Vitor Amadeu Souza

Antologia de projetos em

Java

No ambiente IntelliJ IDEA

© 2024 by Cerne Tecnologia e Treinamento Ltda.

© 2024 by Vitor Amadeu Souza

Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Janeiro de 2024

Direitos reservados por:
Cerne Tecnologia e Treinamento Ltda
Produção: Cerne Tecnologia e Treinamento
E-mail da Empresa: cerne@cerne-tec.com.br
Home Page: www.cerne-tec.com.br.com.br
Atendimento ao Consumidor: sac@cerne-tec.com.br
Contato com o Autor: vitor@cerne-tec.com.br



FEITO NO BRASIL

“Recompensou-me o Senhor conforme a minha justiça,

retribuiu-me conforme a pureza das minhas mãos.”

Sl 18:20

Kits Didáticos e Gravadores da Cerne Tecnologia

A Cerne tecnologia têm uma linha completa de aprendizado para os microcontroladores da família PIC, 8051, Holtek, dsPIC, ARM, STM etc. Veja os detalhes de um kit na figura abaixo:



Kit Cerne PIC12F

- Microcontrolador PIC12F675
- Botões
- Leds
- Gravação ICSP
- E muito mais!

Uma linha completa de componentes para o desenvolvimento de seus projetos eletrônicos como displays, PICs, botões, leds, cristais dentre outros. Visite a nossa página na Internet, no endereço www.cerne-tec.com.br e conheça melhor nossos serviços e produtos.

Cerne
Conhecimento para o Desenvolvimento

Sumário

Introdução.....11

Capítulo I – Programação em Java.....12

1. Introdução.....12
2. Ambiente IntelliJ IDEA.....14
3. Apresentando mensagens.....18
4. Comentários.....20
5. Capturando dados no Java.....23
6. Calculadora I.....24
7. Calculadora II.....25
8. Tipos de dados.....27
9. Operadores da linguagem.....41
10. Estrutura de repetição for.....44
11. Estrutura de repetição while.....49
12. Estrutura de repetição do-while.....53
13. Declaração de controle if.....57
14. Declaração de controle switch.....63
15. Classe String.....64
16. Palavras reservadas.....65
17. Vetor ou array.....66
18. Matriz.....67
19. Operador ternário.....68
20. Procedures.....70
21. Funções.....71
22. Foreach.....71

Capítulo II – POO – Programação Orientada a Objetos.....73

1. Classes e objetos.....73
2. Criação de objetos.....79
3. Atributos e métodos de classe.....81
4. Encapsulamento.....83
5. Métodos.....88

6. Passagem por valor....90
7. Palavra reservada this....91
8. Sobrecarga ou overloading....93
9. Herança....96
10. Classe abstrata....101
11. Classe final....102
12. Exemplo Classe Conta....103
13. Exemplo Classe Professor....105
14. Exemplo Classe Círculo....108
15. Exemplo de Herança Classe Pessoa....110
16. Exemplo de Herança Classe Funcionário....112
17. Exemplo de Herança Classe Animal....114
18. Interface....116
19. Polimorfismo dinâmico....117
Capítulo III – Interface gráfica....120
1. Lendo um nome....120
2. Concatenando strings....121
3. Calculadora....122
4. Caracteres maiúsculos....124
5. Adicionando evento addActionListener....126
6. Saudação com caixa de texto....128
7. Carregando imagens....129
8. Carregando imagens 2....134
9. Classe gráfica....137
10. Combo Box....140
11. Option Box....142
12. List Box....145
13. Alterando cor do form....148
14. Calculadora completa....151
Capítulo IV – Impressão pdf....155
1. