Pontifícia Universidade Católica de Minas Gerais
Linguagens de Programação

Java - Everywhere

Aluno(s): Eduardo Pereira Costa
Jonathan Douglas Diego Tavares
Gustavo Gomes de Souza
Stephanie Silva Vieira Gomes
Professor(a): Marco Rodrigo Costa

Maio
2021

Pontifícia Universidade Católica de Minas Gerais
Ciência da Computação
Linguagens de Programação

Java - Everywhere

Relatório de apresentação de conteúdo apresentado a disciplina de Linguagens de Programação, durante o Trabalho Teórico Prático, no Curso de Ciência da Computação da PUC-MG, como requisito final para entrega e avaliação do trabalho.

Alunos: Eduardo Pereira Costa
Jonathan Douglas Diego Tavares
Gustavo Gomes de Souza
Stephanie Silva Vieira Gomes

Professor: Marco Rodrigo Costa

Maio
2021

Conteúdo

1 Introdução

2 Histórico

Entre 1991 e 1994, a Sun Microsystems começou a trabalhar em uma nova linguagem de programação inicialmente denominada Oak como parte do Green Project. Para mostrar a tecnologia Oak e fazer com que seu software pudesse ser utilizado pelo máximo de pessoas possível, o time do Green Project desenvolveu seu próprio browser usando a linguagem Oak, o qual rodava aplicações feitas em Oak.

Em 1995, na preparação para a sua primeira publicação, ou seja, acesso público a nova linguagem, a Sun Microsystems tentou registrar o nome Oak, mas este já havia sido registrado. Logo, em sessões de brainstorm, alguns nomes foram sugeridos, sendo a maioria descartados, até que houvesse apenas um, o Java. Em pouco tempo a nova linguagem ganhou sua logo, muito conhecida, a xícara de café.

Figura 1: Logo Java

O browser criado para rodar as aplicações Oak, agora Java, WebRunner, foi demonstrado pela primeira vez em uma TED conference (conferência para trocar ideias de Tecnologia, Entretenimento e Design) e foi renomeado para HotJava.

A primeira versão do Java é anunciada na conferência SunWorld, onde a Netscape também anunciou um browser com suporte para Java. A Netscape também licenciou o Java e lançou a primeira versão do browser Netscape Navigator com suporte a Java.

A Sun apresenta o slogan "Write Once, Run Anywhere" para descrever os recursos multiplataforma do Java. Em 1996, o JDK 1.0 (Java Development Kit) é lançado e o sistema operacional JavaOS é apresentado. O JavaOS é escrito principalmente em Java e se destina a sistemas incorporados. Ainda neste ano, a Sun apresenta seu primeiro programa focado em desenvolvedores, o Java Developer Connection.

Entre 1997 e 2000, o JDK 1.1 foi lançado junto com recursos principais como classes internas, JavaBeans, RMI Compiler, Reflection, o compilador just-in-time (JIT), internacionalização e suporte à Unicode. Logo após, o JDK 1.2, renomeado para Java 2 Platform, Standard Edition (J2SE 1.2). Novos recursos incluindo AWT, Java 2D, Swing e Coleções de Frameworks.

O J2SE 1.3 é lançado com mais uma leva de novos recursos incluindo o HotSpot Java VM, Java Naming, Interface de Diretório e a Java Platform Debugger Architecture. Mais de 100 milhões de smart cards com suporte a Java, foram vendidos ao final de 2000. O primeiro protótipo de Blu-ray CD Player com suporte para Java foi revelado.

2.1 2001 a 2004

Em 2005, a Sun estima que Java gera mais de US $100 bilhões em receita anualmente. Existem mais de 4,5 milhões de desenvolvedores Java, 2,5 bilhões de dispositivos que utilizam Java e 1 bilhão de smart cards baseados em Java. O programa Java Champions é lançado no JavaOne para reconhecer os líderes da comunidade de desenvolvedores.

2.2 2006 a 2008

Em 2009, a Oracle comprou a Sun e, sendo uma empresa que atuava basicamente na área de Dados, acabou ingressando também no universo do desenvolvimento de software e linguagens de programação. Larry Ellison, co-fundador e diretor executivo da Oracle, juntou-se à liderança da Sun no palco do JavaOne para falar sobre o compromisso da Oracle em investir na tecnologia Java para o benefício dos clientes e da comunidade. Entre 2009 e 2011, a Oracle lançou a Java Magazine, um blog de publicações técnicas sobre Java, escritas por desenvolvedores para desenvolvedores. Como blog oficial da linguagem, fornece detalhes de novas aplicações Java, artigos técnicos de tutoriais, notícias da comunidade e muito mais.

