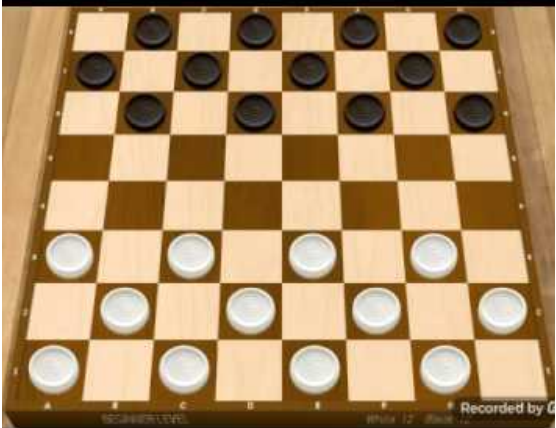


Aula Prática

Jogo da velha

			Implementação Matriz
			/*
			Legenda:
			0 - local vago
			1 - X
			2 - O
			*/
			int jogoDaVelha[][] = { {2,1,1},
			{1,2,1},
			{1,1,2}};

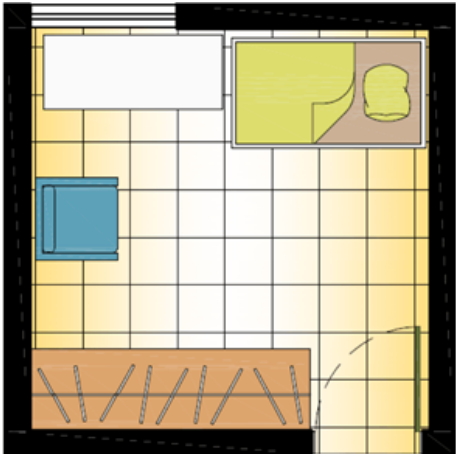
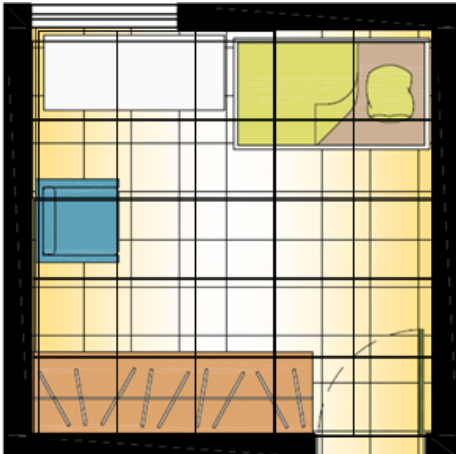
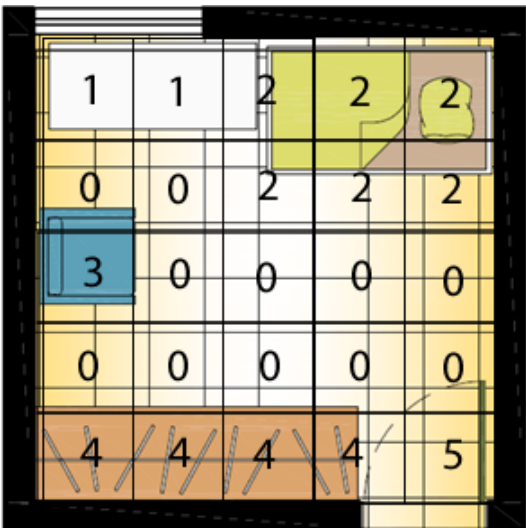
Damas

	Implementação Matriz <pre>public static void main(String[] args) { /* Legenda: 0 - local vago escuro 1 - local vago claro 2 - peça branca 3 - peça preta 4 - rainha branca 5 - rainha preta */ int tabuleiro[][] = { {0,3,0,3,0,3,0,3}, {3,0,3,0,3,0,3,0}, {0,3,0,3,0,3,0,3}, {0,1,0,1,0,1,0,1}, {1,0,1,0,1,0,1,0}, {2,0,2,0,2,0,2,0}, {0,2,0,2,0,2,0,2}, {2,0,2,0,2,0,2,0} }</pre>
---	--

