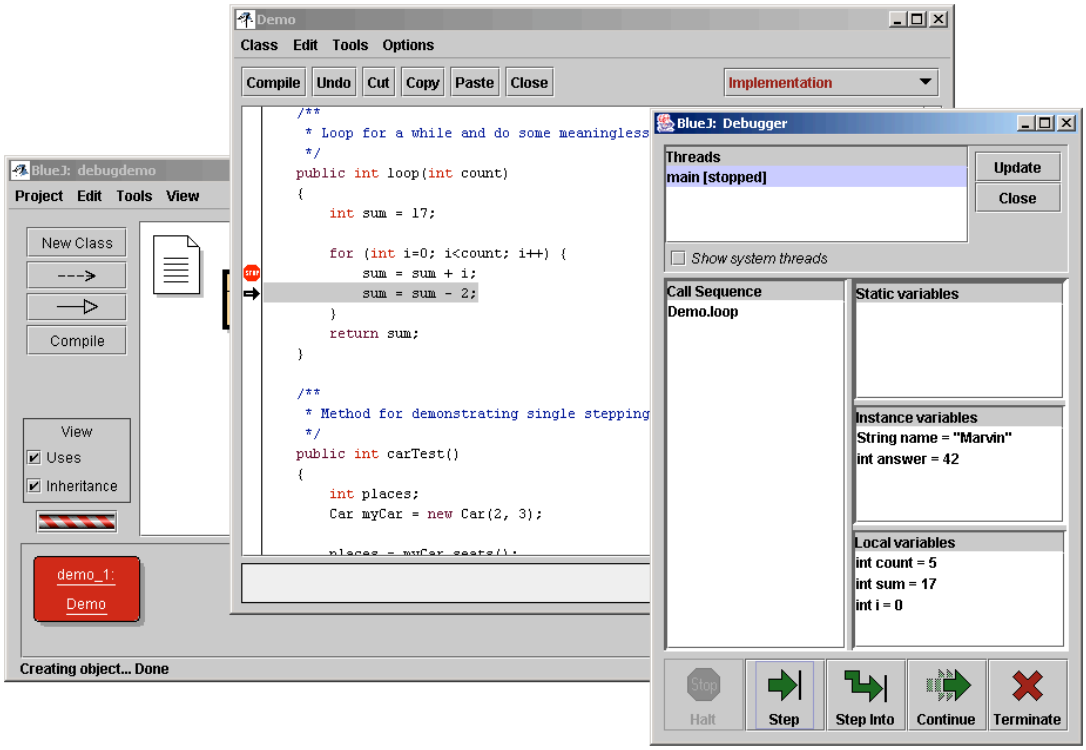

Figura 14: A janela do depurador

A linha em destaque no editor mostra qual linha será executada a seguir (a execução é parada antes da linha ser executada).

6.2 Caminhando passo-a-passo no código

Sumário: Para um passo simples no seu código, use os botões Step *e* Step Into *no depurador.*

Neste momento nos temos o código paralisado de sua execução (que nos convence realmente que o método foi executado até o ponto que tínhamos marcado), e nós podemos andar em passos-simples (single-step) pelo código e ver o progresso da execução do código. Para isto, click repetidamente no botão *Step* na janela do depurador. Você deve ver que a marca no editor está movimentando (a marca move através das linhas que vão sendo executadas). Toda vez que você clica no botão *Step*, uma linha de código é executada e a execução é paralisada novamente. Perceba também que os valores das variáveis mostradas na janela do depurador mudam (por exemplo, o valor da variável *sum*). Assim, você pode executar passo-a-passo o código e ver o que acontece. Quando você já estiver satisfeito, você pode remover o breakpoint clicando no ícone. Para continuar a execução, click no botão *Continue* e a execução continuará normalmente.