Em 2011, o Java SE 7 é lançado com novos recursos que incluem:

- Project Coin, que tem como objetivo, determinar qual conjunto de pequenas alterações na linguagem deve ser adicionado ao JDK 7.
- Invokedynamic, que permite que um compilador gere códigos com uma especificação mais flexível.
- Estrutura fork / join, que foi projetada para acelerar a execução de tarefas que podem ser divididas em outras subtarefas menores, executando-as em paralelo e combinando seus resultados para obter uma única resposta.
- Uma nova API de sistema de arquivos (NIO.2).

Em 2012, 97% dos computadores empresariais rodavam Java. Em 2014, o Java SE 8 foi lançado com recursos de expressões lambda, anotações de tipos em Java e uma API de Data e Hora. Ao final deste ano, mais de 9 milhões desenvolvedores ao redor do mundo usavam Java.

Em 2015, Java se tornou a plataforma mais utilizada para desenvolvimento no mundo, com mais de 10 milhões de desenvolvedores e 13 bilhões de dispositivos executando Java. O Twitter migrou sua infraestrutura para a JVM e passou a oferecer suporte a mais de 400 milhões de tweets por dia, aumentando o desempenho do aplicativo. No mesmo ano, a Netflix atendia a 2 bilhões de solicitações de conteúdo por dia com arquitetura baseada em Java.

Em 2016, 15 bilhões de dispositivos executavam Java enquanto plataforma principal, a Java Magazine tinha mais de 250.000 assinantes e a Uber começou a utilizar o Java como uma das principais linguagens para o desenvolvimento, com objetivo de tentar garantir alta performance.

Em 2017 o Java SE 9 foi lançado com novos recursos do Projeto Jigsaw (Java Platform Module System), sendo os principais:

- O Jshell, uma ferramenta de loop de leitura e avaliação.
- Compilação antecipada (Ahead-of-time Compilation), uma forma de melhorar o desempenho de programas Java e, em particular, o tempo de inicialização da JVM.
- Jlink, uma ferramenta que gera apenas os módulos da plataforma necessários para um determinado aplicativo.
- Strings compactas, uma forma de minimizar o gasto de memória na representação do que seria uma String na linguagem.

No mesmo ano, Java é ranqueada como a linguagem de programação número um no mundo, sendo a mais utilizada. Além disso, 12 milhões de desenvolvedores ao redor do mundo usavam Java, havendo 38 bilhões de JVMs ativas e 21 bilhões de JVMs conectadas à nuvem. No intuito de acelerar a inovação, a Oracle deu um prazo de seis meses para o lançamento do Java SE 10, que foi lançado em março de 2018.

2.3 2018

2.4 2019

2.5 2020

3 Paradigma

Cada linguagem de programação possui um conjunto de conceitos, princípios e regras e cada conjunto pode ser considerado como um paradigma. Paradigmas de linguagens de programação determinam conjuntos de modelos que possuem uma ou mais características em comum. Essas características são, por exemplo, o estilo ou estrutura de uma linguagem.

A linguagem Java pertence a mais de um paradigma, sendo eles o paradigma imperativo, orientada a objetos (POO), declarativo e funcional. Existem duas categorias principais de paradigmas:

- O imperativo, tendo duas subcategorias: POO e procedural.
- O declarativo, com duas subcategorias: funcional e lógico.

O paradigma imperativo tem como características principais a execução sequencial de instruções e mudança do estado de um programa através de comandos. As instruções são uma definição de passos e determinam como o programa irá operar. As modificações de um estado são armazenadas enquanto variáveis na memória, e então, os comandos são executados sobre os dados armazenados (variáveis). Esse paradigma se preocupa em como alcançar o objetivo proposto e as instruções ditam como o programa opera para realizar esse propósito.

Para definir o que é o paradigma orientado a objetos, primeiro é necessário entender o que é um objeto. Foi dito que a definição de um objeto deriva de "uma entidade que contém dados e comportamentos"(WEISFELD, 2009, p.6). Entende-se um objeto como uma cápsula que tem operações que manipulam a os dados pertencentes a ela. Em Java, essas operações são chamadas de métodos e os dados são chamados de atributos.