	};
	}

Fazer exemplo de mapa

Quarto

Quarto 	Discretização do Quarto 																																				
Adicionando conhecimento  Onde: 0 - Caminho livre 1 - Escrivania 2 - Cama 3 - Sofá 4 - Guarda-roupa 5 - Robô/Entrada	Como o Software vê a quarto <table border="1"> <thead> <tr> <th></th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> </tr> </thead> <tbody> <tr> <th>0</th> <td>1</td> <td>1</td> <td>2</td> <td>2</td> <td>2</td> </tr> <tr> <th>1</th> <td>0</td> <td>0</td> <td>2</td> <td>2</td> <td>2</td> </tr> <tr> <th>2</th> <td>3</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <th>3</th> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <th>4</th> <td>4</td> <td>4</td> <td>4</td> <td>4</td> <td>5</td> </tr> </tbody> </table>		0	1	2	3	4	0	1	1	2	2	2	1	0	0	2	2	2	2	3	0	0	0	0	3	0	0	0	0	0	4	4	4	4	4	5
	0	1	2	3	4																																
0	1	1	2	2	2																																
1	0	0	2	2	2																																
2	3	0	0	0	0																																
3	0	0	0	0	0																																
4	4	4	4	4	5																																

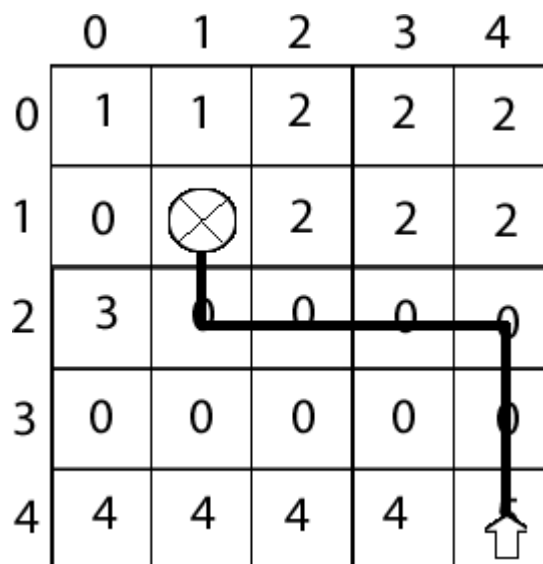
Implementação em Matriz

```
public static void main(String[] args) {  
    /*  
    Legenda:  
    0 - Caminho livre  
    1 - Escrivaninha  
    2 - Cama  
    3 - Sofá  
    4 - Guarda-roupa  
    5 - Robô  
    */  
    int quarto[][] =  
    {  
        {1,1,2,2,2},  
        {0,0,2,2,2},  
        {3,0,0,0,0},  
        {0,0,0,0,0},  
        {4,4,4,4,5}  
    };  
}
```

Qual o problema da representação anterior? A solução está próximo o suficiente da realidade?

Rota no Quarto

Rota no quarto



Implementação Matriz

```
public static void rotaMatriz(){
```

```
int estadoInicial[][] =  
    {  
        {1,1,2,2,2},  
        {0,0,2,2,2},  
        {3,0,0,0,0},  
        {0,0,0,0,0},  
        {4,4,4,4,5}  
    };
```

```
Nodo rota[] = new Nodo[8];  
rota[0] = new Nodo(4,4);  
rota[1] = new Nodo(3,4);  
rota[2] = new Nodo(2,4);  
rota[3] = new Nodo(2,3);  
rota[4] = new Nodo(2,2);  
rota[5] = new Nodo(2,1);  
rota[6] = new Nodo(1,1);
```

```
int estadoFinal[][] =  
    {  
        {1,1,2,2,2},  
        {0,5,2,2,2},  
        {3,0,0,0,0},  
        {0,0,0,0,0},  
        {4,4,4,4,0}  
    };  
}  
public static class Nodo{  
    public int y;  
    public int x;
```

```
public Nodo(int x, int y){  
    this.y = y;  
    this.x = x;  
}  
}
```

Implementação Grafo encadeado

```
public static void rotaGrafoEncadeado(){  
    int estadoInicial[][] =  
    {  
        {1,1,2,2,2},  
        {0,0,2,2,2},  
        {3,0,0,0,0},  
        {0,0,0,0,0},  
        {4,4,4,4,5}  
    };  
  
    // cria os nós da rota  
    NodoEncadeado posicaoInicial = new  
NodoEncadeado(4,4);  
    NodoEncadeado posicao2 = new NodoEncadeado(3,4);  
    NodoEncadeado posicao3 = new NodoEncadeado(2,4);  
    NodoEncadeado posicao4 = new NodoEncadeado(2,3);  
    NodoEncadeado posicao5 = new NodoEncadeado(2,2);  
    NodoEncadeado posicao6 = new NodoEncadeado(2,1);  
    NodoEncadeado posicaoFinal = new  
NodoEncadeado(1,1);  
  
    // encadeia a rota  
    posicaoInicial.proximaPosicao = posicao2;  
    posicao2.proximaPosicao = posicao3;
```

```

posicao3.proximaPosicao = posicao4;
posicao4.proximaPosicao = posicao5;
posicao5.proximaPosicao = posicao6;
posicao6.proximaPosicao = posicaoFinal;

