INSTITUTO INFNET

ESCOLA SUPERIOR DE TECNOLOGIA DA INFORMAÇÃO

GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Java EE - TP3

ALUNO: MAGNO VALDETARO DE OLIVEIRA

E-MAIL: mvaldetaro@gmail.com

TURMA: NOITE - LIVE

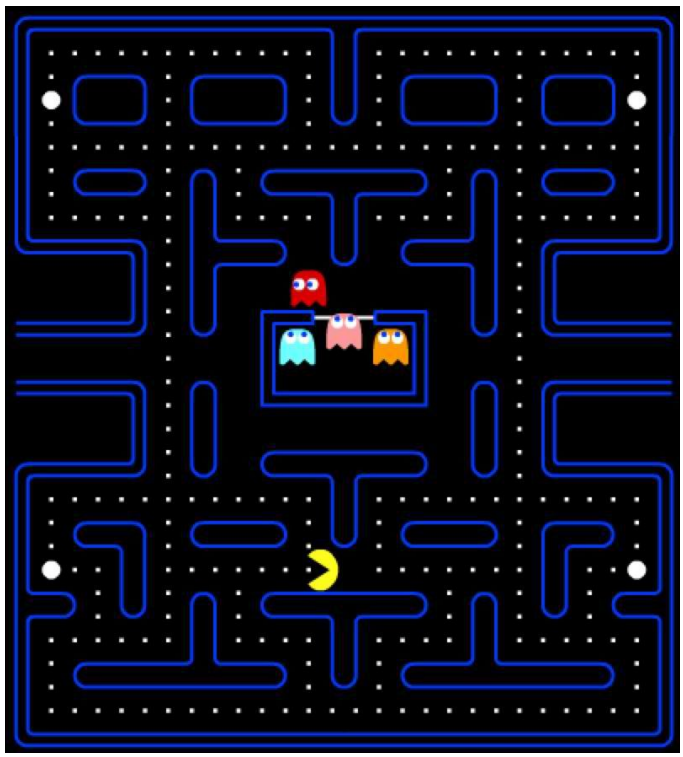
MATRÍCULA: 10403782775

Relatório

Este relatório consiste em uma breve descrição de quatro cenários onde se aplica matrizes, listas, pilhas, filas e quatro estruturas comentadas.

Matrizes

No desenvolvimento de um jogo estilo Pac-Man podemos aplicar uma matriz para mapear o cenário, modelando seus pontos de contato e trilha. Onde na matriz, 0 aponta as casas onde o personagem pode percorrer e 1 as paredes. Ainda podemos aplicar uma segunda matriz que mapeia os pontos e power ups que o pac man consome.

