IMD0040 Trabalho 1

Introdução

Este trabalho foi projetado para demonstrar seu entendimento de classes de bibliotecas, interação entre objetos e leitura de documentação (API). Você vai entregar o trabalho eletronicamente via SIGAA.

Qualquer método escrito por você deverá ser totalmente comentado utilizando tags javadoc.

Prazo de entrega: 23:59 27 de Agosto de 2018

Plágio e Duplicação de Material

O trabalho que você submeter deverá ser somente seu. Iremos rodar programas verificadores em todos os trabalhos submetidos para identificar plágio, e adotar as medidas disciplinares cabíveis quando for o caso. Algumas dicas para evitar problemas de plágio:

- Um dos motivos mais comuns para problemas de plágio em trabalhos de programação é deixar para fazer o trabalho de última hora. Evite isso, e tenha certeza de descobrir o que você tem que fazer (que não significa necessariamente como fazer) o mais cedo o possível. Em seguida, decida o que você precisará fazer para completar o trabalho. Isto provavelmente envolverá alguma leitura e prática de programação. Se estiver em dúvida sobre o que foi pedido pelo trabalho, pergunte ao professor da disciplina.
- Outra razão muito comum é trabalhar em conjunto com outros alunos da disciplina. Não faça trabalhos de programação em conjunto, ou seja, não utilizem um único PC, ou sentem lado a lado, principalmente, digitando código ao mesmo tempo. Discutam as diversas partes do trabalho, mas não submetam o mesmo código.
- Não é aceitável a submissão de código com diferenças em comentários e nomes de variáveis, por exemplo. É muito fácil para nós detectar quando isso for feito, e verificaremos esse caso.
- Nunca deixe outra pessoa ter uma cópia de seu código, não importando o quão desesperado eles possam estar. Sempre aconselhe alguém nesta situação a buscar ajudar com o professor da disciplina.

A Tarefa

Este trabalho consiste em diversas modificações que serão feitas ao projeto *balls* que é discutido no capítulo 5 do livro nas páginas 141-145. Não será necessário fazer qualquer alteração nas classes `Canvas` e `BouncingBall`, mas você deve se familiarizar com os cabeçalhos dos métodos de ambas as classes de maneira que você saiba como invocar os métodos oferecidos por estas classes. Você pode utilizar a visão de documentação do editor do BlueJ para tal.

A classe `BallDemo` contém um método (**`bounce`**) que realiza uma animação com duas bolas pulando. A classe `CanvasDemo` contém um método que ilustra uma variedade de chamadas à classe `Canvas` para desenhar em um objeto `Canvas`. Estude essas classes em detalhe para entender como se dá a interação com as classes `BouncingBall` e `Canvas`.

Neste trabalho, você adicionará algumas funcionalidades à classe `BallDemo`.

Desenhando um quadro

Adicione um método chamado **`drawFrame`** à classe `BallDemo`. Este método deverá exibir um quadro no objeto canvas desenhando um retângulo dentro da borda do canvas, a uma distância de 20 pixels. A classe CanvasDemo tem um exemplo de como usar os métodos da classe Canvas para desenhar um retângulo.

Você precisará descobrir o tamanho do Canvas e como utilizar um objeto da classe `Dimension` que é definido no pacote **java.awt**.

Perceba que os campos para a altura (length) e largura (width) de um objeto Dimension são valores inteiros públicos. Desse modo, eles podem ser acessados diretamente. Isso é mais fácil do que utilizar os respectivos métodos de acesso, já que os últimos retornam **double** e não **int**.

Uma vez que isso esteja funcionando, tenha certeza de que o método **`drawFrame`** funcione com qualquer tamanho de canvas (não escreva em código o tamanho padrão de um objeto canvas nesse método). Você pode testar isso redimensionando o objeto canvas e chamando o método **`drawFrame`** novamente. Perceba que seu quadro não precisa ser redesenhado automaticamente quando a janela é redimensionada, mas somente depois de uma chamada ao método **`drawFrame`**.

Tenha o cuidado de apagar quaisquer quadros antigos antes de desenhar um novo.

Documente o método **`drawFrame`** utilizando comentários javadoc.

Posição no solo

O solo de onde as bolas pulam é desenhado em uma posição fixa e com um tamanho fixo. Modifique isso de modo que o solo é desenhado um pouco acima da base do quadro, e que ele inicie e finalize dentro dos limites do quadro. Você pode decidir a distância dos limites em que a linha inicia/termina. A posição e tamanho precisam ser recalculadas toda vez que o método **`bounce`** é chamado. As bolas deveram continuar pulando no solo, e a animação só deverá parar quando todas as bolas chegarem do lado direito (ao fim do solo).

Mais bolas

Modifique o método **`bounce`** para que este agora receba um parâmetro inteiro que permita ao usuário escolher quantas bolas deverão ser incluídas na animação. Dentro do método, tenha certeza de que o parâmetro seja maior que zero antes de criar as bolas.

Utilize uma coleção para armazenar as bolas. Deste modo, o método pode lidar com uma, tres ou 75 bolas (qualquer número). As bolas deverão ser colocadas aleatoriamente em qualquer posição na metade superior do quadro. De novo, não assuma que o quadro tem um tamanho fixo, utilize o método getSize da classe Canvas para calcular onde a metade superior se encontra.

Cores aleatórias

Faça com que as bolas criadas pelo método **`bounce`** tenha cores aleatórias. A documentação da API para a classe `java.awt.Color` apresenta um número razoável de variáveis estáticas que podem ser usadas para as cores, mas você precisará armazenar elas em uma estrutura de dados apropriada

para permitir a seleção aleatória.

Finalmente

Antes de submeter seu trabalho, teste exaustivamente a classe inteira para se certificar de que nada do que foi acrescentado recentemente quebrou algo adicionado anteriormente. Se você não conseguir completar a classe - mesmo se você não conseguir compilar – submeta o que você tem, pois é provável que você vai ter pelo menos algum ponto por isso.

Carlos Eduardo Silva, 20/08/2018