Suporte para GitHub:

Carregando arquivos pelo navegador: https://drive.google.com/open?id=1Klf0HCJcB_4O5B7efMXrs_YYDXGwH89U
Tutorial Basico GitHub com Eclipse e EGit Usando Chave SSH: http://www.youtube.com/watch?v=fFBSazTSGZw
Usando Github com Github Desktop em Projetos Eclipse: http://www.youtube.com/watch?v=EqHIjlYyS4U
Usando Github com SSH no Terminal Linux com chave gerada no Eclipse: http://www.youtube.com/watch?v=0s699q5Sja4
Usando Github com SSH no Terminal Windows com chave gerada no Eclipse: http://www.youtube.com/watch?v=DaydwPB2WSI

Vídeo Suporte:
Gerar Bibliotecas Java: https://youtu.be/9x3_c_0i6O0
Pilha Dinâmica: https://youtu.be/ahi8U7OnKco

Gerando link compartilháveis:
https://drive.google.com/file/d/1cyoxa5W67MY5xDM6gCYpklel1GU3QCKa

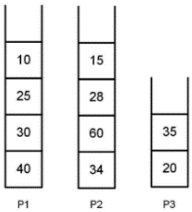
Para todos os exercícios, quando solicitado teste de mesa, carregar a solução para um drive compartilhado e quando solicitado desenvolvimento, definir o que se pede e aplicar o código em Java e carregar a solução no Github.

1. Simular o comportamento de pilhas dinâmicas para os algoritmos abaixo (A simulação deve deixar evidente a pilha que sobrou na memória):

```
a)
Para (i = 0 ; i < 10 ; i++) {
        Se (i % 2 == 0) {
                Push(i * i);
        } Senão {
                Se (i <= 5) {
                        Push(i);
                } Senão {
                        Imprimir(Pop());
                }
        }
        Imprimir(Top());
}
Imprimir(Size());
```

```
b)
Para (i = 100 ; i < 115 ; i++) {
        Se (isEmpty()) {
                Push(i + 100);
        } Senão {
                Se (Size() <= 4) {
                        Push(i + 50);
                } Senão {
                        Imprimir(Pop());
                }
        }
        Imprimir(Top());
}
Imprimir(Size());
```

2. Considere as pilhas iniciais já criadas e populadas



Admita que um método Java, chamado exibePilha, receba essas três pilhas como parâmetros e execute os seguintes passos (Fazer como teste de mesa):
1. Cria duas pilhas auxiliares, A1 e A2, inicialmente vazias;
2. Remove um elemento de P1 e o insere em A1. Em seguida, remove um elemento de P2 e o insere em A1. Repete esses dois procedimentos até que P1 e P2 fiquem, ambas, vazias;
3. Remove um elemento de P3 e o insere em A1. Repete esse procedimento até que P3 fique vazia;
4. Remove um elemento de A1 e o insere em A2. Repete esse procedimento até que A1 fique vazia;
5. Remove um elemento de A2 e o exibe no console. Repete esse procedimento 4 vezes.

O que será exibido no console e como ficarão as pilhas, quando o método exibePilha for executado?

3. Em Java:
a) Transformar o projeto em uma biblioteca de uma Pilha de Inteiros, gerando o JAR PilhaInt.
b) Adaptar o modelo de Pilha Dinâmica desenvolvido em aula, com os métodos esperados, para uma Pilha de Strings. Transformar o projeto em uma biblioteca, gerando o JAR PilhaStrings.

4. Implementar um novo projeto Java com base biblioteca PilhaInt que permita a conversão de decimais para binários, a qual se dá dividindo, sucessivamente, o valor de entrada por 2 e concatenando os restos da divisão do último para o primeiro.

O projeto deve ter uma classe ConverteController no package controller, que inicialize uma pilha de inteiros e com um método decToBin(int decimal): String, que, recebendo um número decimal e realizando as operações, irá inserindo os restos das divisões na pilha. Ao término do empilhamento, deverá ser feita a operação de desempilhar, concatenando cada número desempilhado (Convertendo para String) com o próximo, até a pilha esvaziar.

Deve-se ter também uma classe Principal no package view que permita ao usuário inserir um número decimal limitado a 1000.

5. Sabe-se que, em uma Pilha, não é possível conhecer os elementos, no entanto, é possível adaptar o modelo de fila para auxiliar os desenvolvedores. Modificar o código da PilhaInt para incluir um novo método chamado max, que apresenta o maior valor da pilha. Considere que a pilha tem operações destrutivas e esse método não pode mudar o conteúdo, tampouco a estrutura da pilha.