Vamos tentar novamente com outro método. Marque o breakpoint na linha abaixo da classe *Demo*, método *carTest()*

```
places = myCar.seats();
```

Acione o método. Quando o breakpoint for encontrado, você estará no momento antes da chamada do método *seats()* da classe *Car*. Com um click em *Step*, você terá a linha executada sem entrar no detalhe do método. Vamos tentar agora o *Step Into*. Se você acionar o *step into*, você entrará no detalhe da execução do método e poderá executar linha a linha (diferente do passo-simples). Neste caso, você verá a execução do método *seats()* de dentro da classe *Car*. Você poderá passear pelo método até que você encontre o comando end e retorne para o método chamador. Perceba como a janela do depurador se altera.

Step e *Step Into* comportam-se da mesma forma se a linha corrente não contem uma chamada de método.

6.3 Inspecionando as variáveis

Sumário: Inspecionar variáveis é facil – Elas são automaticamente mostradas no depurador.

Quando você depura erros no seu código, é importante poder inspecionar o estado de seus objetos (variáveis locais e variáveis da instância).

Fazer isto é trivial – a maior parte disto voce já viu. Você não necessita de comandos especiais para inspecionar as variáveis; variáveis estáticas, variáveis de instancia de um objeto corrente e variáveis locais de um método corrente são sempre automaticamente mostradas e atualizadas.

Você pode selecionar os métodos na pilha de chamada de métodos para ver as variáveis de outros objetos ativos ou de outros métodos. Tente, por exemplo, um ponto de parada no método *carTest()* novamente. No lado esquerdo da janela do depurador, você pode ver a pilha de chamada dos métodos. Ela atualmente mostra :

```
Car.seats
Demo.carTest
```


Isto indica que `Car.seats` foi chamado por `Demo.carTest`. Voce pode selecionar `Demo.carTest` na lista para inspecionar o código e os valores atuais das variáveis deste método.

Se você passar pela linha que contem o comando `new Car(...)` você pode observar que o valor da variável local `myCar` foi mostrado como *<object reference>*. Todos os valores do tipo objeto (exceto as Strings) são mostrados desta forma. Você pode inspecionar os valores das variáveis deste objeto através de um duplo-click. Fazendo desta forma uma janela de inspeção idêntica as já descritas anteriormente será aberta (seção 4.1). Não há nenhuma diferença entre a inspeção de objetos descritas aqui da inspeção dos objetos feitas na bancada de objetos.

6.4 Halt e terminate

Sumário: Halt *e* Terminate *podem ser utilizados para parar uma execução temporariamente ou permanentemente.*

Às vezes um programa fica rodando por muito tempo, e você gostaria de saber se a execução está sendo feita corretamente. Talvez exista um laço infinito, ou somente demora por algum motivo. Bem, nós podemos verificar. Chame o método *longloop()* da classe *Demo*. Ele ficará rodando.

Agora nos queremos saber o que está acontecendo. Mostre a janela do depurador, se ela já não estiver disponível (a propósito, selecionando o símbolo de turning, que mostra que o computador está processando, é um atalho para a chamada do depurador.)

Agora selecione o botão *Halt*. A execução será interrompida como se nós tivéssemos marcado um ponto de parada. Você pode agora passar os comandos, observar as variáveis, e verificar se está tudo dentro do esperado – e eventualmente continuar executando por mais algum tempo para completar a atividade. Você pode simplesmente selecionar *Continue* e *Halt* varias vezes para ver quanto rápido está sendo o processamento. Se você não quiser terminar esta execução (por exemplo, você descobriu que está no meio de um loop infinito) você pode simplesmente selecionar *Terminate* para parar a execução que está em andamento. O botão Terminate não deve ser usado com freqüência – você pode deixar o software em um estado não previsto e inconsistente, portanto é recomendável que seja usado somente como um mecanismo de emergência.

7 Criando aplicações stand-alone

Sumário: Para criar aplicações stand-alon, use o menu Project - Export...

BlueJ pode criar arquivos executáveis no formato jar. Os arquivos executáveis jar podem ser executados através de um duplo click (por exemplo no Windows e no MacOS X), ou através da linha de comando `java -jar <file-name>.jar` (no prompt do Unix ou do DOS).

