SENAI Zerbini

Java

SENAI

O que é Java?

O Java, em computação, pode ser tanto uma linguagem de programação, como também, o programa que interpreta códigos escritos em Java, ou melhor, bytecodes. Assim, se você ler que o Java é uma linguagem de programação, está correto e também, se você ler que o Java é um interpretador, também estará correto!

Java

Compilador

O compiladores são programas que traduzem para linguagem de baixo nível (linguagem de máquina) um programa fonte escrito em linguagem de alto nível . Ao fazer a tradução, o programa fonte se torna um programa objeto, (escrito em linguagem de alto nível compatível com o processador em uso), e depois faz a ligação do programa objeto com as rotinas de execução de programas de sistema operacional, tornado um código executável.

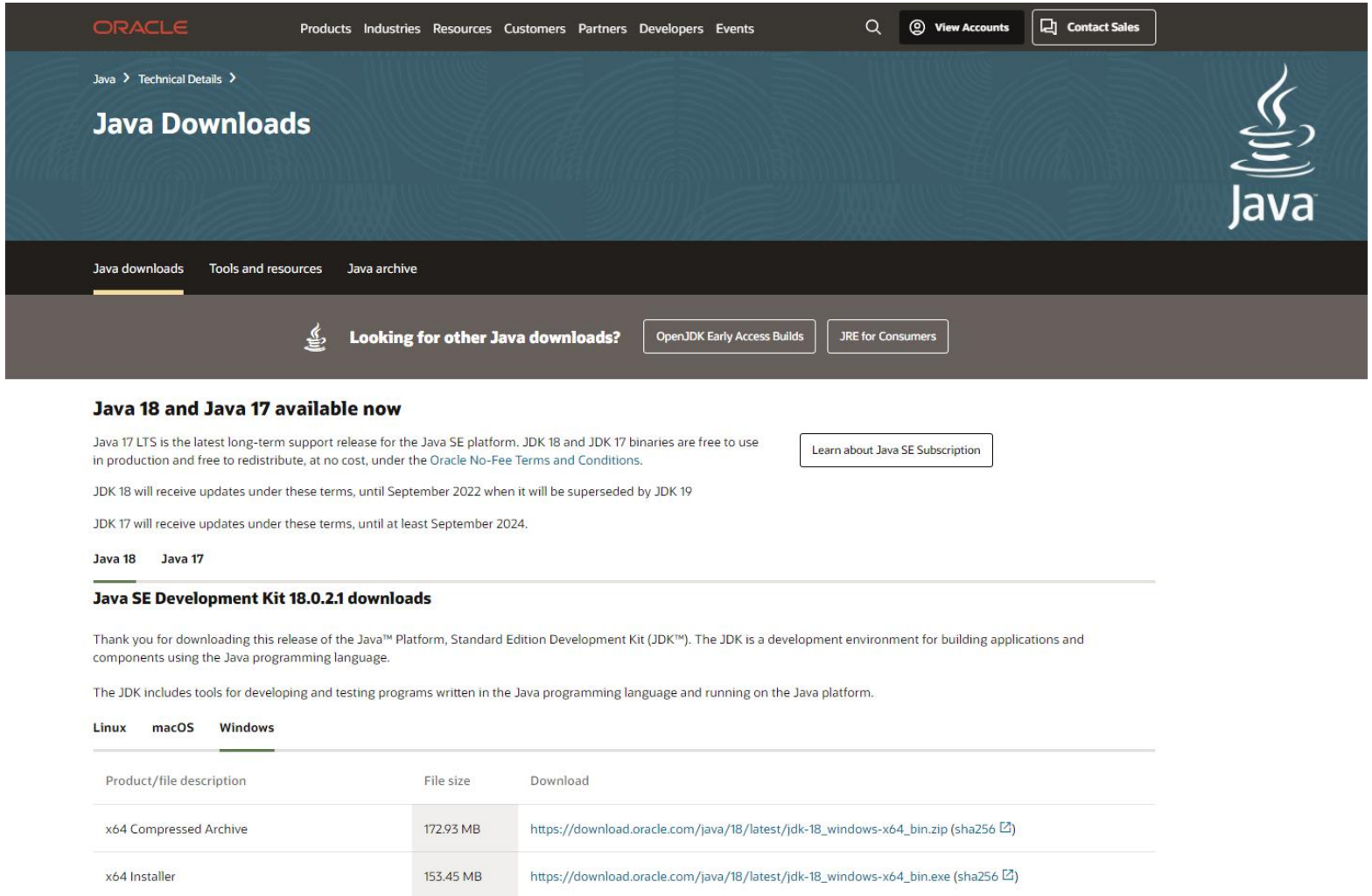
Etapas de criação de um programa em Java