```

```

int estadoFinal[][] =
{
    {1,1,2,2,2},
    {0,5,2,2,2},
    {3,0,0,0,0},
    {0,0,0,0,0},
    {4,4,4,4,0}
};
}

```

```

public static class NodoEncadeado {
    public int y;
    public int x;
    public NodoEncadeado proximaPosicao;

    public NodoEncadeado(int x, int y){
        this.y = y;
        this.x = x;
        proximaPosicao = null;
    }
}

```

Como descrever as ações de um Robô para ele executar tal rota?

Rota



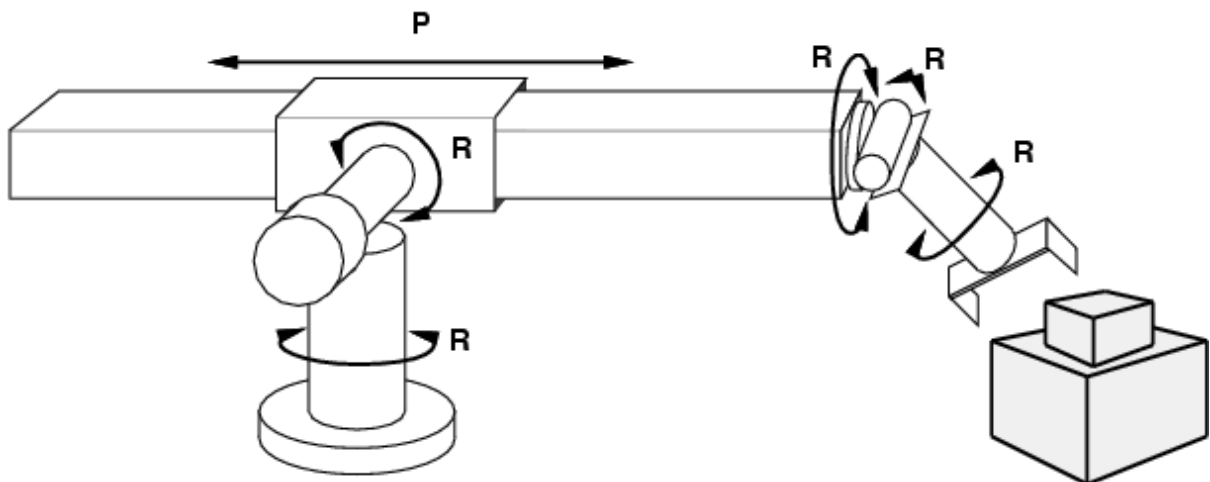
Ações do Robô:

1. Andar para o próximo espaço a frente
2. Continuar andando para o próximo espaço a frente
3. Girar 90° para a esquerda
4. Andar para o próximo espaço a frente
5. Continuar andando para o próximo espaço a frente
6. Continuar andando para o próximo espaço a frente
7. Girar 90° para a Direita
8. Andar para o próximo espaço a frente
9. Parar de executar

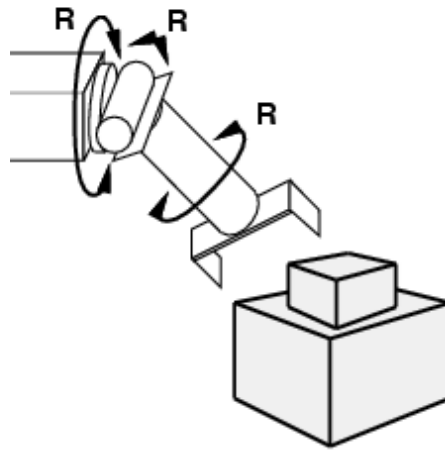
O caso acima o conhecimento do robô sobre suas próprias ações exige uma representação dos seus estados mentais para decidir as ações. Veja uma máquina de estados entre os exercícios para maiores detalhes.

Exemplo de representação do que é uma máquina

O conhecimento de como o robô sobre suas ações pode ser formalizado usando técnicas de representação de conhecimento. Para tanto considere o exemplo na figura abaixo.