Nos tentaremos fazer isto com o projeto *hello*. Abra-o (ele está no diretório *examples*). Tenha certeza que o projeto foi compilado. Selecione o menu *Project -> Export...*

Será aberto uma janela que permite a você especificar o formato de armazenamento (Figura 15). Escolha a opção "jar file" para criar um arquivo executável do tipo jar. Para que o executável seja efetivo, você deve ter criado uma classe de inicio de execução. Esta classe deve ter um método válido chamado *main* definido. (com a assinatura `public static void main(String[] args)`).

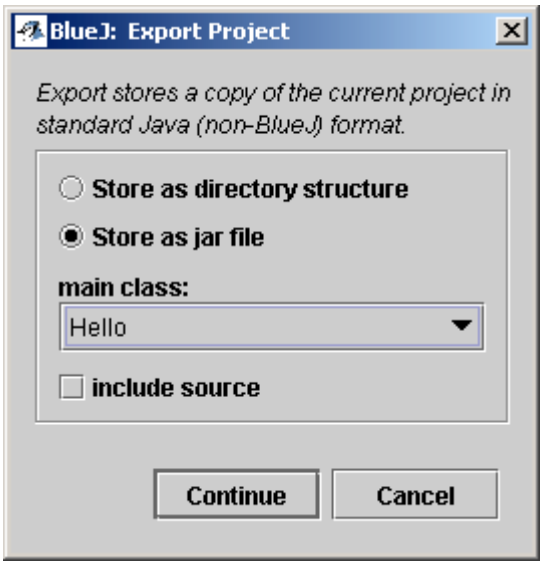

Figura 15: A janela de "Export"

Em nosso exemplo, escolher a classe principal é fácil; há somente uma classe. Selecione *Hello* no menu popup. Se você tiver outros projetos, selecione a classe que contém o método "main" que você deseja executar.

Geralmente, você não incluiria seus fontes em um arquivo executável. Mas você pode fazê-lo, se você quiser distribuir os fontes também.

Selecione o botão *Continue*. Em seguida você verá a janela para escolha de arquivo que permite que você especifique o nome do arquivo do tipo jar a ser criado. Digite *hello* e selecione o botão OK. A criação do arquivo executável do tipo jar foi completada.

Você pode executar o arquivo jar somente se a aplicação utiliza uma interface gráfica. Nosso exemplo usa a interface baseada em texto, portanto nos temos que aciona-lo em uma janela com prompt de comando. Agora tente executar o arquivo jar

Abra um terminal ou uma janela de prompt de comando. Então vá ao diretório onde você salvou o arquivo jar. (você deve verificar se o diretório contém o arquivo *hello.jar*). Supondo que o java foi instalado corretamente em seu sistema, você deve ser capaz de realizar o seguinte comando :

```
java -jar hello.jar
```

para executar o arquivo.

8 Criando applets

8.1 Executando uma applet

Sumário: Para executar uma applet, selecione Run Applet *no menu popup da classe.*

BlueJ permite criar e executar applets como as outras aplicações. Nós incluímos um applet no diretório de exemplos da distribuição do BlueJ. Primeiro, Nos tentaremos executar o applet. Abra o projeto *appletdemo* que está dentro do diretório exemples.

Voce verá que este projeto possue somente uma classe; ela é chamada de *CaseConverter*. O ícone que representa a classe foi marcado (com a tag <<applet>>) como sendo uma applet. Após a compilação, selecione *Run App*no menu popup da classe.

Uma janela sera exibida e ele permique que voce efetue algumas escolhas (Figura 16).