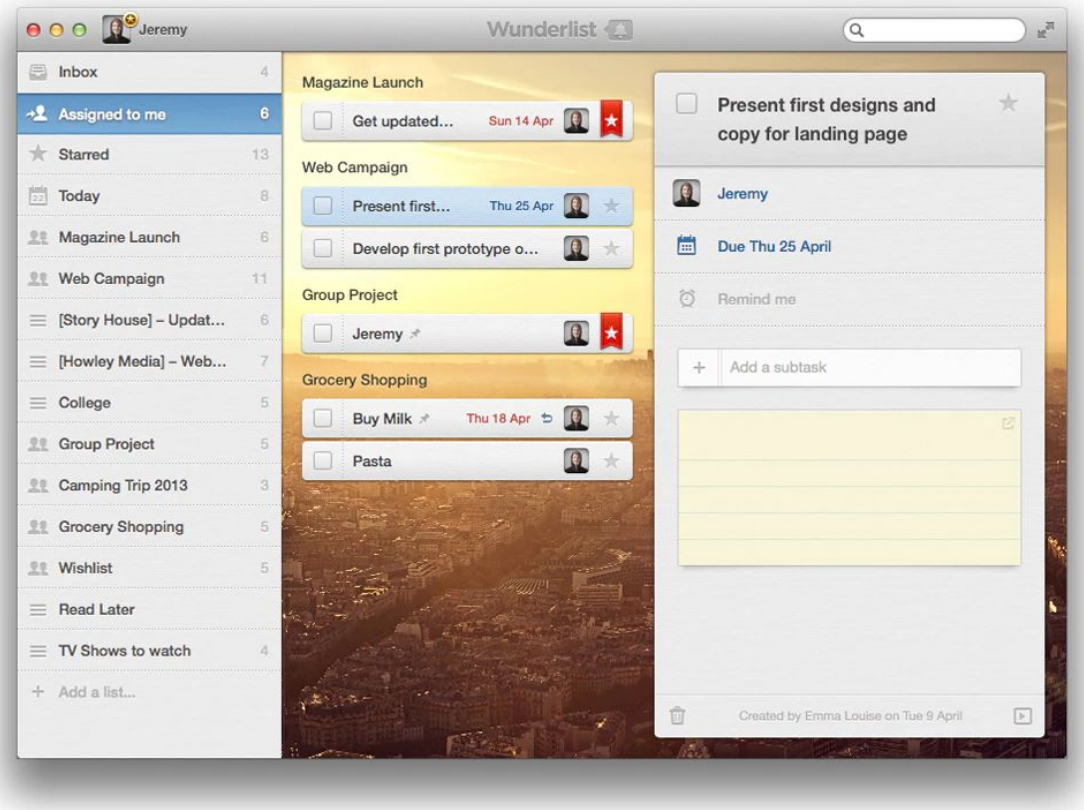


Jogo - Pac-man
Fonte: http://www.downloadtizenapps.com/

```
1111111111111111111111111111
1000000000000110000000000001
1011110111110110111110111101
1011110111110110111110111101
1011110111110110111110111101
1000000000000000000000000001
1011110110111111110110111101
1011110110111111110110111101
1000000110000110000110000001
1111110111110110111110111111
1111110111110110111110111111
1111110110000000000110111111
1111110110111111110110111111
1111110110100000010110111111
0000000000100000010000000000
1111110110111111110110111111
1111110110000000000110111111
1111110110000000000110111111
1111110110111111110110111111
1111110110111111110110111111
1000000000000110000000000001
1011110111110110111110111101
1011110111110110111110111101
1000110000000000000000110001
1000110000111111110000000001
1110110110111111110110110111
1000000110000110000110000001
1011110110000110000110111101
1011111111110110111111111101
1000000000000000000000000001
```

Listas

Uma aplicação de To Do List, é um cenário onde listas podem ser aplicadas como solução para armazenar as tarefas, consumidas de uma base de dados. Aplicando lista encadeada neste sistema, ao concluir uma tarefa o item seguinte ocupa o espaço da concluída que será removida da lista.



Aplicativo de To Do list - Wunderlist
Fonte: https://www.getapp.com

Pilhas

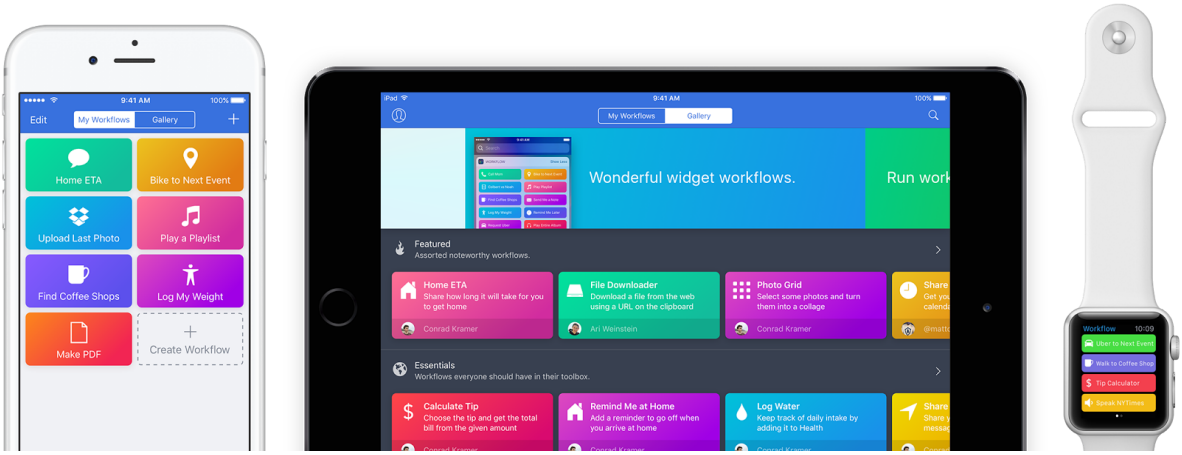
Em um jogo de card games como Magic the Gathering e Hearthstone, a construção do grimório/deck (conjunto de cartas utilizadas ao longo de uma partida) onde o jogador só pode ter acesso a carta do topo do grimório, sendo esta carta a última da lista.

