Relatório do Projeto Final de Linguagens de Programação

Alunos:

Crisley Prestes Linhares

Cristina Souza Araújo

Diógenes Luis Barros de Freitas

1. INTRODUÇÃO

1.1 Python

Foi criada em 1990, por Guido van Rossum do Instituo Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda (CWI), tendo como foco inicial engenheiros e físicos. Sua concepção veio de outra linguagem, o ABC.

Python é uma linguagem de alto nível orientada a objetos, de tipagem dinâmica e forte, interpretada e interativa.

Possui sintaxe clara e concisa, favorecendo a legibilidade, aumentando a produtividade da linguagem. Dispõe de inúmeras estruturas de alto nível e uma vasta coleção de módulos, além de frameworks. É multiparadigma, suportando programação funcional e modular, além da orientação a objetos, neste trabalho, porém, utilizou-se o paradigma procedural.

Sua interpretação é feita através de bytecodes da máquina virtual do Python, tornando o código portável, além de ser um software de código aberto.

É muito utilizado como linguagem de script, permitindo automatizar tarefas e adicionar novas tarefas aos programas, ou integrada com outras linguagens como C.

Utiliza tipagem dinâmica, onde o interpretador define em tempo de execução o tipo da variável, verificando se determinadas operações são válidas, não realizando coerções automáticas se os tipos dos dados forem incompatíveis.

Sob uma visão geral, o Python pode ser estendido através de módulos, ou rotinas escritas em outras linguagens, é portável, pois seus aplicativos podem ser facilmente distribuídos para várias plataformas diferentes daquela em que foi desenvolvido, mesmo que ela não tenha o Python instalado.

1.2 Java

Java foi desenvolvido por James Gosling na Sun Microsystems no início dos anos 1990. Seu nome provém de uma gíria para designar café. O objetivo do projeto inicialmente foi o seu uso para uma aplicação para a qual não parecia haver uma linguagem satisfatória, entretanto era de fato uma sequência de aplicações, sendo a primeira a programação de dispositivos eletrônicos incorporados.

Ainda que o impulso inicial do Java tenha sido para os produtos eletrônicos de consumo, nenhum dos produtos com os quais ele foi usado em seus primeiros anos chegou a ser comercializado. Quando a World Wide Web tornou-se amplamente usada, Java passou a ser considerada uma ferramenta útil para a programação Web.

A linguagem foi projetada tendo em vista os seguintes objetivos:

- Orientação a objetos;
- Portabilidade;
- Recursos de rede;
- Segurança;

Java é uma linguagem complexa. A portabilidade de Java, pelo menos na forma intermediária frequentemente tem sido atribuída ao projeto da linguagem, mas não é. Qualquer linguagem poderia ser traduzida para uma forma intermediária e rodada em uma plataforma que dispusesse de uma máquina virtual para essa forma intermediária. O preço desta espécie de portabilidade é o custo da interpretação, que usualmente é mais ou menos uma ordem de magnitude maior do que a execução do código de máquina.

A verificação da faixa de índices torna a linguagem mais segura. A adição de concorrência aumenta o escopo das aplicações que podem ser escritas na linguagem, o mesmo acontecendo com as bibliotecas de classes para applets, interfaces gráficas, acesso a banco de dados e redes.

Atualmente pode e é utilizado com uma variedade imensa de possibilidades, tanto que qualquer coisa que se pode fazer em outras linguagens, pode ser feito em Java, e na maioria das vezes com vantagens adicionais às outras linguagens. Além de ser a linguagem padrão de aplicativos, appllets e sistemas embutidos.

2. DESCRIÇÃO DA SEGUNDA LINGUAGEM

Java é uma poderosa linguagem de programação. Os programadores experientes às vezes sentem orgulho em criar algum uso distorcido ou exótico da linguagem. Esta prática torna os programas mais difíceis de ler, mais propensos a ter um comportamento estranho, além de serem mais difíceis de testar, depurar e de se adaptar em uma eventual alteração de requisitos.