O programa fonte é submetido ao compilador Java , o programa Javac , que cria o arquivo binário do programa (extensão .class), o qual pode ser executado na Máquina Virtual Java (JVM). A extensão .class chama-se arquivo bytecode, em vez de ser chamado de programa executável , pois para ser executável precisa da JVM .Assim ele pode ser executado em qualquer plataforma desde que tenha uma JVM desenvolvida para seu sistema operacional

Etapas de criação de um programa em Java

Seguindo a ideia vamos instalar a IDE para fazermos nosso primeiro Java

https://www.oracle.com/java/technologies/downloads/#jdk18-windows



Programação

Crie o programa fonte abaixo:

```
class AloMundo {
    public static void main(String[] args){
        System.out.println("alo mundo");
    }
}
```

Crie uma pasta “teste” na raiz (c:\). Salve como AloMundo.java

Obs.: O arquivo gravado deve ser igual ao nome da classe.

Programação

- Abrir prompt de comando(cmd.exe)
- Digite o comando cd C:\
  - Isto faz com que o prompt vá para raiz
- Digite :cd teste // entra na pasta teste
- Digite :Javac OlaMundo.java
- Digite :Java OlaMundo

E se estiver tudo certo sua mensagem deve aparecer na tela

Ide Eclipse

O programa Eclipse é uma ferramenta para auxiliar o desenvolvimento de programas. Podemos utiliza-lo para programarmos em várias linguagens, como por exemplo Java, PHP, JavaScript, C, C++ e etc.
Para desenvolver programas em Java, podemos utilizar qualquer editor de texto básico, como por exemplo, o Notepad do Windows. Isso porque, o único programa que realmente é necessário no desenvolvimento é a JVM que é instalada junto com o JDK e utilizada para executar os programas escritos em Java.

Definição de Variável

Variável é uma região de memória reservada para armazenar valores de modo temporário. Carregamos para a memórias as informações que estamos trabalhando em determinado momento, porém, sem remove-las do local de origem.

Em Java, todas as variáveis possuíram um tipo e este será definido ainda na declaração. Podemos alterar o valor armazenado numa variável a qualquer momento, porém, o tipo declarado não poderá ser alterado durante o tempo de execução. Mais precisamente, as variáveis são carregadas para a memória RAM (Random Access Memory) devido à alta velocidade na leitura e gravação dessa mídia. Como já dito, variáveis armazenam valores temporários, no caso, os valores que está sendo manipulando em determinado momento. A memória RAM possui alto desempenho na leitura e gravação, por isso definimos as variáveis ali, no entanto, quando o computador for desligado ou então, houver interrupção no fornecimento de energia, todo valor alocado na memória RAM é perdido de maneira irreversível.

Definição de Variável

```java
public class Variaveis{
    public static void main(String[] args) {

        //declarando uma variável numérica inteira
        int valor;
        //atribuindo o valor 100 para a variável
        valor = 100;

        //imprimindo o valor da variável na saída padrão
        System.out.println(valor);
    }
}
```

No exemplo acima declaramos uma variável do tipo int, de nome valor. Na linha seguinte atribuímos o número inteiro 100 a variável valor. Por fim, utilizamos a função println() para imprimirmos na tela o valor contido na variável valor.

Definição de Variável

```java
public class VarEstudo {

    public static void main(String[] args){

        int num = 0;
        int minhaVariavel = 100;

        System.out.println(num);
        System.out.println(minhaVariavel);

        double dinheiro = 1.55;
        System.out.println(dinheiro);

    }
}
```

Nomenclatura

As variáveis são representadas, geralmente, por uma palavra simples ou composta que a identifica e a torna única em seu escopo. Podemos definir quaisquer nomes a uma variável, porém, existem algumas poucas regras que devemos seguir, do contrário, nossos códigos não funcionaram.