É necessário definir também o conceito de classe, que pode ser entendida como um modelo para um conjunto de objetos. Segundo WEISFELD[1], uma classe pode ser considerada como um template ou blueprint para um objeto, e desse modo, os objetos pertencem a esse template. As classes definem qual dado irá compor o objeto, sendo necessário especificar seus tipos e sua visibilidade. Já os métodos que podem ser executadas sobre os dados dos objetos devem possuir uma assinatura, um conjunto de parâmetros e também sua implementação.

O principal objetivo do paradigma orientado a objetos é permitir o desenvolvimento de programas que possam representar objetos, entidades e sistemas reais, que sejam flexíveis, de fácil manutenção e que permitam reuso de "estruturas". Existem 4 princípios fundamentais que são base para esse tipo de programação, sendo eles: abstração, encapsulamento, herança e polimorfismo.

- **Abstração:** Abstração é o processo de identificar as qualidades ou propriedades importantes do fenômeno que está sendo modelado[2]. Um exemplo disso pode ser visto em um carro, em que para dirigir o carro não é necessário saber como os sistemas auxiliares funcionam, mas somente é preciso deter o conhecimento sobre os controles básicos (acelerar, frear, embreagem).
- **Encapsulamento:** Significa uma amarração entre os atributos (campos e métodos) de um objeto com uma ação[3]. A ideia é manter as propriedades e comportamentos de um objeto em um único local para que a manutenção e a extensão do código sejam possíveis e mais fáceis de serem realizadas. O encapsulamento também proporciona um mecanismo para que os atributos não possam ser acessados diretamente por uma fonte externa que foi declarada fora da classe de origem e abstrai detalhes que não precisam ser revelados aos usuários de objetos derivados da classe.

```
IntStream.range(0, 10).filter(i -> i % 2 == 0).
forEach(System.out::println);
```

Listing 1: Uso de Streams

```
//Preenche um Arraylist com n elementos
List<Integer> lista = new ArrayList<Integer>();
for(int i = 0; i < n; i++){
  lista.add(i);
}

//Preenche uma lista de pares atrav s um foreach sobre
//uma Collection que cont m valores inteiros
List<Integer> pares = new ArrayList<Integer>();
for(int val : lista){
  if(val % 2 == 0) pares.add(val);
}

//imprime os elementos presentes na Collection pares
for(int val: pares){
 System.out.println(val);
}
```

Listing 2: Uso de Collections

- **Herança:** É um mecanismo que permite a definição de novos objetos ou classes baseado no reuso de classes previamente existentes. É preciso existir uma classe base(classe pai) que define o comportamento geral para uma entidade. Cada subclasse, que deriva da classe base, podem herdar da mesma alguns comportamentos e atributos que já foram declarados, além de poder alterar/adicionar métodos já existentes ou mesmo criar novos métodos ou novos atributos.

- **Polimorfismo:** Em termos gerais, polimorfismo fornece uma opção ao uso de uma mesma interface sobre entidades de diferentes tipos[4].

O paradigma declarativo especifica o que um programa tem que fazer, sem definir como isso será alcançado. O paradigma funcional é um sub-paradigma do declarativo, nele as funções são similares as presentes na matemática, sendo a saída de cada função depende apenas de seus argumentos, e sendo assim, o estado atual do programa não altera os argumentos e a forma com que os mesmos serão avaliados. Java tem suporte para esse paradigma em apenas alguns casos e por isso não é frequentemente associada ao paradigma funcional e declarativo, ou seja, não é uma linguagem funcional pura. Um exemplo de suporte a esse paradigma pode ser encontrado a partir da versão 8 do Java, como o Streams.

Os streams são definidos no pacote "java.util.stream"e são usados para gerenciar fluxos de objetos em que operações no estilo funcional podem ser feitas. Para fazer esse mesmo exemplo de uma maneira imperativa, é possível utilizar o seguinte trecho de código usando Collections:

Percebe-se que foi necessário uma quantidade maior de linhas de código para alcançar o mesmo objetivo através do paradigma imperativo. O paradigma funcional tem a vantagem de ser menos "verboso", o que pode levar a um entendimento mais fácil do que está acontencendo durante a execução do programa.

