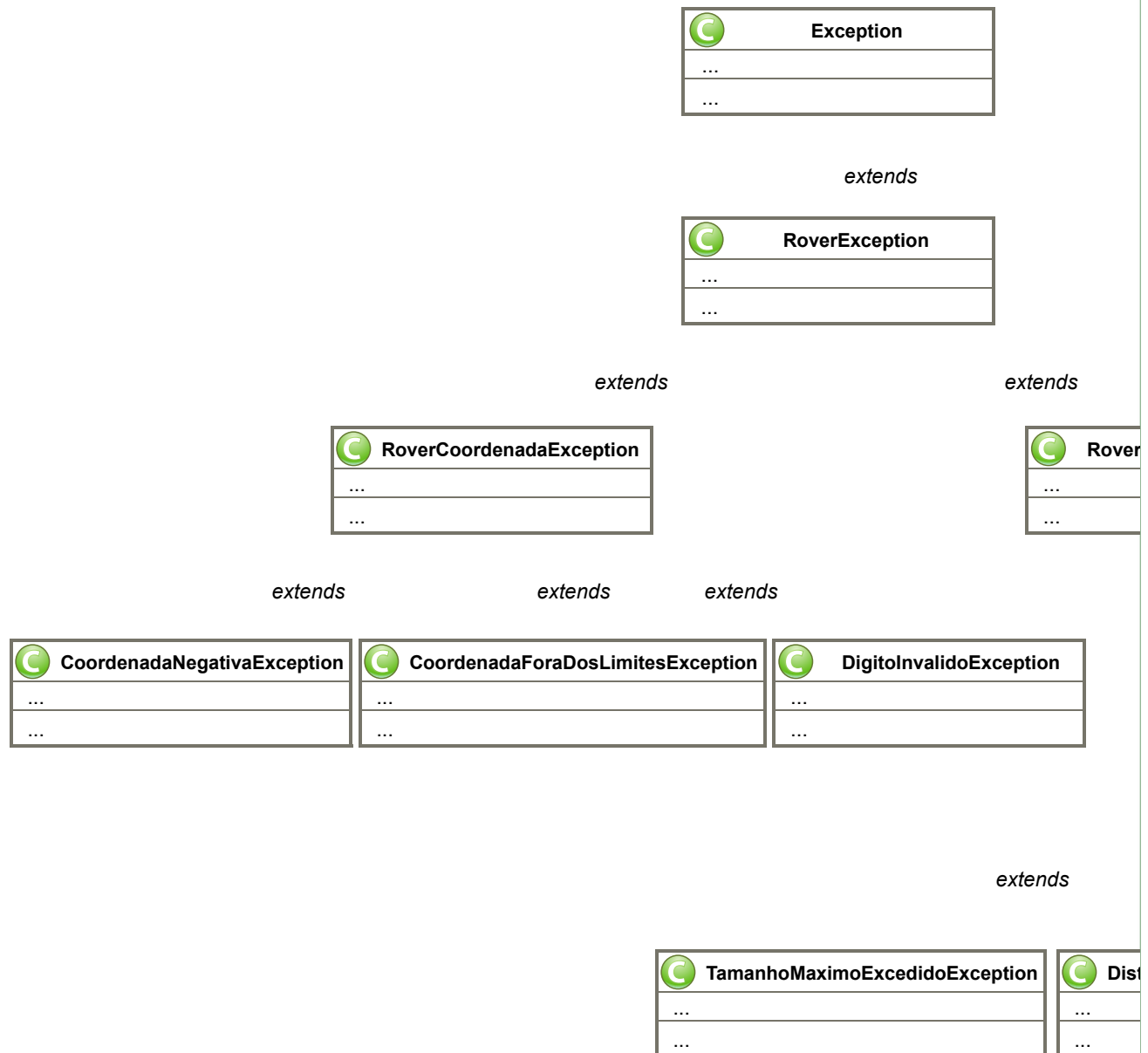


Questão 1 – Classes de Exceções

Crie uma hierarquia de classes de exceções conforme figura abaixo. Estas classes serão usadas pelas classes das questões seguintes.

Atenção: Use o último exemplo dos slides para criar essas classes.



Cada classe de exceção possui uma "mensagem", que será impressa quando ela ocorrer. Seguem as mensagens das classes:

- **RoverException**: "Exceção geral do rover"
- **RoverCoordenadaException**: "Exceção geral de coordenada do rover"
- **RoverCaminhoException**: "Exceção geral de caminho do rover"
- **CoordenadaNegativaException**: "Coordenada com valor negativo"

- `CoordenadaForaDosLimitesException`: "Coordenada com valores fora dos limites"
- `DigitoInvalidoException`: "Digito da coordenada inválido"
- `TamanhoMaximoExcedidoException`: "Quantidade máxima de coordenadas excedida"
- `DistanciaEntrePontosExcedidaException`: "Distância máxima entre duas coordenadas vizinhas excedida"

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

Enviar "RoverException.java"

Solução correta!