Se você está utilizando nesse momento uma IDE, como por exemplo, Eclipse, Netbeans, IntelliJ IDEA, ao digitar um nome inválido, o editor já acusará um problema na linha e facilmente você perceberá. No entanto, se estiveres estudando com um editor simples, terás de manter a atenção redobrada, até porque, só saberás que o nome é invalido quando o compilador imprimir a mensagem de erro na saída.

As regras que aqui estudaremos também servirão para a definição de nomes de todas as demais estruturas do Java, isto é, existe, basicamente um padrão que serve para a composição de nomes de maneira geral e, as regras que aqui estudaremos servirão futuramente quando formos definir nomes de outras estruturas.

Composição do Nome

Na construção de referências, pode-se utilizar, quaisquer letras, sejam elas maiúsculas ou minúsculas. Variáveis não devem utilizar nomes de classes ou pacotes, ou seja, não devemos declarar uma variável de nome "int", ou então, "String", até porque, esses são nomes de tipos primitivos do Java e não pode ser utilizado. Porém, como veremos, há soluções para esses tipos de situações, isto é, não podemos utilizar a palavra "int", porém, se precedermos a referência com por exemplo um underline, temos o nome torna-se válido, por exemplo: "_int" ou "int_".

CARACTERES NUMÉRICOS

Uma referência pode ser composta por letras e números, desde que este não o nome não inicie com um dos 10 algarismos. Ou seja, podemos utilizar qualquer número na formação do nome da variável, desde que este não esteja à frente do como, como por exemplo: *9num*, ou então, *1var*.

```
public class Aula {
    public static void main(String[] args) {

        int num1 = 1;
        char caractere = 'x';
        bool flag = True;
        String a2b3 = "qualquer texto... ";

    }
}
```

Caracteres Especiais

Caracteres especiais, como por exemplo, os caracteres latinos, como o c-cedilha ç, o tio ~ e etc, não são permitidos na composição de nomes de variáveis. O único caractere tido como especial e que pode ser utilizado é o underline _. A seguir, temos uma lista com os caracteres que não podem ser utilizados na composição de referências e entenderemos a razão nas aulas seguintes, haja vista que os mesmos desempenham funções especificas na linguagem.

- ç
- /
- =
- !
- @
- #
- $
- %
- &
- /
- ( )
- [ ]
- ^
- ~
- ´

Novamente, é importante dizer que caracteres especiais não podem ser utilizados, seja qual for a posição em que estes aparecerem na definição da referência!

Definição de Variável

```
public class Aula {
	public static void main(String[] args) {

		int 4num = 0; //ERRO
		String 3 = "informe seu nome"; //ERRO
		float 2d = 4 //ERRO
	}
}
```

PALAVRAS RESERVADAS (KEYWORD)

A linguagem Java tem definido um conjunto de palavras reservadas que possuem uso especifico na linguagem. Estes nomes não poderão ser utilizados como nomes de variáveis, até porque se assim form, haverá conflito e o interpretador não conseguirá distinguir se a palavra é uma instrução ou então uma variável.

A EXCEÇÃO A REGRA

Como já foi dito, porém, para formalizar a regra temos que o único caractere especial, que pode ser utilizado na composição de nomes, e que também pode estar situando em qualquer posição, é o caractere underline _.

```
public class Aula {
	public static void main(String[] args) {
		int _minhaVar = 55;
		String ____texto = "";

	}
}
```

MACETEZINHO

Na dúvida sobre quais caracteres são permitidos, utilize só e somente só letras sem acentuação, sem espaços em branco, e para garantir, não utilize nomes em Inglês. Por isso, na dúvida quanto a validade da referência, utilize somente letras que terás, com certeza absoluta, um nome de variável válido.

Se você quer decorar o que pode e o que não pode, pense da seguinte maneira:

1. números só se for o segundo caractere
2. letras que não estejam no alfabeto Inglês NÃO permitidos
3. caracteres que não são letras e não são número não podem ser utilizados!
4. há 50 palavras reservadas, porém, todas são em Inglês, logo, evite termos em Inglês num primeiro momento.