Também foi introduzido no Java 8 o pacote "java.util.function", que segundo a documentação da linguagem,

"As interfaces funcionais fornecem tipos para expressões lambda e referências de método. Cada interface funcional possui um único método abstrato, denominado método funcional para aquela interface funcional, ao qual o parâmetro da expressão lambda e os tipos de retorno são correspondidos ou

```
1
2       // contexto de atribuicao
3       Predicate<String> p = String::isEmpty;
4
5       // contexto de invocacao de metodo
6       stream.filter(e -> e.getSize() > 10) ...
7
8       // contexto de cast
9       stream.map((ToIntFunction) e -> e.getSize()) ...
```

Listing 3: Exemplos de programação funcional em Java

adaptados. As interfaces funcionais podem fornecer um tipo em vários contextos, como contexto de atribuição, invocação de método ou contexto de cast".[5]

Exemplo:

4 Características mais marcantes

4.1 Java OO

Java surgiu como uma linguagem que está intimamente conectada ao paradigma orientado a objetos, sendo também inclusa ao paradigma imperativo. Uma vez que Java lida com OO, ela torna-se uma linguagem capaz de explorar uma vasta extensão de mecanismos inerentes a esse paradigma. Todavia, Java não é uma linguagem puramente orientada a objetos, pois implementa também tipos primitivos de dados, os quais dentro da linguagem, não são representados enquanto objetos. Outra razão pela qual Java poderia não ser considerada puramente OO, é o fato de que atualmente, a linguagem implementa estruturas características do paradigma funcional, permitindo obter um desempenho muito superior quando se trata de contextos relacionados ao processamento paralelo de dados e recursividade, além de outros cenários pertinentes.

Pelo fato de Java ser uma linguagem que aplica fortemente os conceitos de OO, é possível utilizar-se da mesma para criar inúmeras frameworks que se aproveitam dos pilares básicos de OO implementados na linguagem. São exemplos de frameworks muito interessantes: "Spring", "Hibernate", "JUnit", "Ant", "Struts(J2EE)".

4.2 Portabilidade - JVM

Java é uma linguagem que foge aos padrões quando o assunto é o processo de compilação. É bastante conhecida pelo uso de uma máquina virtual (JVM), que por sua vez interpreta e executa programas através da geração de *bytecodes*, utilizando-se de um processador virtual e uma estrategia de compilação Just-In-Time que melhora o desempenho diminuindo o tempo necessário para a compilação.

Tal característica peculiar garante à linguagem a capacidade de executar qualquer programa independentemente da plataforma que estiver presente. Uma vez que a execução é feita pela JVM, torna-se possível alcançar uma portabilidade até então não antes vista em outras linguagens anteriores ao Java.

Ser portável garante também a possibilidade de executar software em dispositivos com hardware distintos, e sendo assim, é fácil compreender a razão pela qual o sistema operacional Android, desenvolvido pela Google e que roda Java para Android nativo, está presente em cerca de 71,8% dos dispositivos móveis por todo o mundo. Vale citar também que Java foi a linguagem utilizada para criar o jogo "Minecraft", que foi comprado pela Microsoft junto de sua produtora, a Mojang, pelo valor de aproximadamente $2,5 bilhões de dólares.

4.3 Flexibilidade da linguagem

Java é uma linguagem cuja aplicabilidade é extensa, estando presente desde o cenário empresarial até o entretenimento. São exemplos de possíveis cenários em que Java está sendo utilizado atualmente e fora utilizado no passado:

- Banco Inter, no cenário empresarial, em que Java é utilizado amplamente no backend dos sistemas que dão suporte a plataforma.
- Entretenimento voltado a jogos eletrônicos, tendo nomes como The Sims, Runescape, GTA: San Andreas(Mobile), Flappy Bird, Tibia, e muito mais.
- Educação e aprendizagem de programação, sendo utilizado dentro de escolas e universidades para ensinar conceitos de lógica de programação e paradigma OO. Vale citar nesse contexto, que podem ser trabalhados os usos através de modo console ou interface gráfica.
- Na WEB, em que Java no passado foi amplamente utilizado para criar os famosos "Applets", que eram aplicações nativas em Java, mas que eram executadas pelos navegadores. Atualmente, é utilizado ainda mais na WEB através do JSP(Java Server Pages).

Vale ressaltar que há muitos outros cenários em que a linguagem é aplicável e também utilizada na atualidade.

4.4 Ubiquidade

Algo que é ubíquo, representa a capacidade de estar ao mesmo tempo em toda a parte(onipresente) e difundido por todos os lados(geral, universal). A portabilidade da linguagem permitiu a ela alcançar tamanho feito, uma vez que mais de 2 bilhões de dispositivos no mundo inteiro estão rodando alguma aplicação feita em Java.