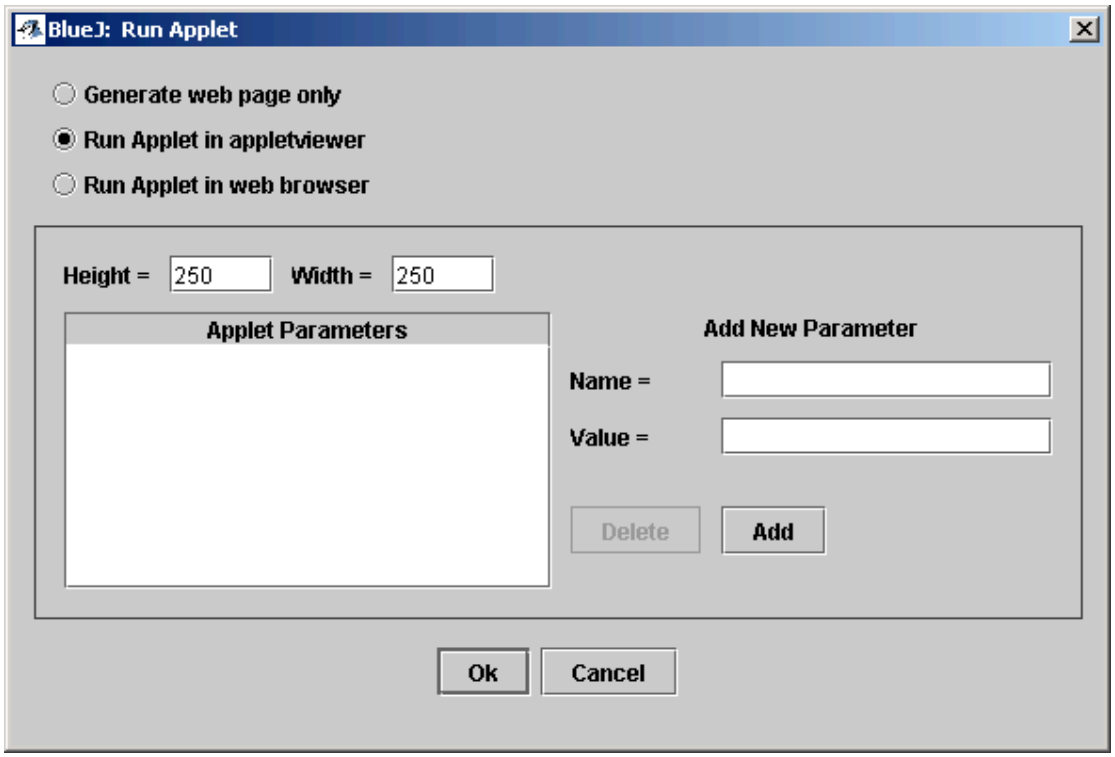

Figura 16: A janela "Run Applet"

Você verá que pode escolher entre rodar o applet em um browser ou em um visualizador de applets (ou simplesmente gerar uma página web sem executá-lo). Deixe as opções com os valores já selecionados e selecione o botão *OK*. Após alguns segundos, o visualizador applet será exibido e mostrará o applet case converter.

O visualizador applet foi instalado junto com o seu JDK, portanto é muito provável que ele seja da mesma versão do seu compilador Java. Isto geralmente evita os problemas que a execução em browsers. O seu browser pode possuir uma versão diferente de Java, e,

dependendo da versão que o seu browser estiver utilizando, pode causar problemas. Apesar disto, a maior parte dos browser deve funcionar corretamente.

Nos sistemas Microsoft Windows e MacOS, o BlueJ usa o browser default. Nos sistemas Unix systems, o BlueJ possui em suas configurações qual será o browser a ser acionado.

8.2 Criando um applet

Sumário: Para criar um applet, selecione o botão New Class *e escolha o tipo da classe como* Applet.

Após ter visto como executar uma applet, nós vamos ver como criar uma.

Crie uma nova classe do tipo *Applet* (você pode selecionar o tipo na janela de criação da classe). Compile e depois execute-a. É só isto ! Isto não foi difícil, não é ?

Applets (como as outras classes) são geradas com um corpo padrão que contém um código básico e válido. Para applets, este código mostra um applet simples com duas linhas de texto. Você pode abrir o editor de texto e editar a applet para inserir suas próprias linhas de código.

Você verá que todos os métodos mais comuns de uma applet já estão construídos, cada um com um comentário que explica o propósito do método. O código exemplo de construção dos textos está no método *paint*.

8.3 Testando um applet

Em algumas situações pode ser util criar um objeto applet na bancada de objetos (como as classes normais). Você pode fazer isto – o construtor é mostrado no menu popup da classe applet. Da bancada de objetos você não pode executar uma applet na sua forma completa, mas você pode chamar alguns métodos. Isto pode ser útil para realizar testes unitários nos métodos que você escreveu como sendo parte da implementação da sua applet.

