Exercícios 3.01 ao 3.18.

3.01 - Considere novamente o projeto 'lab-classes' que discutimos no Capítulo 1 e no Capítulo 2. Imagine que criamos um objeto 'Labclass' e três objetos 'Student'. Em seguida, registramos todos os três alunos nesse laboratório. Tente Desenhar um diagrama de classes e um diagrama de objetos para essa situação. Identifique e explique as diferenças entre eles.
R:

Diagrama de Classes

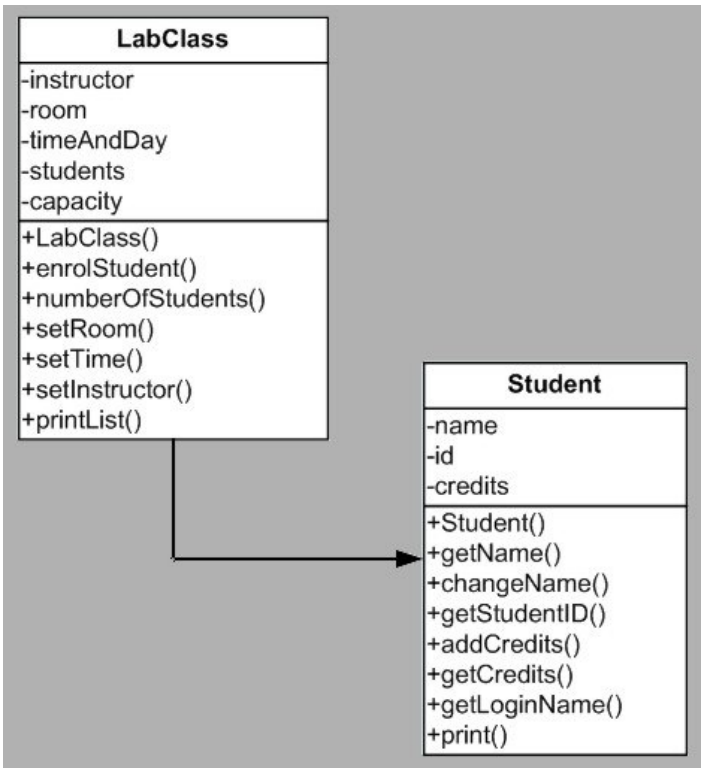
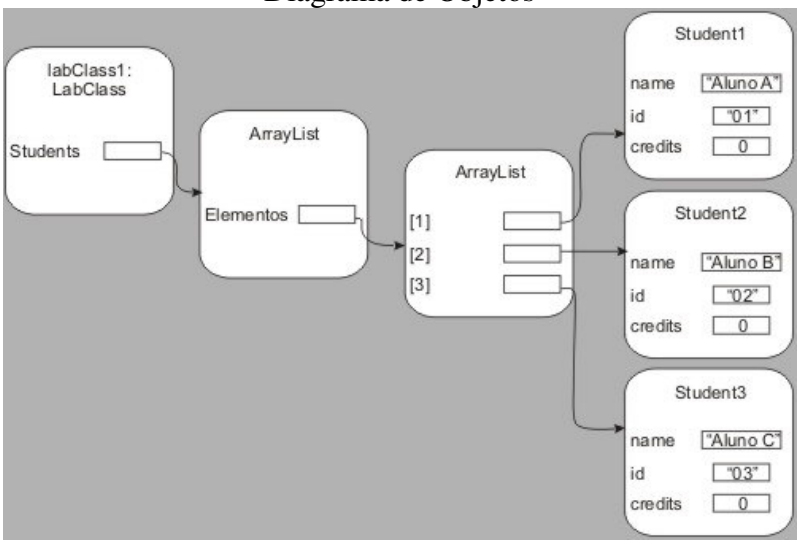


Diagrama de Objetos



3.02 - Inicie o BlueJ, abra o exemplo 'clock-display' e teste-o. Para utilizá-lo, crie um objeto 'ClockDisplay', em seguida, abra uma janela do inspetor para esse objeto. Com o inspetor aberto, chame os métodos do objeto. Observe o campo 'displayString' no inspetor. Leia o comentário do projeto (dando um clique duplo no ícone de nota de texto na tela principal) para obter informações adicionais.
R: A janela de Inspetor mostra o que ocorre com as variáveis utilizadas no programa.

3.03 - O que acontece quando o método 'setValue' é chamado com um valor inválido? Essa é uma boa solução? Você pode pensar em uma solução melhor?
R: Ele simplesmente não altera o valor ta variável 'value'.

3.04 - O que aconteceria se você substituisse o operador '>=' no teste por '>', de modo que ele fosse escrito assim:
if((replacementValue > 0) && (replacementValue < limit))
R: Se digitado, o valor 0 seria descartado no condicional.