Para simplificar vamos selecionar somente a parte do braço mais perto da extremidade.



Ações:

- Rotor Giratório 1: Girar (15°, 30°, 45°, 60°, 90° ou 180°) para a direita ou para a esquerda;
- Rotor Flexão 2: Rodar (Posição de 0° À 180°) para a cima ou para a baixo;
- Rotor Giratório 3: Girar (15°, 30°, 45°, 60°, 90° ou 180°) para a direita ou para a esquerda;

Uma representação mais próxima da formal:

Entidade Rotor Giratório

Propriedades: Posição Atual em °;

Função para ação:

girar(movimentação em °)

Entidade Rotor Flexão

Propriedades: Posição Atual em °; Limite inferior em °; Limite superior em °;

Função para ação:

girar(movimentação em °)

checaSeAcaoEhPossivel(movimentação em °)

Entidade Braço Mecânico

Contém:

- 2 Instâncias de Rotor Giratório inicialmente em posição 0°;
- 1 Instância de Rotor Flexão com limite inferior 0° e limite superior 180°; Posição inicial 0;

Observações: as descrições das ações podem ser extendidas interações entre pessoas ou outros robôs: Comprimetar, negociação, cooperação, etc.

Exemplo modelagem do conhecimento sobre uma Casa

Fotos de casa (o que nos vemos como casa)



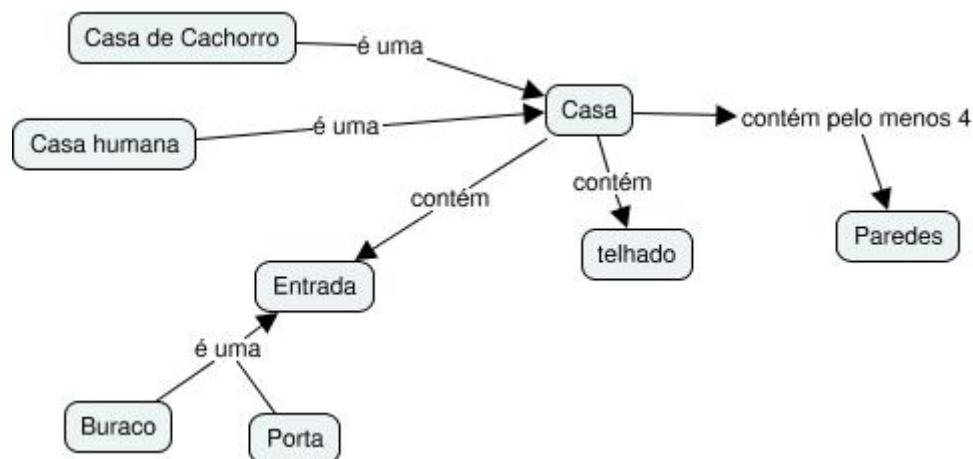
Descrição de casa

Uma casa é uma estrutura que possui telhado, paredes e ao menos uma entrada. Neste local pessoas podem viver e conviver. Algumas casas possuem janelas. Existem tipos de casas diferentes (Cachorro, humana, pássaro, etc). Algumas casas possuem garagem. Algumas casas possuem portas. As casas possuem ao menos um andar e podem ter até dois.

As portas podem ser de madeira, acrílico ou metal e deve ter altura, largura e comprimento como propriedades. Toda porta possui uma maçaneta que é utilizada para abrir ou fechar ela com a mão. Toda porta é presa em uma das laterais.

Etc.

Modelagem Conceitual de uma casa (simplificada)



Implementação lógica

Considere E um objeto que necessita ser identificado na linguagem lógica baixo. A função somente retorna verdadeiro se todas as afirmações retornarem verdadeiro.

```

É_UMA_CASA(E):
    CONTÉM(entrada, E.entrada) e CONTÉM(telhado, E.telhado) e CONTÉM(paredes, E.paredes)

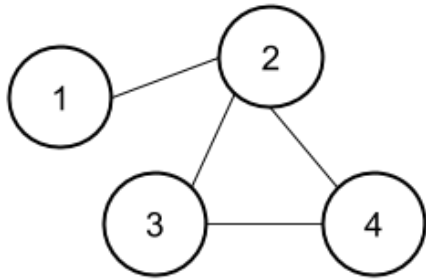
CONTÉM(tipo, S):
    SE tipo == ENTRADA:
        retorna E_UMA_PORTA(S) ou E_UM_BURACO_A_PARTIR_CHAO(S)
    SE tipo == PAREDE:
        retorna ESTA_ACIMA_DO_CHAO(S) e
            É_SOLIDO(S) e ESTA_ABAIXO_DO_TELHADO(S)

    etc.

