



Nota: 10.0



Tempo: Terminou



Sair

Laboratório 7: Interfaces, Encapsulamento

Objetivo

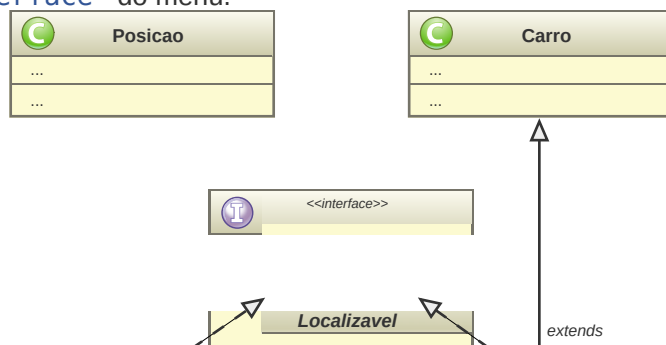
- Implementação de [classes](#) e [interfaces](#) em Java usando os conceitos de [encapsulamento](#).

Descrição

- Neste trabalho, você implementará as classes ao lado necessárias para representar alguns objetos em um Sistema de Informação Geográfica (GIS - *Geographic Information System*). Nestes sistemas, alguns objetos são passíveis de se localizarem (usando GPS, localização WiFi). Tais objetos são instâncias de classes que implementam a interface [Localizavel](#).

Passos Iniciais

- Inicie o Eclipse. Vá em "File" → "New" → "New Java Project". Nome do projeto: "Lab-Encapsulamento".
- Nas questões a seguir, será pedido para criar diversas classes. Para isso, vá em "File" → "New" → "Class". Na caixa que abre, indique o nome da classe e o nome do pacote ("br.edu.ufam.icomp.lab_encapsulamento").
- Para criar uma interface, siga os mesmos passos, mas na opção "Interface" do menu.





Nota: 10.0



Tempo: Terminou



Sair

implements

implements

	Celular
...	
...	

	CarroLuxuoso
...	
...	

Questão 1: Classe Posicao

	Posicao
	latitude: <i>double</i>
	longitude: <i>double</i>
	altitude: <i>double</i>
	Posicao(<i>double latitude</i> , <i>double longitude</i> , <i>double altitude</i>)
	setLatitude(<i>double latitude</i>): void
	getLatitude(): <i>double</i>
	setLongitude(<i>double longitude</i>): void
	getLongitude(): <i>double</i>
	setAltitude(<i>double altitude</i>): void
	getAltitude(): <i>double</i>
	toString(): <i>String</i>

Crie uma classe para representar uma **Posicao** (composta de latitude, longitude e altitude). Os atributos da classe são todos privados (**private**) e não poderão ser acessados diretamente, apenas através dos métodos *getters* e *setters*.

Além dos métodos *getters* e *setters*, sobreponha o método:

- **toString**: retorna uma **String** contendo a descrição da posição conforme o exemplo abaixo (na ordem latitude, longitude, altitude):

Posição: -3.089242, -59.964874, 88.374

Para testar a classe, crie uma nova classe chamada **GISMain**. Nesta classe, crie o método **main**, que será o ponto de partida do seu programa. No método **main**, crie um ou mais objetos da classe **Posicao** e, em seguida, imprima os objetos (eles serão convertidos automaticamente para **String** usando o método



Nota: 10.0



Tempo: Terminou



Sair

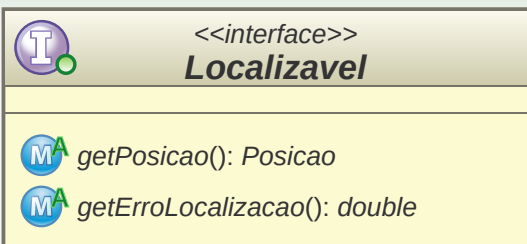
Para essa questão, submeta apenas a classe `Posicao`. Não precisa submeter a classe `GISMdlm`.

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Enviar "Posicao.java"

Solução correta!

Questão 2: Interface Localizavel



Crie uma interface para representar um objeto `Localizavel`. As classes que implementarem esta interface terão que implementar os métodos descritos nela.

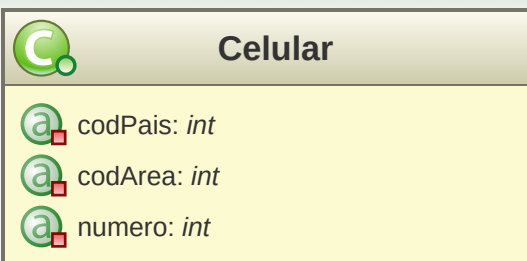
Por ser uma interface, não tem como instanciar um objeto dela. Para usarmos a interface, primeiro precisamos criar uma classe que a implemente (próxima questão).

