Suporte para GitHub:

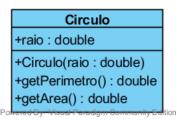
Carregando arquivos pelo navegador: https://drive.google.com/open?id=1Klf0HCJc8 405B7efMXrs YYDXGwH89U

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=EqHIjIYyS4U
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

Para cada exercício, criar um projeto no Github para submeter à tarefa.

Para cada exercício, criar uma classe no package view que se possa testar as implementações que se foram pedidas nos objetos

1. Crie uma classe chamada "Círculo" que possua um atributo para armazenar o raio e métodos para calcular a área e o perímetro do círculo. A criação do objeto e dos testes deve seguir a modelagem abaixo.



2. Implemente uma classe chamada "ContaBancária" que possua atributos para armazenar o número da conta, nome do titular e saldo. Adicione métodos para realizar depósitos e saques.

```
+numeroConta: int
+nome: String
+saldo: double
+ContaBancaria()
+realizaDeposito(valor: double)
+realizaSaque(valor: double)
```

3. Crie uma classe chamada "Funcionário" com atributos para armazenar o nome, o salário e o cargo do funcionário. Implemente métodos para calcular o salário líquido, considerando descontos de impostos e benefícios. Nenhum funcionário pode receber menos de 1 salário-mínimo R\$1412,00.

```
Funcionario
-nome: String
-salario: double
-cargo: String
+Funcionario()
+getNome(): String
+setNome(nome: String): void
+getSalario(): double
+setSalario(salario: double): void
+getCargo(): String
+setCargo(cargo: String): void
+getSalarioLiquido(descontos: double, beneficios: double)
```

4. Implemente uma classe chamada "Produto" que possua atributos para armazenar o nome, o preço e a quantidade em estoque. Adicione métodos para calcular o valor total em estoque e verificar se o produto está disponível. Não existem preço e quantidade negativas de estoque.

```
Produto

-nome: String
-preco: float
-qtdEstoque: int

+Produto()
+getNome(): String
+setNome(nome: String): void
+getPreco(): float
+setPreco(preco: float): void
+getQtdEstoque(): int
+setQtdEstoque(qtdEstoque: int): void
+verificaProdutoDisponivel(nome: String): boolean
+totalValorEstoque(): float
```

5. Crie uma classe chamada "Pessoa" que possua atributos para armazenar nome, dataNascimento e profissão. Nenhum ser vivente pode ter mais de 150 anos. Implemente métodos para calcular a idade da pessoa.

```
-nome: String
-dataNascimento: LocalDate
-profissao: String
+getNome(): String
+setNome(nome: String): void
+getDataNascimento(): LocalDate
+setDataNascimento(dia: int, mes: int, ano: int): void
+getAttribute()
+setAttribute(attribute): void
+getProfissao(): String
+setProfissao(profissao: String): void
+getIdade()
```

Para se colocar uma data em um tipo LocalDate:

```
LocalDate I = LocalDate.of(2000, 3, 18); inteiros ano, mês e dia
```

LocalDate hoje = LocalDate.now(); data do dia

Para se buscar os elementos de uma data tipo LocalDate:

```
int dia = I.getDayOfMonth();
int mes = I.getMonthValue();
int ano = I.getYear();
```