EXERCÍCIO FINAL DA DISCIPLINA IOS 1

O objetivo deste exercício é aprimorar a sua calculadora de modo a criar um gráfico conforme o usuário vai fazendo os cálculos. O gráfico pode ser ampliado e movimentado usando gestos. O aplicativo deve funcionar em iPhone e em iPad, se adaptando à cada tela corretamente.

DO QUE VOCÊ VAI PRECISAR?

1. É necessário que você tenha completado todos os exercícios anteriores relacionados à calculadora;
2. É importante que você tenha feito os demais exercícios e exemplos.
3. É importante que você tenha cumprido todas as tarefas de leitura;
4. O professor vai fornecer uma classe chamada **AxesDrawer**, que vai ser muito útil neste exercício.

REQUISITOS OBRIGATÓRIOS

1. A classe **ViewController** dos nossos exemplos da calculadora deve ser renomeada para algo como **CalculatorViewController**, caso você ainda não tenha feito. Dica: lembre-se de ajustar o apontamento da classe no Storyboard;
2. Crie um novo MVC que gera um gráfico de uma função matemática. Você está livre para definir a API pública para o ViewController. Tenha muita cautela na hora de definir o Model;
3. O MVC que você criou para o item 2 deve ser um componente completamente genérico, desacoplado e reutilizável. Ele não deve ter, de maneira alguma, qualquer relação com o MVC da calculadora;
4. Como parte deste exercício, você vai precisar implementar uma subclasse de **UIView** que seja capaz de traçar um gráfico do tipo X vs Y. Esta classe deve ser genérica, desacoplada e reutilizável;
5. A view do item 4 não deve, em momento algum, armazenar o dado que está sendo exibido, nem mesmo temporariamente. Ela deve solicitar ao respectivo ViewController

o dado, ponto por ponto. Importante: esta view não desenha um array de pontos, sim uma função matemática;

6. Adicione um novo botão à sua interface da calculadora. Este novo botão deve provocar uma **segue** para o MVC que mostra o gráfico. O gráfico deve mostrar o que está no **CalculatorBrain** naquele instante. Por exemplo, se o **CalculatorBrain** contém uma função de seno, o gráfico deve ser uma onda senóide. Entradas subsequentes ao desenho do gráfico não devem alterá-lo, até que o botão seja pressionado novamente. O pressionar do botão deve ser ignorado caso o resultado esteja pendente (dica: **pendingBinaryOperation**);
7. No iPad e nos iPhones Plus o gráfico deve ser capaz de ser exibido lado-a-lado com a interface da calculadora. Nos demais iPhones, o botão do gráfico deve provocar um push usando o NavigationController;
8. Na tela do gráfico, deve existir uma descrição da função que está sendo exibida na forma de gráfico. Exemplo: caso esteja sendo desenhada uma onde senóide, a descrição deve ser “sin(10)”. Você decide onde colocar a descrição;
9. A view do gráfico deve ser **@IBDesignable** e a escala do gráfico deve ser **@IBInspectable**. Os eixos do gráfico devem aparecer no storyboard, conforme a escala definida;
10. O gráfico deve suportar os seguintes gestos:
    a. Pinça (para aplicar zoom no gráfico inteiro, incluindo nos eixos);
    b. Arrastar (movimenta o gráfico para qualquer direção, incluindo os eixos);
    c. “Duplo Clique” ou Double-Tap (move a origem do gráfico para o ponto onde ocorreu o gesto)
11. Você **não pode** adicionar nenhuma nova API pública no CalculatorBrain e você também **não pode** usar os tipos Any e AnyObject em ponto algum da solução (exceto no desafio 6)

DESAFIO (AUMENTA A NOTA EM 5% POR ITEM)

O desafio somente será considerado quando você cumprir **todos** os requisitos obrigatórios. Cada item atendido do desafio aumenta a sua nota em 3%. Cumprir todos os desafios significa uma nota 20% maior (se você tirar nota máxima, o excedente acumula para a disciplina iOS 2).

1. Fazer o botão de mostrar o gráfico dizer para o usuário que é ou não possível traçar um gráfico que está atualmente na calculadora. Você pode desativar o botão, mostrar um ícone diferente, mudar a cor, etc;
2. Manter a origem e a escala do gráfico entre as inicializações do seu aplicativo. Não há local exato para colocar a implementação deste item, lembrando que você deve respeitar o MVC sempre;
3. Quando acontecer rotação do dispositivo, manter a origem do gráfico relativa ao centro em vez de relativa ao canto superior esquerdo;
4. Quando seu aplicativo for iniciado, o MVC de gráfico deve exibir o último gráfico em vez de abrir em branco. Você pode modificar o estado do CalculatorBrain para atingir o objetivo. Tenha muito cuidado para não violar o MVC ao projetar sua solução. Dica: o mecanismo de persistência mais simples do iOS é o **UserDefaults**, mas ele somente aceita PropertyLists, então você vai precisar bolar uma maneira para guardar o estado neste formato e por causa disso você pode usar aqui e somente aqui os tipos Any e AnyObject.

CONTEÚDOS APLICADOS NO EXERCÍCIO

1. Entendimento do MVC;
2. Semântica de tipos de valor e tipos de referência;
3. Subclasses de UIViewController;
4. Aplicação Universal (iPad e iPhone);
5. Split View Controller;
6. Navigation Controller;
7. Segue;
8. Subclasses de UIView;
9. Redesenho de uma UIView;

10. Desenho usando UIBezierPath e tipos de dado associados (CGFloat, CGPoint, CGSize, CGRect);
11. Gestos;
12. Entendimento da diferença entre Pixels e Pontos.

CRITÉRIOS DE AVALIAÇÃO

Assim como em todos os demais exercícios, a qualidade do código será objeto de avaliação, assim como se seu código tem ou não alertas (warning) ou erros. O uso do aplicativo entregue também faz parte da avaliação – o maior objetivo de todos é que o app funcione sem qualquer problema na visão do usuário. Assim sendo, segue uma lista de **motivos para reduzir sua nota**:

1. O projeto não compila (reduz muito a nota – risco de reprovação);
2. O projeto compila, mas apresenta alertas (warnings);
3. Um ou mais itens obrigatórios não está presente;
4. Um conceito fundamental foi esquecido ou violado;
5. O código é bagunçado (difícil de ler, endentação ausente ou inconsistente);
6. A solução apresentada é de difícil compreensão (falta de comentários, nomes de variáveis/métodos que não dizem nada, métodos demasiadamente longos, má estruturação da solução);
7. A interface visual é um desastre. Os elementos visuais devem estar, no mínimo, bem alinhados e espaçados;
8. Não há correta distinção entre API pública e privada;
9. Houve violação do MVC (reduz muito a nota – risco de reprovação);
10. O aplicativo contém **retain cycle** (memory leak, memory cycle, etc).

PRAZO PARA ENTREGA

A entrega deve ser feita exclusivamente através do Blackboard, na data e horário nele estipulados. O item da entrega não irá aceitar entregas atrasadas, embora enquanto dentro do prazo são aceitas tentativas ilimitadas sendo o último envio o objeto a ser avaliado.