Enviar "RoverCoordenadaException.java"

Solução correta!

Enviar "RoverCaminhoException.java"

Solução correta!

Enviar "CoordenadaNegativaException.java"

Solução correta!

Enviar "CoordenadaForaDosLimitesException.java"

Solução correta!

Enviar "DigitoInvalidoException.java"

Solução correta!


Enviar "TamanhoMaximoExcedidoException.java"

Solução correta!

Enviar "DistanciaEntrePontosExcedidaException.java"

Solução correta!

Questão 2 – Classe Coordenada

3.25 / 3.25 

Crie uma classe para representar uma `Coordenada`. O construtor da classe deverá verificar os dados passados e *disparar* as seguintes exceções caso seja necessário (em ordem de prioridade):

- `CoordenadaNegativaException`: esta exceção deverá ser disparada caso alguma das coordenadas seja menor que zero.
- `CoordenadaForaDosLimitesException`: disparada quando alguma das coordenadas não estiver entre os valores 0 e 30000.
- `DigitoInvalidoException`: disparada quando o resto da divisão da soma das coordenadas ($\text{posX} + \text{posY}$) por 10 for diferente do dígito (que deverá estar entre 0 e 9).

O método `distancia(Coordenada coordenada)` deverá retornar a distância entre a coordenada atual e a do argumento.

Já o método `toString` deverá imprimir as coordenadas de acordo com o exemplo abaixo:

```
32, 67
```

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

Enviar "Coordenada.java"








Solução correta!

Coordenada	
	<code>posX: int</code>
	<code>posY: int</code>
	<code>digito: int</code>
	<code>Coordenada(int posX, int posY, int digito)</code>
	<code>getPosX(): int</code>
	<code>getPosY(): int</code>
	<code>distancia(Coordenada coordenada): double</code>
	<code>toString(): String</code>

Questão 3 – Classe Caminho

4.25 / 4.25

Crie uma classe para representar um **Caminho**, que gerenciará um vetor de **Coordenadas**. O método `addCoordenada` poderá disparar as seguintes exceções (em ordem de prioridade), caso algo inesperado ocorra:

Caminho	
	<code>caminho: Coordenada[]</code>
	<code>tamanho: int</code>
	<code>Caminho(int maxTam)</code>
	<code>tamanho(): int</code>
	<code>addCoordenada(Coordenada coordenada): void</code>
	<code>reset(): void</code>
	<code>toString(): String</code>

- **TamanhoMaximoExcedidoException**: disparada ao tentar adicionar mais uma coordenada, mas o tamanho máximo foi atingido.
- **DistanciaEntrePontosExcedidaException**: disparada quando a distância entre o último ponto adicionado e o ponto atual é maior que 15m.

Note que caso alguma das exceções acima ocorra, o método `addCoordenada` não deverá adicionar a coordenada na lista.

O método `reset` deverá zerar a lista de coordenadas (criando uma nova ou setando os elementos para nulo) e também o seu tamanho.

O método `toString` deverá imprimir o caminho completo, conforme exemplo abaixo:

```
Dados do caminho:  
- Quantidade de pontos: 6  
- Pontos:
```

```
-> 32, 30  
-> 35, 40  
-> 38, 30  
-> 30, 36  
-> 40, 36  
-> 33, 31
```

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

Enviar "Caminho.java"

Solução correta!

Questão 4 – Classe RoverMain

0.03 / 0.50 

Por fim, crie uma classe chamada **RoverMain** que irá conter o método `main` e simular a recepção de um caminho vindo da central de comando na Terra para o Rover criando várias **Coordenadas** e adicionando-os a um objeto da classe **Caminho**. Por fim, imprima o caminho completo. Seu código deverá capturar qualquer exceção do rover (**RoverException**) e imprimir a mensagem de erro caso ocorra. Adicionalmente, em caso de qualquer exceção, o método `reset` do caminho deverá ser executado, para evitar que o rover use um caminho inválido (i.e., ele deverá ficar parado caso tenha algum problema com o caminho).

Para imprimir a mensagem de erro da exceção, você pode executar o método `getMessage()` do objeto de exceção passado pelo `catch` ou pode simplesmente mandar imprimir o objeto, uma vez que a classe `Throwable` sobrepõe o método `toString`. Tente criar coordenadas e caminhos inválidos para ver as mensagens de exceção.

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

Enviar "RoverMain.java"

Compilou, mas a resposta está parcialmente errada. Veja mensagens abaixo.

Método 'main(String[])' não encontrado....