É uma linguagem portável, com seus programas podendo ser executados em diferentes computadores com diferentes arquiteturas. Para programação em geral, portabilidade significa um objetivo de difícil descrição.

Criação de programas em Java consiste inicialmente da criação de classes, sendo que cada uma delas é composta por um conjunto de métodos que manipulam os campos e fornecem serviços aos usuários. Para se entender melhor a importância das classes para Java, imagine que ela é tão importante para o paradigma de orientação a objetos quanto uma planta arquitetônica é para uma casa.

Um aplicativo Java é um programa de computador que é executado quando se utiliza o comando java para carregar a máquina virtual da linguagem (JVM).

Os tipos de dados em Java são divididos em duas categorias, os tipos primitivos e os tipos por referência, também chamados de tipos não primitivos. Os tipos primitivos são `boolean, char, short, int, long, float, double` e `byte`. Enquanto que os demais tipos declarados em Java são considerados tipos por referência. Uma variável do tipo primitivo pode armazenar exatamente um tipo de valor. Quando outro valor é declarado a essa variável, seu valor original será substituído.

Java é uma linguagem fortemente tipada, uma vez que suas variáveis requerem um tipo. Atribui-se a todas as variáveis de instância do tipo `int, float, long, double, char, byte` e `short` o valor padrão 0, enquanto que as variáveis do tipo `boolean` recebem o valor false por padrão

Os programas utilizam as variáveis de tipo por referência para armazenar as localizações de objetos na memória do computador. Diz-se que estas variáveis referenciam objetos no programa. Os objetos que são referenciados podem conter muitas variáveis de instância e métodos.

As variáveis de instância de tipo por referência são inicializadas por padrão com o valor `null`, que significa uma referência a nada. Uma referência a um objeto é requerida para

invocar os métodos do objeto. Cada classe que é declarada fornece um construtor que pode ser utilizado para inicializar um objeto de uma classe quando o objeto for criado.

De fato Java requer uma chamada de construtor para todo objeto que é criado. A palavra chave `new` chama o construtor da classe para realizar a inicialização. Um construtor deve ter o mesmo nome de sua classe, como um método, um construtor especifica em sua lista de parâmetros os dados que ele requer para realizar esta tarefa.

Como métodos, os construtores também podem aceitar argumentos, porém uma diferença importante entre métodos e construtores é que construtores não podem retornar valores, portanto não podem especificar um tipo de retorno. Se uma classe não incluir um construtor, as variáveis da instância da classe são inicializadas com seus valores-padrão.

Existem ainda três tipos de estruturas de controle em Java, que são as estruturas de sequência, de seleção e de repetição. Cada programa é formado pela combinação destas estruturas ou instruções

Estruturas de sequência estão incorporadas ao Java. A menos que instruído de outra forma, o computador executa as instruções uma após outra dentro de uma ordem em que foram escritas, ou seja, respeitando uma sequência, permitindo ter o número de ações desejado em uma estrutura sequencial.

Java dispõe de três tipos de instruções para seleção, `if` que realiza uma ação se ela for verdadeira, `else` que é executado se uma condição for falsa e `switch` que executa uma entre várias ações distintas de acordo com um valor requerido.

Há três instruções para repetição, que permitem aos programas executarem ações de forma repetida. As instruções `while` e `for` executam uma ação zero ou mais vezes, desde que a condição permaneça verdadeira, enquanto que a instrução `do... while` executa uma ação uma ou mais vezes.

A linguagem disponibiliza operadores de coerção que podem ser utilizados em qualquer tipo. Um operador de coerção é formado colocando parênteses em torno do nome do tipo, como no exemplo abaixo:

```
media = (double) total / contador;
```

Existem operadores lógicos para permitir que os programadores formem condições mais complexas combinando condições simples. Os operadores são && (e condicional), || (ou condicional), & (e lógico booleano), | (ou inclusivo lógico booleano), ^ (ou exclusivo lógico booleano) e ! (não lógico).

