UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Aula 6: Programação orientada a objetos [4] (Laboratório 2)

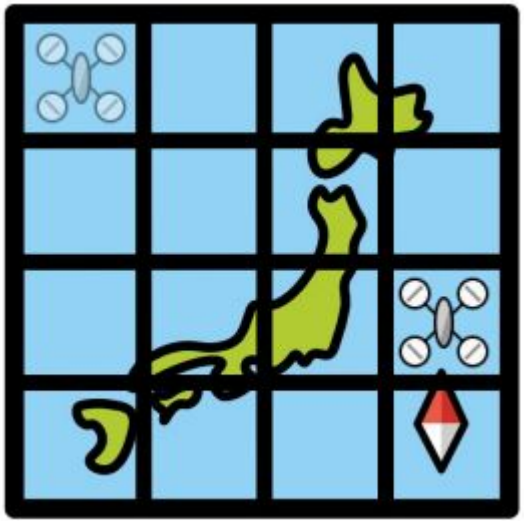
Prof. Dennis Giovani Balreira
(Material adaptado do Prof. Thiago L. T. da Silveira)

INF01120 - Técnicas de Construção de Programas

Laboratório 2

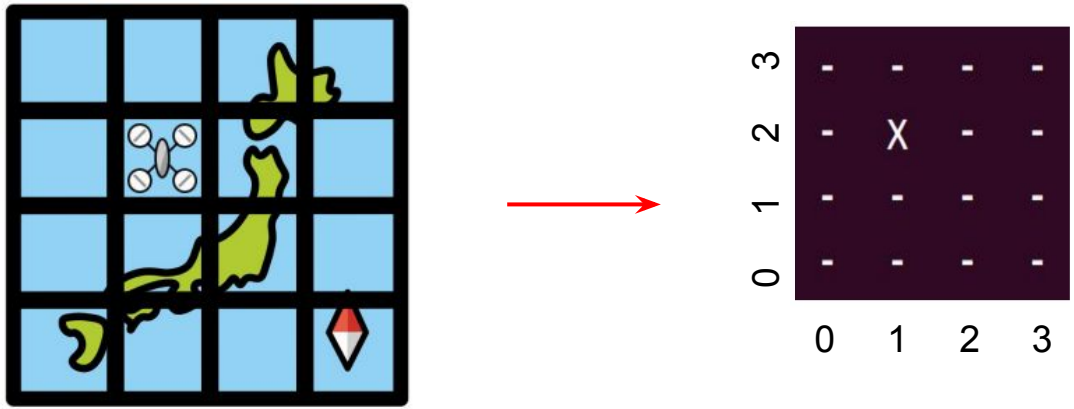
Exercício 1 - Parte 0

- Um simulador de voo de drone precisa manter uma representação dos pontos de partida e de chegada programados. O simulador considera um mapa (grid) com tamanho fixo (4x4). Quer-se medir a distância (de Manhattan) percorrida nesse trajeto com o intuito de estimar a autonomia do drone em termos de consumo de bateria.



Exercício 1 - Parte 1

- Crie uma classe em Java (`PosicaoMapa4x4`) que representa uma posição em um mapa que tem 4 linhas e 4 colunas e outra classe para teste (`AplicacaoTestaPosicoes`) conforme especificação detalhada na sequência

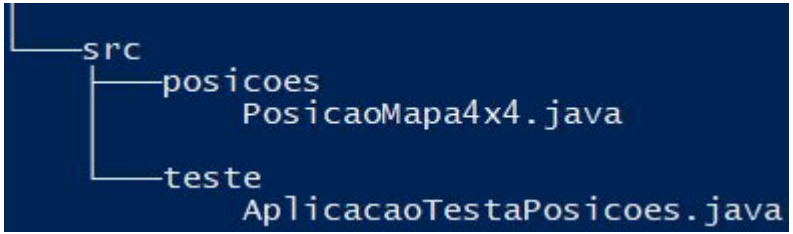


Exercício 1 - Parte 1

- Crie uma pasta `src` em uma pasta do seu computador (ou alternativamente arrume-a no projeto do seu IDE)
  - Dentro dessa pasta crie duas subpastas chamadas `posicoes` e `teste`