Citado este fato, é preciso então mencionar a importância da linguagem no mais novo conceito que está sendo introduzido nos tempos atuais, a "Internet das Coisas"ou IoT(Internet of Things). O IoT, segundo a própria Oracle, descreve a rede de objetos físicos—"coisas"—que são incorporados a sensores, software e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos, e sistemas pela internet.

Esses dispositivos variam de objetos domésticos comuns a ferramentas industriais sofisticadas, havendo mais de 7 bilhões de dispositivos IoT conectados atualmente, os especialistas esperam que esse número aumente para 10 bilhões até 2020 e 22 bilhões até 2025.

5 Linguagens similares ou "confrontantes" (contraditórias)

5.1 C

C surgiu em 1972 como uma linguagem que se encontra dentro do paradigma imperativo, mais especificamente utilizando um paradigma procedural para a estruturação e execução do código fonte. Tal linguagem não permite a utilização do paradigma OO, mas tem como característica relevante um desempenho acima da média, sendo então uma das linguagens mais utilizadas após seu lançamento no desenvolvimento de diversos tipos de software em vários domínios de aplicação.

Java por sua vez, inspirando-se no fato de que C era uma linguagem que explorava fortemente os conceitos básicos do paradigma imperativo, adotou como padrão da linguagem uma estrutura caracterizada como "C-like", ou seja, as estruturas básicas de código como repetições, condicionais, tipos primitivos, etc, são extremamente parecidas com o que se observa na linguagem C.

Ao passo que Java implementa o paradigma OO, mesmo que não completamente, ele abriu as portas em 1995 para uma nova forma de desenvolver software, permitindo a utilização de conceitos de abstração mais profundos, portabilidade através da JVM e também uma curva de aprendizado tênue, comparado

a que se tinha em C, tornando o seu aprendizado muito mais interessante e as aplicações muito mais simples em termos de produção.

Sendo assim, Java e C são linguagens que tem um aspecto contraditório no que se trata do paradigma que ambas focam, apesar de estarem contidas no imperativo. Por sua vez, há também similaridades, pois Java contém estruturas muito parecidas ao que se observa em C e claramente é possível observar que a linguagem desenvolvida pela Sun Microsystems implementa muitas características já vistas na linguagem criada por Dennis Ritchie no Bell Laboratories.

5.2 C++

Acompanhando o lançamento da primeira versão do C++ em 1985 por Bjarne Stroustrup, e posteriormente da segunda versão em 1989, o Java teve múltiplas fontes de inspiração que influenciaram o seu desenvolvimento. O fato de C++ já implementar em sua segunda versão o paradigma OO, herança, classes abstratas e métodos estáticos, mas não permitir criar um código que pudesse ser portado entre diversas plataformas, sem a necessidade de uma recompilação, foi um dos gatilhos para que Java trouxesse as mesmas características do paradigma OO já vistas em C++, mas também assegurasse a possibilidade de executar um mesmo programa, construído através do mesmo código fonte, em plataformas distintas.

Sendo assim, é fácil dizer que C++ e Java tem similaridades, mas também são linguagens concorrentes, pois tem como principal característica tornar mais fácil o que era até então difícil, ou seja, programar em C.

Em termos de "*code debugging*", a liberdade que C++ garante não existe em Java, pois a JVM tem como principal objetivo garantir que os erros possam ser facilmente identificados e corrigidos, ao passo que C++ sequer retorna ao usuário a razão pela qual ocorre um Segmentation Fault, nem mesmo fornecendo um *stack trace*.

Já acerca do desempenho de ambas as linguagens, C++, sendo um derivado de C, está mais próximo do baixo nível, permitindo acessar diretamente recursos que, em Java, são bloqueados pela JVM, a qual atua como uma interface entre o sistema operacional e o programa que está sendo executado, dificultando assim o acesso a recursos de nível mais baixo no intuito de reduzir a quantidade de erros produzidos pelos programadores. Tal característica presente em Java faz com que esta seja mais segura em relação a C++. Dentre os aspectos que influenciam o desempenho comparativo das linguagens, é possível citar o fato de C++ ser totalmente compilado e Java passar por um processo de interpretação dos bytecodes.

Vale ressaltar que em C/C++ não há um recurso automático que trabalhe em função de auxiliar o programador a gerenciar os recursos de memória alocados durante a execução dos programas, sendo o desenvolvedor responsável por tal controle. Java por sua vez traz uma inovação que é o "*Garbage Collector*, um mecanismo automático que desaloca os recursos não mais necessários para a execução do software, dando ao programador a liberdade para se preocupar mais com a resolução do problema proposto do que com a utilização dos recursos computacionais(memória, processador, etc) disponíveis.