9 Outras Operações

9.1 Abrindo pacotes feitos em outro ambientes no BlueJ

Sumário: Pacotes feitos em outros ambientes podem ser abertos no BlueJ atraves do menu Project: Open Non BlueJ...

BlueJ permite você abrir pacotes já existentes e que foram criados fora do ambiente do BlueJ. Para fazer isto, selecione Project – Open Non BlueJ... no menu. Selecione o diretório que contém os arquivos com os códigos fonte em java e selecione o botão Open in BlueJ. O BlueJ pedirá uma confirmação que você quer realmente abrir este diretório.

9.2 Inserindo classes já existentes no seu projeto

Sumário: Classes podem ser copiadas para um projeto utilizando o comando Add Class do menu File...

Em alguns casos, voce pode precisar usar uma classe que voce já possui em um projeto usando o BlueJ. Por exemplo, um professor pode fornecer uma classe Java aos estudantes para que eles usem em seus projetos. Você pode incorporar facilmente uma classe existente em seu projeto através do menu Edit – Add Class from File.... Isto selecionará o código fonte indicado (com a terminação do nome de arquivo em *.java*) para ser importado.

Quando uma classe é importada para um projeto, uma cópia é feita para o diretório corrente do projeto. Este efeito é exatamente igual a você criar uma classe e reescrever as linhas de código.

Uma alternativa a este processo é adicionar o código fonte, via outra ferramenta de organização de arquivo, da nova classe no diretório do projeto. A próxima vez que você abrir o projeto, a classe será incluída no diagrama do projeto.

9.3 Acionando o método *main* e outros métodos estáticos

Sumário: Métodos estáticos podem ser acionados a partir do menu popup da classe.

Abra o projeto *hello* a partir do diretório *examples*. A única classe no projeto (classe *Hello*) contém a definição do método main.

Click com o botão direito sobre a classe, e você verá o menu popup inclue não só os métodos construtores da classe, ele contém também o método estático *main*. Você pode chamar *main* diretamente a partir deste menu popup (sem primeiro criar um objeto, exatamente como é a definição de um método estático).

Todos os métodos estáticos podem ser acionados desta forma. O método padrão *main* espera receber um array com as Strings como sendo o argumento. Você pode passar um array com String usando o padrão de sintaxe Java para a declaração de array constantes. Por exemplo, você poderá passar

```
{"one", "two", "three"}
```

(incluindo as chaves) para o método. Tente fazer isto!

> Nota: *No padrão java, constantes no formato array constantes não podem ser usadas como parâmetros para a chamada de métodos. Eles podem somente ser utilizados como inicializadores. No BlueJ, para permitir a chamada interativa de métodos do tpo main, nós permitiremos a passagem de constantes no formato array como parâmetro.*

9.4 Gerando a documentação

Sumário: Para gerar a documentação de um projeto, selecione Project Documentation *no menu* Tools.

Você pode gerar a documentação do seu projeto no formato padrão *javadoc* a partir do BlueJ. Para fazer isto, selecione no Tools - Project Documentation menu. Esta função gerará a documentação para todas as classes do projeto a partir dos códigos fontes e abrirá um browser para mostrar a documentação gerada.

Você pode também gerar e visualizar a documentação de uma única classe diretamente a partir do editor de código fonte do BlueJ. Para fazer isto, abra o editor de código fonte e use o menu drop-down na barra de botões do editor. Altere a seleção de *Implementation* para *Interface*. Isto irá mostar a documentação no padrão *javadoc* (interface da classe) no editor.

9.5 Trabalhando com bibliotecas

Sumário: As classes das API padrão do java podem ter sua documentação visualizada através do Help - Java Standard Libraries.

Frequentemente, quando voce está escrevendo um programa em java, voce precisa utilizar a documentação das bibliotecas padrão do Java. Você pode abrir um browser mostrando a documentação da API do JDK através do menu Help - Java Standard Classes (se voce estiver na internet).