3.05 - O que aconteceria se você substituisse o operador '&&' no teste por '||', de modo que ele fosse escrito assim:
if((replacementValue >= 0) || (replacementValue < limit))
R: O operador '&&' significa que ambas as condições devem ser verdadeiras. Assim, o valor de substituição tem que necessariamente ser maior ou igual a zero e também menor que o valor de limite. Se utilizarmos o operador '||', dizemos que apenas uma das condições devem ser verdadeiras. Assim, o valor pode ser maior ou igual a zero, mas não menor que o limite. Pode ser menor que o limite, mas não maior ou igual a zero. Ambas sendo verdadeiras, também é aceito.

3.06 - O método 'getDisplayValue' funciona corretamente em todas as circunstancias? Quais suposições são feitas nelas? O que acontece se você cria um mostrador de número limite 800, por exemplo?
R: O limite que se pode passar para o método é infinito, sendo que no relógio ele só possui duas casas. Se criarmos um limite de 800, não haverá nenhum bloqueio ou aviso do alto limite escolhido.

3.07 - Há alguma diferença no resultado se escrevermos:
return value + "";
em vez de:
return "" + value;
no método 'getDisplayValue?
R: Não, pois ele sempre vai transformar o tipo menos abrangente no mais abrangente. Se existem dois tipos: inteiro e String, o inteiro se torna string. Se existe um double e um inteiro, o inteiro vira double.

3.08 - Explique o operador de módulo. Você pode precisar consultar mais recursos (recursos de linguagem Java on-line, outros livros de Java, etc.) para descobrir os detalhes.

R: O operador '%' calcula o resto de uma divisão de numeros inteiros, armazenando o apenas em uma variável de tipo inteiro.

3.09 - Qual o resultado da expressão (8%3)?
R: 8 % 3 = 2

3.10 - Quais são todos os possíveis resultados da expressão (n%5), onde n é uma variável do tipo inteiro?
R: Possíveis resultados de n % 5 = {0, 1, 2, 3, 4}

3.11 - Quais são todos os possíveis resultados da expressão (n%m), onde n e m são variáveis do tipo inteiro?
R: Possíveis resultados de n % m, onde n e m são inteiros = {0,..., m-1}, sendo m <> 0

3.12 - Explique em detalhes como o método de incremento funciona.
R: Soma a variável 'value' atual 1, e armazena na variavel 'value', o resto dessa soma pelo valor da variável 'limit', isso nunca atingindo o valor de 'limit', ou seja, variando entre 0 e 'limit'.

3.13 - Reescreva o método de incremento sem o operador de módulo, utilizando uma instrução if. Qual é a melhor solução?
R:

```
public void increment()
{
        value = value +1;

        if (value >= limit)
        {
                value = 0;
        }
}
```

3.14 - Utilizando o projeto 'clock-display', no BlueJ, teste a classe NumberDisplay, criando alguns objetos 'NumberDisplay' e chamando seus métods.
R: Teste no BlueJ.

3.15 - Crie um objeto 'ClockDisplay' selecionando o seguinte construtor:
new ClockDisplay()
Chame seu método 'getTime' para descobrir a hora inicial em que o relógio fui ajustado. Você consegue imaginar por que ele foi iniciado nessa hora em particular?
R: Na criação do objeto clockdisplay, ele cria objetos numberdisplay, os quais apenas solicitam o 'limit' e configura 'value' como 0, por isso o relógio começa com os valores '00:00'.

3.16 - Quantas vezes você precisa chamar o método 'tick' em um objeto 'ClockDisplay' recém-criado para fazer sua hora alcançar 01:00? De que outra forma você faria ele exibir essa hora?

R: Teriamos que usar o método 60 vezes, ou usaria o método 'SetTime' e configuraria o relógio, passando como parâmetro '1' e '0'.

3.17 - Veja o segundo construtor no código-fonte de 'ClockDisplay'. Explique o que ele faz e como faz.
R: O método "pede" dois valores inteiros, horas e minutos, respectivamente, e após criar os objetos 'hours' e 'minutes', ele chama o método 'setTime' e passa como parâmetro, os parâmetros passados anteriormente para a configuração do relógio. No método 'setTime', ele passa para os objetos, no método 'setValue' os valores anteriormente passados.

3.18 - Identifique as semelhanças e as diferenças entre os dois construtores. Por que não há nenhuma chamada para 'updateDisplay' no segundo construtor, por exemplo?
R: O segundo primeiro construto não pede parametros para a configuração da hora, e já chama o método 'updateDisplay' para a configuração do horário '"00:00"', e o segundo chama o método 'setTime' para ajustar o relógio aos valores informados, e esse mesmo método chama o método 'updateDisplay', por isso essa chamada não se encontra no segundo método construtor.