5.3 C#

A Microsoft, com o objetivo de criar uma linguagem que tornasse mais fácil trabalhar com estruturas complicadas presentes em C, C++ e Java, propôs uma linguagem que fosse simples, moderna e orientada a objetos. Surgiu então em 2000 o C#, cujas razões para ser utilizado eram ser uma linguagem poderosa e flexível, utilizar poucas palavras reservadas, ser modular e garantidamente popular. Sendo assim, a versão 1.0 de C# foi basicamente similar ao que Java já apresentava no momento em questão, e era claramente inferior a sua concorrente.

Através do lançamento de novas versões, o C# foi se tornando bastante familiar com o Java em diversos aspectos. Desta forma, é impossível dizer que não há similaridade entre as linguagens, pois C# carrega em si o objetivo de melhorar coisas que Java não havia conseguido realizar de forma eficiente o bastante na visão da Microsoft.

Buscando comparar as duas linguagens, há diversos domínios de aplicação em que as duas podem ser concorrentes. Vale citar que tanto Java quanto C# estão intensamente presente no mundo dos jogos eletrônicos. Uma curiosidade interessante é que o Minecraft, feito em Java, foi adquirido enquanto propriedade da Microsoft, ao passo que o Xbox é construído em C e roda o Minecraft utilizando a plataforma XNA, por sua vez construída em sua maior parte usando C#. Há também diversos sistemas WEB que foram construídos utilizando-se JSP, mas que poderiam ser migrados para .NET utilizando C# enquanto linguagem principal, dada a capacidade de converter um software em Java para C# sem muita complexidade adicional.

5.4 Python

Similar

No início da década de 90, surgia uma linguagem que futuramente viria a disputar com a Java no quesito popularidade: Python.

Python é uma linguagem de alto nível, open source, orientada a objetos e, assim como Java, surgiu com o propósito de ser uma linguagem simples e versátil também tendo sua sintaxe derivada, em partes, do C. Apesar de se assemelharem nesses pontos, Python possui características derivadas de Haskell, Modula-3 e Perl e é utilizado, em sua maioria, em nichos distintos dos quais Java se destaca. Java é comumente utilizado em aplicações desktop, sistemas embarcados, mobile e softwares empresariais, enquanto Python ganhou fama em áreas da tecnologia que são tendências, como inteligência artificial e análise de dados.

5.5 Kotlin

5.6 PHP

-JSP

6 Exemplo(s) de programa(s)

7 Considerações finais

8 Conclusão

Referências

[1] WEISFELD, M. The Object-Oriented Thought Process. 2009. Acesso em 23 abr. 2021. Disponível em: https://www.researchgate.net/publication/262350549_The_object-oriented_thought_process_fourth_edition_by_Matt_Weisfeld.

[2] SILVA, F. Programação Orientada a Objetos. Universidade Federal de Uberlândia. Uberlândia, Brasil. 2004. Acesso em 23 abr. 2021. Disponível em: http://www.facom.ufu.br/F̃lavio/poo/files/2004-01/Pootad.pdf.

[3] GABBRIELLI, M. , MARTINE, S. Programming Languages: Principles and Paradigms. Springer-Verlag , Londres. 2010. Acesso em 23 abr. 2021. Disponível em: http://websrv.dthu.edu.vn/attachments/newsevents/content2415/Programming_Languages_Principles_and_Paradigms_thereds1106.pdf.

[4] SINGH, K. , IANCULESCU,A. , TORJE, L. Design Patterns and Best Practices in Java. 2018. Acesso em 23 abr. 2021. Disponível em: https://subscription.packtpub.com/book/application_development/9781786463593/1/ch01lvl1sec13/objectorientedparadigm.

[5] ORACLE. Java.util.fuction. Acesso em 23 abr. 2021. Disponível em: https://docs.oracle.com/javase/8/docs/api/java/util/function/packagesummary.html.

[6] University of Maryland. Department of Computer Science. Lecture 15: QuickSort. Visitado em 06/19. Disponível em https://www.cs.umd.edu/users/meesh/cmsc351/mount/lectures/lec15-quicksort.pdf e https://www.cs.umd.edu/users/meesh/cmsc351/mount/lectures/