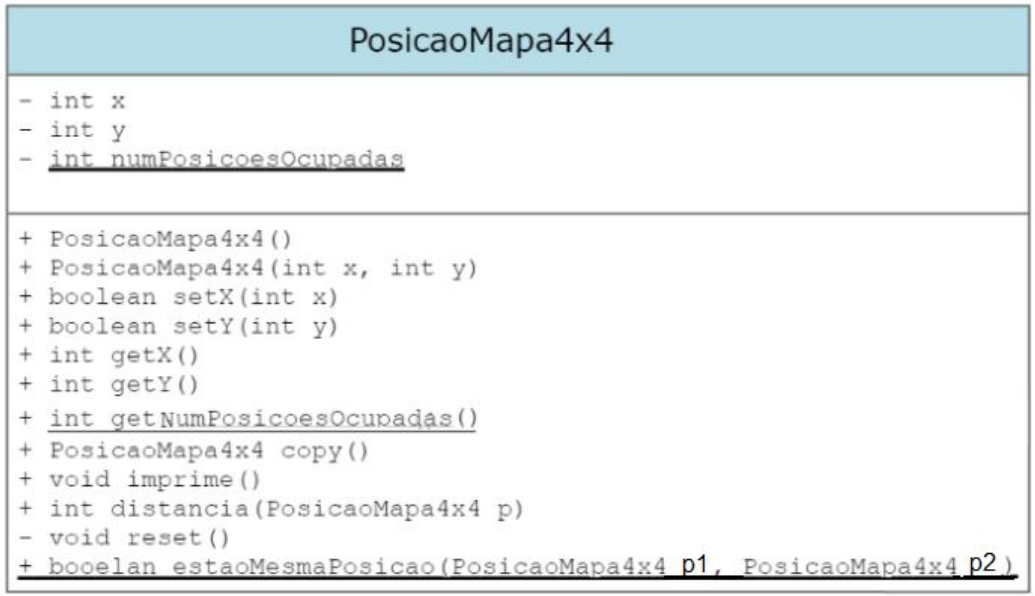
- Crie um arquivo fonte Java chamado `PosicaoMapa4x4.java` dentro da pasta `posicoes` e outro chamado `AplicacaoTestaPosicoes.java` na pasta `teste`

- Lembre-se que os arquivos devem estar organizados hierarquicamente e de incluir "`package posicoes`" e "`package teste`" nos seus devidos lugares

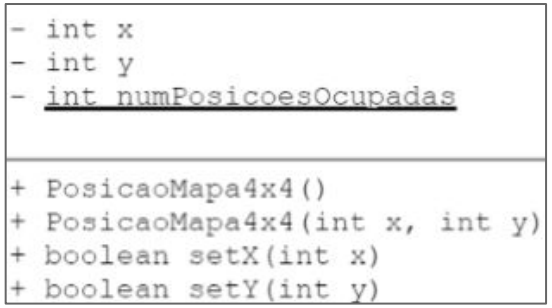


Exercício 1 - Parte 2

- A classe `PosicaoMapa4x4` deve seguir a estrutura abaixo (cuide para seguir os parâmetros e retornos):



Exercício 1 - Parte 2

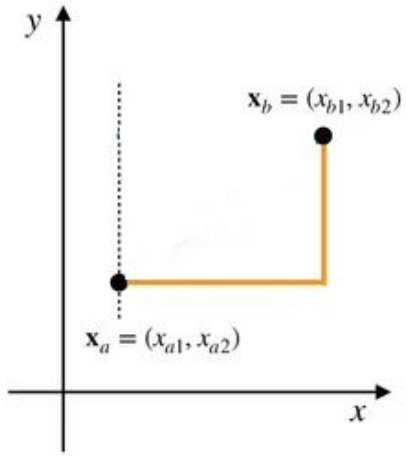


- Na classe `PosicaoMapa4x4`:
  - Definir atributos e métodos conforme notação: (-) privados, (+) públicos e (_) de classe
  - Os atributos `x` e `y` devem armazenar a respectiva posição do drone no grid
  - O atributo de classe `numPosicoesOcupadas` deve conter o número atual de drones existentes
  - O método construtor `PosicaoMapa4x4()` deve invocar o método `reset`
  - O método construtor `PosicaoMapa4x4(x,y)` deve atribuir os valores passados para os atributos x e y
  - O método `setX` só deve atribuir um valor x ao atributo `this.x` se $x \in [0,4)$
  - O método `setY` só deve atribuir um valor y ao atributo `this.y` se $y \in [0,4)$
    - Os métodos `setX` e `setY` devem retornar true apenas se a atribuição for bem sucedida
    - Do contrário, os métodos `setX` e `setY` devem invocar o método `reset` e retornar `false`

Exercício 1 - Parte 2

```
+ int getX()
+ int getY()
+ int getNumPosicoesOcupadas()
+ PosicaoMapa4x4 copy()
+ void imprime()
+ int distancia(PosicaoMapa4x4 p)
```