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Enviar "Localizavel.java"

Solução correta!

Questão 3: Classe Celular





Nota: 10.0



Tempo: Terminou



Sair

setCodPais(int codPais): void
 getCodPais(): int
 setCodArea(int codArea): void
 getCodArea(): int
 setNumero(int numero): void
 getNumero(): int
 getPosicao(): Posicao
 getErroLocalizacao(): double

Crie uma classe para representar um **Celular**. Um celular é **Localizavel** (ou seja, ele implementa a interface **Localizavel**) e possui algumas regras para seus atributos que deverão ser escondidos (**private**) e verificados nos métodos **setters**. Caso algum valor passado para o **setter** não esteja correto, o valor do atributo deverá ser setado para -1:

- codPais: deve estar entre os valores 1 e 1999 (inclusive).
- codArea: deve estar entre os valores 10 e 99 (inclusive).
- numero: deve estar entre os valores 10000000 e 999999999 (inclusive).

Note que os métodos **setCodPais()**, **setCodArea()** e **setNumero()** estão marcados como **final**, indicando que estes métodos não poderão ser sobrepostos por subclasses da classe **Celular**. Além dos métodos **getters** e **setters**, implemente os métodos declarados na interface **Localizavel**:

- **getPosicao**: deverá retornar uma posição geográfica aleatória dentro da cidade de Manaus (isso irá simular um "GPS"), ou seja:
 - latitude: entre -3.160000 e -2.960000.
 - longitude: entre -60.120000 e -59.820000.
 - altitude: entre 15.0 e 100.0.
 - Em java, para gerar um número aleatório entre dois números, importe todas as classes do pacote `java.util` e use:

```
Random r = new Random();  
double latitude = valorMinimo + (valorMaximo - valorMinimo) * r.nextDouble();
```

- **getErroLocalizacao**: como estamos usando um celular, que possui GPS menos preciso, este método irá simplesmente retornar o valor 50.0.

Para testar a classe, modifique a classe **GISMai**n, criada na primeira questão, para criar um ou mais objetos da classe **Celular** e, em seguida, imprima a posição



Nota: 10.0



Tempo: Terminou



Sair

Para essa questão, submeta apenas a classe `Celular`. Não precisa submeter a classe `GISMall`.

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Enviar "Celular.java"

Solução correta!

Questão 4: Classe Carro



	Carro
	placa: String
	Carro(String placa)
	setPlaca(String placa): void
	getPlaca(): String

Crie uma classe para representar um `Carro`. O seu único atributo, `placa` deve ser setado como `protected` e seu `getter/setter` deve ser implementado. Note que:

- um carro simples não é `Localizavel`, uma vez que ele não possui GPS.
- o atributo `placa` da classe é `protected`, ou seja, ele só poderá ser acessado pelas subclasses e pelas classes no mesmo pacote.

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Enviar "Carro.java"

Solução correta!

Questão 5: Classe CarroLuxuoso



	CarroLuxuoso





Nota: 10.0



Tempo: Terminou



Sair

 getPosicao(): Posicao getErroLocalizacao(): double

Crie uma classe para representar um **CarroLuxuoso**, que herda a classe **Carro** e implementa a interface **Localizavel**. Como o **CarroLuxuoso** implementa a interface **Localizavel**, este deverá implementar os seus métodos, que serão iguais aos da classe **Celular** com exceção do método `getErroLocalizacao` que deverá retornar `15.0`.

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Enviar "CarroLuxuoso.java"

Solução correta!

Questão 6: Classe GISMain



Por fim, para exercitar o conceito de *polimorfismo* (agora usando interfaces) modifique a classe **GISMain**. Crie um vetor de objetos de classes que implementam a interface **Localizavel**. Crie e insira no vetor um ou mais objetos das classes **Celular** e **CarroLuxuoso**. Em seguida, faça um **for** para iterar entre todos os elementos e mande imprimir a posição de cada um dos objetos. Como você sobrepôs o método `toString` na classe **Posicao**, você pode mandar imprimir diretamente o resultado do método `getPosicao` diretamente. Exemplo: `System.out.println(vetorLocalizaveis[i].getPosicao());`

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Enviar "GISMain.java"

Solução correta!