```

A linguagem acima é determinística.

Representação de um grafo de forma conceitual



Representação formal, onde o G é o grafo, V é o conjunto de vértices, e E é o conjunto de arestas.

Onde $G = (V, E)$:

$$V = \{ 1, 2, 3, 4 \}$$

$$E = \{ 1-2, 2-3, 2-4, 2-1, 3-2, 3-4, 4-2, 4-3 \}$$

De forma opcional um grafo pode ser representado com pesos nas arestas, um para cada aresta.

Onde $G = (V, E, P)$:

$$V = \{ 1, 2, 3, 4 \}$$

$$E = \{ 1-2, 2-3, 2-4, 2-1, 3-2, 3-4, 4-2, 4-3 \}$$

$$P = \{ 1, 1, 1, 3, 2, 1, 4, 3 \}$$

Implementação em Java

Implementação com Matriz

```
public static void grafoComMatriz() {  
    int matriz[][] =  
    {  
        {0,1,0,0},  
        {1,0,1,1},  
        {1,0,0,1},  
        {1,0,1,0}  
    };  
}
```

Implementação Grafo com lista de adjacência

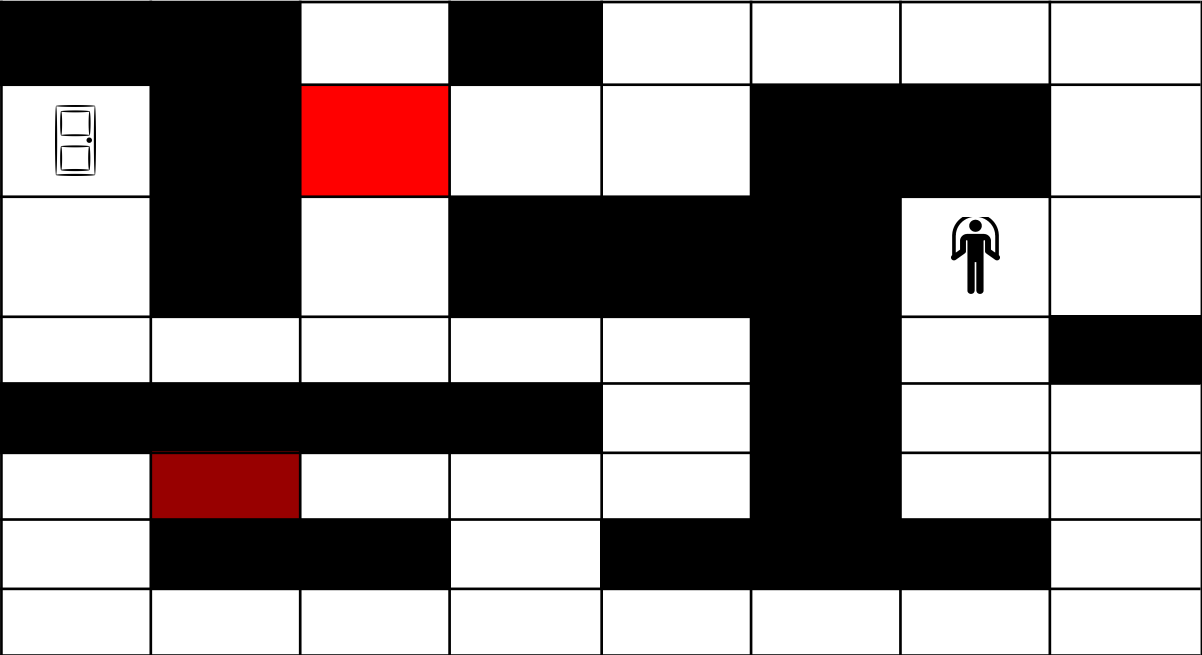
```
public static void grafoComListaDeAdjacencia() {  
    // declara nós  
    Nodo n1 = new Nodo(1);  
    Nodo n2 = new Nodo(2);  
    Nodo n3 = new Nodo(3);  
    Nodo n4 = new Nodo(4);  
    // faz a relação dos nós  
    n1.visinhos.add(n2);  
    n2.visinhos.add(n1);  
    n2.visinhos.add(n2);  
    n2.visinhos.add(n3);  
    n3.visinhos.add(n2);  
    n3.visinhos.add(n4);  
    n4.visinhos.add(n2);  
    n4.visinhos.add(n3);  
}  
  
public static class Nodo {  
    public int id;  
    public ArrayList<Nodo> visinhos;
```

```
public Nodo(int id){
    this.id = id;
    visinhos= new ArrayList<>();
}
```