É aplicado a estrutura de pilha, onde a primeira carta da lista é a última a sair (Last In, First Out – LIFO).

Hearthstone: Heroes of Warcraft
Fonte: http://2p.com/

Filas

Um cenário onde filas podem ser aplicações de automatização de tarefas, como o Workflow para iOS. Neste tipo de aplicação é definido um sequência de ações (métodos) que deve obedecer uma estrutura FIFO, onde o primeiro item da lista é o primeiro a ser executado e sair da lista.



Workflow
Fonte: https://workflow.is/

Códigos comentados

Matriz

```
// Importa a classe Scanner
import java.util.Scanner;

// Define a classe matrizTeste
public class matrizTeste {
// Define o método main
    public static void main(String[] args){
// Define um matriz de 3 linhas e 3 colunas
            int[][] matriz = new int[3][3];
// Define um objeto da classe Scanner
            Scanner entrada = new Scanner(System.in);
// Imprime na uma mensagem na tela
            System.out.println("Matriz M[3][3]\n");
// Inicia um loop para posicionar o preenchimento de cada linha
            for(int linha=0 ; linha < 3 ; linha++){
// Inicia um loop para posicionar o preenchimento de cada coluna
                for(int coluna = 0; coluna < 3 ; coluna ++){
// Exibe uma mensagem orientando a ação do usuário
                    System.out.printf("Insira o elemento M[%d][%d]:
",linha+1,coluna+1);
// Recebe a entrada de dados e alimenta a matriz na posição atual
                    matriz[linha][coluna]=entrada.nextInt();
                }
            }
// Exibe mensagem para o usuário
            System.out.println("\nA Matriz ficou: \n");
// Inicia um loop para posicionar a cada linha
            for(int linha=0 ; linha < 3 ; linha++){
// Inicia um loop para posicionar a cada coluna
                for(int coluna = 0; coluna < 3 ; coluna ++){
// Imprime o valor de cada posição da matriz
                    System.out.printf("\t %d \t",matriz[linha][coluna]);
                }
// A cada 3 colunas pula uma linha para exibir a matriz formatada
                System.out.println();
            }

        }
}
```