Java fornece três tipos de módulos, métodos, classes e pacotes, sendo que os programas Java são escritos através da combinação de métodos e classes escritos a partir de métodos e classes pré-definidos. Em geral, classes relacionadas são agrupadas em pacotes. Os métodos permitem que o programa seja modularizado separando as tarefas em unidades autocontidas. Há vários motivos para se modularizar um programa Java através de métodos, um deles é capacidade de reutilização de código para a construção de outros programas. Dividir um programa em métodos significativos torna o programa mais fácil de depurar e manter.

São fornecidas muitas classes pré-definidas que ficam agrupadas em categoria de classes relacionadas chamadas pacotes. Um ponto de Java é o grande número de classes nos pacotes de API da linguagem. As declarações `import` as classes exigidas para compilar um programa Java. Por exemplo, um programa inclui a declaração:

```
import java.util.Scanner;
```

As declarações introduzem nomes que podem ser utilizados para referenciar entidades em Java. O escopo de uma declaração é a parte do programa que pode referenciar a entidade declarada pelo seu nome. Um exemplo de declaração é demonstrado abaixo:

```
public class TesteEscopo {
	public static void main (String args[]){
		Escopo testeEscopo = new Escopo ();
		testeEscopo.begin();
	}
}
```

Os métodos com mesmo nome podem ser declarados na mesma classe, desde que tenham parâmetros diferentes, isto é denominado de sobrecarga de métodos. Seu uso é comum para declarar métodos com mesmo nome, executando as mesmas tarefas, mas com diferentes parâmetros em sua declaração. Métodos sobrecarregados são diferenciados por sua assinatura, não podendo ser diferenciados pelo seu tipo de retorno.

Os arrays são objetos, sendo considerados tipos por referência, porém seus elementos podem ser do tipo primitivo ou do tipo por referência. Cada objeto do array conhece o seu próprio comprimento e mantém essas informações em um campo `lenght.` Os objetos array ocupam espaço na memória e como todo objeto deve ser criado com a palavra-chave `new`.

Quando criados pela primeira vez os elementos do array são nulos por padrão. Tal expressão retorna uma referência que pode ser armazenada em uma variável array.

Exemplo:

```
int c = new int[12];
String b = new String [100];
String c = new String [27];
```

Os arrays podem ser passados como argumentos de métodos não sendo necessário passar o seu comprimento como argumento adicional.

Exemplo:

```
double calculaTemperatura = new double [24];
modifyArray (calculaTemperatura);
void modifyArray (int [b])
```

Os arrays multidimensionais com duas dimensões costumam ser utilizados para representar tabelas de valores. Java não suporta arrays multidimensionais diretamente, mas permite a especificação de arrays unidimensionais, cujos elementos também são arrays unidimensionais.

Exemplo:

```
int b [][];
b = new int [2][];
b[0] = new int [5];
b[1] = new int [3];
```

Cada objeto pode acessar uma referência a si próprio com a palavra-chave `this`. Quando um método não estático é chamado por um objeto particular, o corpo do método utiliza implicitamente a palavra-chave `this` para referenciar as variáveis de instância do objeto e outros métodos.

O construtor-padrão inicializa as variáveis de instância com os valores iniciais especificados nas suas declarações ou com seus valores-padrão. Os campos `private` de uma classe podem ser manipulados apenas por métodos dessa classe. Fornecer as capacidades de `get` e `set` é essencialmente o mesmo que tornar `public` as variáveis de instância.

Um tipo `enum` (enumerador) pode opcionalmente incluir outros componentes de classes tradicionais como construtores, campos e métodos. São tipos por referência, suas constantes podem ser utilizadas em qualquer lugar em que constantes são utilizadas como nas instruções `switch`.

No Java a hierarquia de classe inicia com a classe `Object` que toda classe em Java estende direta ou indiretamente. Java ao contrário de C++, não suporta herança múltipla, que ocorre quando uma classe é derivada de mais de uma superclasse direta.