A documentação do JDK pode também ser instalada e usada localmente (offline). Os detalhes desta configuração estão explicados detalhadamente na seção de help no web site do BlueJ.

9.6 Criando objetos a partir da biblioteca de classes

Sumário: Para criar objetos a partir de uma biblioteca de classes, utize o menu Tools – Use Library Class.

O BlueJ também oferece a funcionalidade de criar objetos de uma classe que não faz parte do seu projeto, mas que esteja definida em uma biblioteca. Você pode, por exemplo, criar objetos de uma classe String ou ArrayList. Isto pode ser muito útil para simulações simples do uso destes objetos.

Você pode criar uma biblioteca através do menu Tools – Use Library Class.... Uma janela será exibida para você entrar o nome qualificado da classe, como *java.lang.String*. (Perceba que voce deve entrar com o nome complete, ou seja o nome da classe incluindo o nome dos pacotes que contém a classe.)

O campo de entrada possui um combo-box associado para mostrar as classes utilizadas recentemente. Uma vez que o nome da classe foi entrado, basta precionar *Enter* para verificar todos os construtores e os métodos estáticos da classe entrada. Qualquer um destes métodos, construtores ou métodos estáticos, podem ser invocados através da seleção deles na lista.

Este acionamento precederá qualquer outra chamada de método ou do construtor da classe.

10 Os sumários

Iniciando o BlueJ

1. Para abrir um projeto, selecione *Open* no menu *Project*.
2. Para criar um objeto de uma classe, selecione o construtor no menu popup da classe desejada.
3. Para executar um método, selecione-o através do menu popup.
4. Para editar o código fonte de uma classe, de um duplo-click no ícone da classe.
5. Para compilar uma classe, click no botão *Compile* no editor de texto. Para compilar um projeto, click no botão *Compile* na janela de visualização do projeto.
6. Para obter ajuda com as mensagens de erro durante a compilação, click no sinal de interrogação que está próximo da mensagem de erro.

Fazendo um pouco mais …

7. inspecionar objetos permite um nível de depuração através da verificação dos estados internos do objeto.
8. Um objeto pode ser passado como um parâmetro para uma chamada de método.

Criando um novo projeto

9. Para criar um projeto, selecione *New...* no menu *Project*.
10. Para criar uma nova classe, click no botão *New Class* e especifique o nome da classe.
11. Para criar uma seta de dependência, click no botão com a seta e araste-a para o diagrama, ou simplesmente codifique o que ela representa no editor de código.
12. Para remover uma classe, selecione a função remover no menu popup da própria classe.
13. Para remover uma seta, selecione remove no menu *Edit* e click na seta a ser removida.

Depurando

14. Para marcar um ponto de parada, click na área de breakpoint parte esquerda do editor de texto.
15. Para um passo simples no seu código, useos botões *Step* e *Step Into* no depurador.Inspecting variables is easy – they are automatically displayed in the debugger.
16. Inspecionar variáveis é facil – Elas são automaticamente mostradas no depurador.
17. *Halt* e *Terminate* podem ser utilizados para parar uma execução temporariamente ou permanentemente.

Criando aplicações stand-alone

18. Para criar aplicações stand-alon, use *Project - Export...*

Criando applets

19. Para executar uma applet, selecione *Run Applet* no menu popup da classe.
20. Para criar um applet, selecione o botão *New Class* e escolha o tipo da classe como *Applet* .

Outras Operações

21. Pacotes feitos em outros ambientes podem ser abertos no BlueJ atraves do menu *Project: Open Non BlueJ....*
22. Classes podem ser copiadas para um projeto utilizando o comando *Add Class do menu File...*
23. Métodos estáticos podem ser acionados a partir do menu popup da classe.
24. Para gerar a documentação de um projeto, selecione *Project Documentation* no menu *Tools*.
25. As classes das API padrão do java podem ter sua documentação visualizada através do *Help - Java Standard Libraries*.
26. Para criar objetos a partir de uma biblioteca de classes, utize o menu *Tools – Use Library Class*.