Lista

```
import javax.swing.*;

// Define a classe No
class No{
    // Define o atributo valor
    int valor;
    // Define o atributo prox como o ponteiro da lista
    No prox;

    // Define um construtor para a classe que recebe um parâmetro do tipo inteiro
    public No(int v){
        // o atributo valor recebe o parâmetro
        valor = v;
        // atribui null ao ponteiro
        prox = null;
    }
}

//Define a classe lista
class Lista{
    //Define que os atributos são do tipo No
    No primeiro,ultimo;
    //guarda a quantidade de itens da lista
    int totalNos;
    // Define um construtor para a classe
    public Lista(){
        primeiro = ultimo = null;
        totalNos = 0;
    }

    // Define um metodo get que retorna o tamanho da lista
    public int getTotalNos(){
        return totalNos;
    }

    // Verifica se a lista está vazia, se o valor retornado por getTotalNos() for
igual a 0 retorna "true", caso contrário retorna false
    public boolean checkIfListaVazia(){
        if (getTotalNos() == 0){
            return true;
       }
       return false;
    }
    // Este método insere um objeto do tipo No no início da lista
    public void inserirNoInicio(No n) {
        if ( checkIfListaVazia() ){
```

```
            //Caso não existam nós inseridos, insere o primeiro nó na lista
            primeiro = ultimo = n;
        }
        else{
            //Caso exista algum nó inserido na lista, pega o primeiro nó e atribui
ao ponteiro do objeto atual (n)
            n.prox = primeiro;
            // define o objeto (n) como primeiro da lista
            primeiro = n;
        }
        //Atualiza o tamanho da lista, incrementando o atributo totalNos
        totalNos++;
    }
    //Este método insere um objeto do tipo No no final da lista
    public void inserirNoFim(No n){
        if ( checkIfListaVazia() ){
            //Caso não existam nós inseridos, insere o primeiro nó na lista
            primeiro = ultimo = n;
        }
        else{
            //Caso exista algum nó inserido na lista, pega o ponteiro do último nó
e objeto atual (n) ao ponteiro
            ultimo.prox = n;
            //Define o objeto (n) como último da lista
            ultimo = n;
       }
       //Atualiza o tamanho da lista, incrementando o atributo totalNos
       totalNos++;
    }

// Define método para exibir a lista
public void exibirLista(){
    //Atribui a uma variável temporária o primeiro objeto da lista
    No temp = primeiro;
    //Atribui um valor vazio a String valores
    String valores = "";
    //Inicia o contador para o loop
    int contador = 1;
    //Verifica se a lsita está vazia, caso retorne false o loop será executado
    if ( checkIfListaVazia() == false ){
        //Enquanto o contador for menor igual a total de itens na lista o loop
continuará executando
        while (contador <= getTotalNos()){
            //Incrementa a String valores com o atributo valor do primeiro objeto
da lista
            valores += Integer.toString(temp.valor)+"-";
            //Aponta para o próximo objeto da lista
            temp = temp.prox;
            //Incrementa o contator
            contador++;
```

```
        }
    }
    JOptionPane.showMessageDialog(null, valores);
 }
}

//Define a classe ListaEncadeadaSimples
public class ListaEncadeadaSimples {
    //Define o metodo main
    public static void main(String[] args) {
    //Define um objeto do tipo Lista
    Lista l = new Lista();
    //Insere um no objeto do tipo No com valor 2 no final da lista
    l.inserirNoFim(new No(2));
    //Insere um no objeto do tipo No com valor 12 no final da lista
    l.inserirNoFim(new No(12));
    //Insere um no objeto do tipo No com valor 22 no final da lista
    l.inserirNoInicio(new No(22));
    //Insere um no objeto do tipo No com valor 32 no final da lista
    l.inserirNoFim(new No(32));
    //Insere um no objeto do tipo No com valor 2 no final da lista
    l.inserirNoFim(new No(2));
    //Exibe a lista
    l.exibirLista();
 }
}
```