Exemplo

```
public class Aula0005 {
    public static void main(String[] args) {
        int inteiro;
        int Inteiro;
        int intEiro;
        int int iro;

        double fracao = 0;

        boolean bool = true;

        String t = "texto";

        System.out.println( inteiro );
//          int 3soma;
        /*
         * tipo nome da variável = null;
         * CaseSensitive
         */
    }
}
```

VARIÁVEIS

Esta aula aprofunda o conhecimento sobre as variáveis e mostra outras maneiras que podemos utilizá-las. Para demonstrar essas funcionalidades é desenvolvido um pequeno sistema que imprime valores inteiros e também, valores do tipo String.

```java
public class Aula0006 {

    public static void main(String[] args) {
        int num1, num2;

        num1 = 100;
        num2 = num1 + 100;

        String s = "";

        System.out.println( "O resultado é: " + num2 );
        System.out.println( "Num1 é igual: " + num1 );

        int i = 1111;
    }
}
```

LENDO DADOS DO TECLADO

Agora, iremos aprender a ler os valores que o usuário digitou na tela. Dessa forma, a partir desse momento, estaremos aptos a receber e armazenar informações, ou seja, agora conseguiremos interagir com o usuário do nosso programa.

```
//import java.util.Scanner;
//import java.net.*;
//import java.net.URL;

Import java.util.Scanner;

public class Aula0007 {

    public static void main(String[] args) {
        System.out.println("Digite um número: ");
        Scanner in = new Scanner( System.in );
        System.out.println(in.nextLine());
    }

}
```

ENTRADA E SAÍDA DE DADOS

Essa aula será dedicada ao estuda da entrada e saída de dados (informações) do nosso programa. Nós temos que a função principal de uma aplicação é processar informações, logo, um programa sempre executa um ciclo de recebimento de informações, processamento e retorno. Assim, a entrada e saída de informações sempre irá seguir no mínimo essas 3 etapas:

1) Entrada de dados A primeira etapa é quando o nosso programa recebe dados. Esses, podem ter sido digitados por um usuário ou então, lido de um arquivo, baixado da internet, recebido de um sensor e etc..... Nós temos diversas formas para receber informações, porém, mesmo que as informações estejam sendo digitadas ou então, lidas de um site da web, nós estamos na primeira fase do ciclo, isto é, estamos recebendo informações.

2) Processamento A segunda etapa é o processamento. Não há como prever em que momento uma informação será processada, porém, raramente alguém irá armazenar uma informação que não tenha utilidade e assim, nunca seja processada. Nós só armazenamos aquilo que é necessário ou então, aquilo que achamos que um dia poderemos precisar. Logo, o processamento é uma etapa que não é possível prever quando que será executada.

3) Saída dos dados. Por fim, nós temos a saída de dados. Existem centenas de formas em que podemos retornar informações para o nosso usuário, assim, as informações podem ser exibidas em um monitor, em um display, em um led, em uma impressora e etc. A forma como a informação será extraída do nosso programa não é o mais importante agora, o que realmente importa é saber que toda informação que entra em nosso programa, em algum momento será processada e o ciclo termina, quando está é retornada de alguma forma para o (s) usuário (s).

ENTRADA E SAÍDA DE DADOS

```
import java.util.Scanner;

public class Aula0008 {
    public static void main(String[] args) {
/*
 * 1) entrada de dados
 * 2) processamento de dados
 * 3) saída de dados
*/
        System.out.println("Informe a idade do seu cachorro: ");
        Scanner in = new Scanner( System.in );

        int idadeCachorro = in.nextInt();
        idadeCachorro = idadeCachorro * 7;

        System.out.println("O seu cachorro tem "+idadeCachorro+"
anos.");
    }
}
```

CALCULADORA

Nessa aula desenvolveremos uma primeira versão de uma simples calculadora. O nosso objetivo não é sofisticação, mas sim, treinar as operações matemáticas da linguagem Java. Em aulas futuras, iremos desenvolver outras versões dessa calculadora, onde a mesma ira efetuar outras operações e também, terá a sua logística melhorada.

```java
import java.util.Scanner;

public class Aula0009 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        double num1, num2, soma, subtracao, multi, divi;

        System.out.println("Insira um número: ");
        num1 = in.nextInt();

        System.out.println("Insira outro número: ");
        num2 = in.nextInt();
        num2 = num1 + num2;
        System.out.println("O resultado da soma é: ");

        subtracao = num1 - num2;
        System.out.println("O resultado da subtração é: ");

        multi = num1 * num2;
        System.out.println("O resultado da multiplicação é: ");

        multi = num1 / num2;
        System.out.println("O resultado da divisão é: ");


    }
}
```