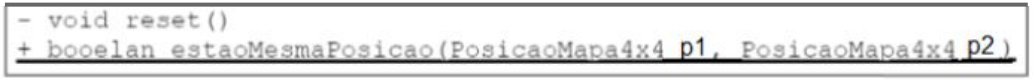
- Na classe `PosicaoMapa4x4`:
  - O método `getX` e `getY` retornam os valores dos atributos `x` e `y`
  - O método `getNumPosicoesOcupadas` retorna o valor do atributo `numPosicoesOcupadas`
  - O método `copy` deve criar e retornar um objeto `PosicaoMapa4x4` que contenha os valores dos atributos do objeto corrente
  - O método `imprime` deve mostrar na tela os valores dos atributos x e y na forma `(x,y)`
  - O método `distancia` recebe um objeto do tipo `PosicaoMapa4x4` e retorna a distância de Manhattan (dM) entre o objeto corrente e o objeto parâmetro



Manhattan distance

$$\|\mathbf{x}_a - \mathbf{x}_b\|_M = |x_{a1} - x_{b1}| + |x_{a2} - x_{b2}|$$

Exercício 1 - Parte 2

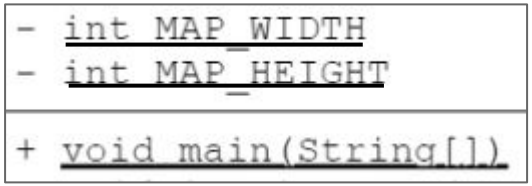


- Na classe `PosicaoMapa4x4`:
  - O método `reset` deve atribuir o valor zero aos atributos x e y
  - O método `estaoMesmaPosicao` deve retornar `true` se dois objetos se encontram na mesma posição do *grid* ou `false` caso contrário

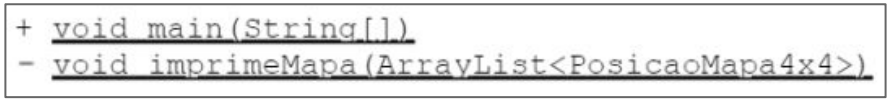
Exercício 1 - Parte 3

- A classe `AplicacaoTestaPosicoes` deve seguir a estrutura abaixo

Exercício 1 - Parte 3



- Na classe `AplicacaoTestaPosicoes`:
  - Definir atributos e métodos conforme notação: (-) privados, (+) públicos e (_) de classe
  - Definir constantes da classe para indicar o número de linhas e colunas do *grid* (MAP_WIDTH e MAP_HEIGHT)
  - O método main deve:
    - Declarar e instanciar um objeto `p1` do tipo `PosicaoMapa4x4`
    - Os valores dos atributos do objeto `x` e `y` devem ser informados pelo usuário
      - Use um objeto `Scanner` para leitura de dados
      - O método `main` deve pedir pares de valores x e y, até que o usuário informe valores válidos para ambos (isto é, quando `p1.setX(x) && p1.setY(y)` é `true`)
    - Assim que os atributos x e y de `p1` assumem valores válidos, deve-se invocar o método `imprime`



Exercício 1 - Parte 3

- Na classe `AplicacaoTestaPosicoes`:
  - Crie um segundo objeto do tipo `PosicaoMapa4x4 p2` e repita todos os passos feitos para `p1` (do slide anterior) para `p2`
  - Invoque o método `distancia` do objeto `p1` passando `p2` como argumento, imprimindo o resultado na tela
  - Crie um terceiro objeto `p3` e copie os valores de `p1` para `p3` utilizando o método `copy` a partir de `p1`
  - Verifique se (p1,p2), (p2,p3) e (p1,p3) estão sobrepostos com `estaoMesmaPosicao`
  - Crie um objeto do tipo `ArrayList*` e coloque `p1`, `p2` e `p3` na lista
  - Crie outro método chamado `imprimeMapa`, que recebe um `ArrayList` de objetos `PosicaoMapa4x4` e imprima na tela o *grid* correspondente ao mapa atual (formato livre, desde que consiga ver as casas nos lugares corretos), indicando com "X" ou "[X]" onde haja pelo menos um drone e "-" ou "[ ]" onde não haja drone
  - Teste o método `imprimeMapa` com a lista recém criada

*https://www.w3schools.com/java/java_arraylist.asp

Exercício 1 - Entrega

- Envie um único arquivo no formato **`ZIP`** contendo a implementação do **Laboratório 2 - Exercício 1** (`PosicaoMapa4x4.java` e `AplicacaoTestaPosicoes.java`)
- O nome do arquivo deve seguir o formato:

  `"l2-<primeiro_nome_do_aluno>-<último_nome_do_aluno>.zip"`

- Atente ao prazo de entrega do trabalho especificado no Moodle!
- Bom trabalho! Qualquer dúvida contate o professor

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Obrigado pela atenção!
Dúvidas?

Prof. Dennis Giovani Balreira
(Material adaptado do Prof. Thiago L. T. da Silveira)

INF01120 - Técnicas de Construção de Programas