Pilha

```
//Define a classe Pilha
public class Pilha {
    //Define o atributo pilha como um array
    public Object[] pilha;
    //Armazena a posição atual da pilha
    public int posicaoPilha;

    //Define o construtor da classe
    public Pilha() {
        // Define que a pilha está vazia
        this.posicaoPilha = -1;
        // Define o tamanho da pilha
        this.pilha = new Object[1000];
    }

    // Este método verifica se a pilha está vazia
    public boolean pilhaVazia() {
        //caso estiver vazia retorna true
        if (this.posicaoPilha == -1) {
            return true;
```

```
        }
        //Se houver algum valor na pilha
        return false;
    }

    // Este método retorna o tamanho da pilha
    public int tamanho() {
        // caso estiver vazia retorna 0
        if (this.pilhaVazia()) {
            return 0;
        }
        // se não estiver vazia retorna o tamanho atual e o incrementa para
sabermos quantos itens tem na pilha
        return this.posicaoPilha + 1;
    }

    //Este método adiciona um item no topo da pilha
    public void empilhar(Object valor) {
        // Verifica se a posição atual é menor que o da pilha
        if (this.posicaoPilha < this.pilha.length - 1) {
            //Insere o valor no topo da pilha
            this.pilha[++posicaoPilha] = valor;
        }
    }

    //Este método remove um item do topo da pilha
    public Object desempilhar() {
        // verifica se a pilha está vazia
        if (pilhaVazia()) {
            // retorna null e nada é feito
            return null;
        }
        // Se houver itens na pilha, retorna o item e o remove da pilha
        return this.pilha[this.posicaoPilha--];
    }

    //Define o método main
    public static void main(String args[]) {
        //Define um objeto do tipo Pilha
        Pilha p = new Pilha();
        // Adiciona a String "Portuguesa" na pilha
        p.empilhar("Portuguesa ");
        // Adiciona a String "Frango com catupiry" na pilha
        p.empilhar("Frango com catupiry");
        // Adiciona a String "Calabresa" na pilha
        p.empilhar("Calabresa");
        // Adiciona a String "Quatro queijos" na pilha
        p.empilhar("Quatro queijos");
        // Adiciona um inteiro "10" na pilha
        p.empilhar(10);
```

```
        //Percorre a lista até que esteja vazia
        while (p.pilhaVazia() == false) {
            //Exibe os valores e os remove da pilha
            System.out.println(p.desempilhar());
        }
    }
}
```

Fila

```
import java.util.LinkedList;
import java.util.List;

//Define a classe Fila aplicando generalização
public class Fila<T> {

    //Define um objeto do tipo LinkedList
    private List<T> objetos = new LinkedList<T>();

    //Define um método para inserir dados na fila
    public void insere(T t) {
        // Adiciona um novo elemento no final da fila
        this.objetos.add(t);
    }
    //Define um método para inserir dados na fila
    public T remove() {
        // remove o item do topo da fila, move os outro elementos da fila e
retorna o elemento removido.
        return this.objetos.remove(0);
    }

    //Define um método para verificar se a lista está vazia
    public boolean vazia() {
        // verifica e retorna se a lista é igual a zero, ou seja vazia
        return this.objetos.size() == 0;
    }

    public static void main(String[] args) {
        // Define um novo objeto do tipo Fila que será do tipo String
        Fila<String> filaDeString = new Fila<String>();
        // Insere a string "Adelaide" no final da lista
        filaDeString.insere("Adelaide");
        // Insere a string "Carolina" no final da lista
        filaDeString.insere("Carolina");

        //Recebe em uma váriavel o item removido
        String carolina = filaDeString.remove();
```

```
        String adelaide = filaDeString.remove();

        // Exibe a os itens processados
        System.out.println(carolina);
        System.out.println(adelaide);
    }

}
```

Referências

Roteiros de Aprendizagem de 5 a 6 e os recursos nestes disponíveis
Disponível em: http://lms.infnet.edu.br/moodle/course/view.php?id=595
Acesso em: 26 de novembro 2016.

Array Multidimensional ou Matriz: Array de arrays
Disponível em:
http://www.javaprogressivo.net/2012/09/array-multidimensional-ou-matriz-array_6673.html
Acesso em: 26 de novembro 2016.

Lista Encadeada Simples
Disponível em:
https://linguagensdeprogramacao.wordpress.com/2011/07/16/lista-encadeada-simples-java/
Acesso em: 26 de novembro 2016.

Pilhas: Fundamentos e implementação da estrutura em Java
Disponível em:
http://www.devmedia.com.br/pilhas-fundamentos-e-implementacao-da-estrutura-em-java/28241
Acesso em: 26 de novembro 2016.

Filas
Disponível em: https://www.caelum.com.br/apostila-java-estrutura-dados/filas/
Acesso em: 26 de novembro 2016.