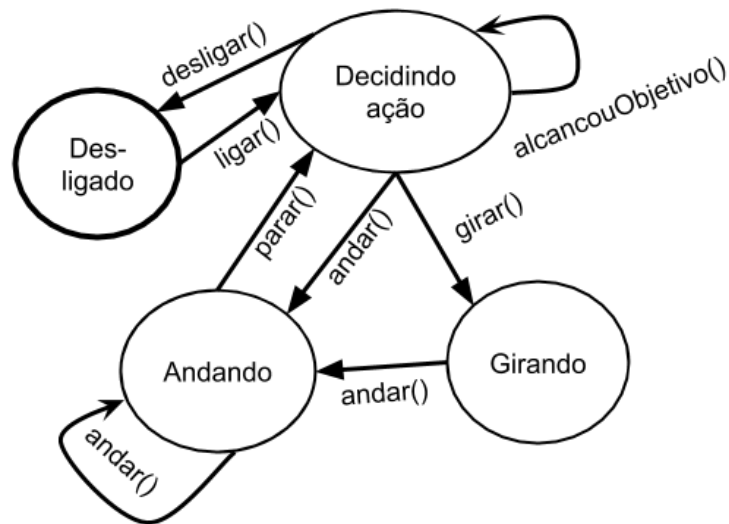
Exercícios

1. Para realizar um programa que executa o jogo de xadrez:
 - Representar um Tabuleiro de xadrez primeiramente em texto e após implemente o tabuleiro em java.
 - Representar cada peça de um tabuleiro de xadrez (incluindo seus movimentos) em texto e após implementar a representação em java.
 - Exemplo de modelagem das ações:
<https://cmapscloud.ihmc.us:443/rid=1TGVJHXSX-X9BT3S-BWJSL4>

2. Considere o labirinto abaixo. Nele uma pessoa está procurando a saída (a porta). Esta pessoa poderá encontrar dois tipos de armadilhas representadas em dois tons de vermelho. Uma vez que a armadilha seja ativada ela não irá funcionar novamente, então sua cor mudaria no mapa. Represente esse labirinto em um programa em java.



3 - Modelar matematicamente e implementar o grafo abaixo que representa uma máquina de estados. Podemos pensar em um robô que está tentando sair do labirinto, para isso representamos os estados do Robô nos Vértices e as ações de troca de estado nas Arestas.



OBS: Em um caso real, mais funções seriam necessárias e elas não necessariamente seriam adequadamente representadas usando uma máquina de estados. Por exemplo, o Robô continuaria andando até encontrar um obstáculo, nesse momento ele executaria a função **parar()** para decidir a ação e identificar se esse obstáculo é o objetivo. Embora, a função que interrompe o andar não esteja representada, mas ela existe e influencia as ações que afetam a máquina de estados.

4. Sobre a questão de rotas/caminho responda:

- 1) De que formas poderíamos implementar um caminho ou rota em um labirinto fixo (labirinto onde todo o mapa é conhecido pelo programa desde seu início e ele não muda)? Qual a técnica que você usaria e porquê? (técnica entre grafo encadeado e matriz).
- 2) De que formas poderíamos implementar um caminho ou rota em um labirinto dinâmico? Isto é, um labirinto que as paredes podem mudar, uma ponte cair, entre outros eventos que podem necessitar modificar uma rota em tempo de execução. Qual a técnica que você usaria e porquê? (técnica entre grafo encadeado e matriz).

5. Em uma sala onde o mapa é completamente desconhecido de que forma você implementaria o mapa sendo descoberto pelo programa? Por quê?

6. Considere as duas salas abaixo. Nessas salas cada uma possui uma pessoa. Represente em java esse ambiente e as pessoas. Cabe ressaltar que as pessoas podem executar as ações: Abrir ou fechar a porta, andar, parar, cumprimentar outra pessoa e sentar nos objetos: sofá, cadeira e chão.

PS.: Se preferir pode ignorar o Dormitório 1 e o corredor de circulação.