Por sua vez suporta herança que permite a reutilização de software na qual uma nova classe é criada, absorvendo membros de uma existente e aprimorada. A linguagem também admite polimorfismo com hierarquias de herança, isto dá a possibilidade de se programar no geral em vez do específico, processando objetos que compartilhem a mesma superclasse em uma hierarquia de classe como se todos fossem objetos da superclasse, a capacidade de objetos de diferentes classes relacionados por herança ou implementação de interface, respondendo diretamente a mesma chamada de método.

O processamento de arquivos é um subconjunto das capacidades de processamento de fluxo do Java que permite a um programa ler e gravar dados na memória, em arquivos e em conexões de rede. Java fornece capacidades substanciais de processamento de fluxos. Ele vê cada arquivo como um fluxo sequencial de bytes, além de conter classes que permitem a programação de ações de entrada e saída de objetos ou variáveis de tipos de dados primitivos.

Java permite a recursão trabalhando com este conceito da mesma forma que em outras linguagens, ou seja, o método recursivo é chamado para resolver o problema, quando na verdade este método só é capaz de solucionar o chamado caso-base, retornando após a execução da pilha um resultado para o problema.

Todo tipo primitivo tem um classe empacotadora de tipo correspondente, que permite manipular valores de tipo primitivos como objetos. Os tipos primitivos não têm métodos, logo seus métodos relacionados encontram-se na classe empacotadora de tipo correspondente. Java não libera memória alocada dinamicamente, em vez disso realiza uma coleta de lixo automática dos objetos que não são mais referenciados no programa.

Classes genéricas fornecem um meio de descrever um conceito de maneira diferente do tipo. Pode-se então instanciar objetos específicos de tipo de classe genérica. Essa capacidade de Java fornece uma excelente oportunidade de reutilização de software. Quando o compilador encontra uma chamada do método, ele primeiro determina os tipos de argumentos e tenta localizar um método com o mesmo nome que especifica parâmetros que correspondem

aos tipos do argumento. Se não houver tal método, o compilador determina se há uma correspondência, inexata, mas aplicável.

A estrutura de coleções do Java fornece componentes reutilizáveis prontos para utilização. As coleções são padronizadas de modo que aplicativos possam compartilhá-las facilmente sem a preocupação com os detalhes de sua implementação. A estrutura de coleções encoraja uma maior reutilização de código.

Java disponibiliza a concorrência de forma simples através do modificador `synchronize` que pode aparecer em métodos e blocos, em qualquer um dos processos, ele faz com que um bloqueio seja anexado, o bloqueio garante o acesso ou a execução mutuamente exclusivo. Em Java é relativamente fácil criar processos concorrentes chamados linhas de execução. O programador especifica os aplicativos que contêm threads de exceção, em que casa thread designa uma parte de um programa que pode executar concorrentemente com outros threads. Esta capacidade é chamada de multithreading, fornecendo capacidades poderosas para o programador de Java não disponível no núcleo de outras linguagens como C e C++ em que Java é baseado. A implementação de multithreading é realizado através da interface `Runnable`, que declara um único método chamado `run`.

A classe `String` fornece vários métodos para realizar operações de expressão regular e localiza o conteúdo do objeto `String` em que ele é chamado na expressão regular.

Exemplo:

```
...
	public static boolean validaPrimeiroNome (String primeiroNome){
		....
	}
```

3. AVALIAÇÃO DO PROJETO

O projeto adotado e implementado neste trabalho é de uma calculadora capaz de realizar as operações básicas da matemática e outras operações relacionadas e existentes em modelos de calculadora científica como a conversão de base.

O objetivo deste projeto em si é projetar uma ferramenta intuitiva e de fácil utilização por parte do usuário que pode através desta realizar operações simplificadas de matemática. Para a parte de implementação o objetivo do projeto foi explorar os recursos existentes em cada uma das linguagens utilizadas bem como nos seus paradigmas.

Para o Python utilizou-se o seu paradigma procedural, e através dele procurou-se compreender os múltiplos mecanismos de programação da linguagem e seu funcionamento dentro da proposta do projeto, utilizando na implementação funções que correspondem as operações descritas em uma calculadora comum, a versão utilizada do Python para o trabalho foi a 2.7.

A segunda linguagem explorada foi o Java já conhecido por sua diversidade de possibilidades de utilização dos seus recursos, tornando-se uma linguagem extremamente poderosa para criação de seus programas. O objetivo de uso desta linguagem foi explorar a orientação a objetos e a parte de interface gráfica de modo a facilitar o uso de uma ferramenta através da intuição do usuário, sem prolongar-se em muitos conceitos relacionados ao paradigma em questão.

Com relação às expressividades ambas as linguagens são bastante poderosas, mostrando uma grande capacidade de escrita de código para o programa, ou seja, várias forma de implementação para o mesmo problema, talvez este poder neste quesito se deva a grande influência de C no projeto e na sintaxe de ambas as linguagens utilizadas.

Relacionado à portabilidade, a execução dos programas só tornaram-se possível com a instalação de ambientes próprios para ambas as linguagens. No caso de Python, o principal problema é a incompatibilidade de versões do software que com suas constantes atualizações faz com que partes do programa não sejam interpretadas em um computador com versão incompatível da linguagem devido à implantação de novos recursos ou mesmo de uma nova sintaxe da linguagem, da mesma maneira como Java que também necessita deste instalado para que a execução do programa seja bem-sucedida, além de IDE's específicas para a programação.

Quanto a eficiência Python mostrou-se mais rápida em sua execução pelo fato de ter-se utilizado o paradigma procedural que dispensou a implementação de uma interface gráfica que no processo de carregamento tira parte da rapidez do programa, fato que pôde ser constatado no Java, pois houve a implementação de uma GUI dentro da proposta de estudo prático do paradigma orientado a objeto.

Para a extensibilidade das linguagens, ambas conseguiram cumprir o que está proposto em seus objetivos de projeto, bem como no objetivo deste projeto de trabalho, não apresentando quaisquer problemas durante o processo de desenvolvimento.

Um dos aspectos mais difíceis de aprendizado em Python foi a necessidade de aprender a linguagem por completo, uma vez que não havíamos ainda programado nesta linguagem, bem como os seus diferentes paradigmas para então se escolher apenas um. Outro

aspecto de dificuldade foi com relação ao ambiente de interpretação do programa, apesar de estar desligado do processo de implementação, foi um processo problemático devido a incompatibilidade de versão do Python nas máquinas utilizadas.

Os aspectos de dificuldade em Java foram mais relacionados com a parte de implementação e compreensão dos conceitos de orientação a objeto e sua prática dentro do código do programa.

Para os ambientes de uso, no caso de Python como já descrito utilizou-se o interpretador padrão da linguagem na versão 2.7 enquanto que em Java utilizou-se a IDE Netbeans 7.0. Os ambientes de desenvolvimento foram bastante proveitosos, com bom aprendizado de uso dos recursos disponíveis em ambas as linguagens. Um detalhe interessante é que no processo de desenvolvimento em Python, deu-se a preferência pela implementação no editor de texto padrão do Ubuntu, o Gedit, interpretando-se o programa diretamente pelo terminal com o interpretador padrão do Python como já citado.

As mensagens de erros, foram mais relacionadas à lógica utilizada na implementação, mas com relação a estas, o ambiente mais intuitivo para este procedimento foi o Netbeans, onde as mensagens apontam para o exato local do código que necessita da correção, enquanto que no Python os erros eram apontados no próprio interpretador com o local do erro no código e sua causa para a correção do programador.

REFERÊNCIAS

[1] DEITEL. Java como Programar – Nova versão atualizada, 6ª ed. Editora Pearson, 2009.

[2] SEBESTA, Robert W. Conceitos de Linguagem de Programação, 5ª ed. Editora Bookman, 2008.

[3] Python (Linguagem de Programação). Disponível em http://pt.wikipedia.org/wiki/pyhon

[4] Java (Linguagem de Programação). Disponível em http://pt.wikipedia.org/wiki/java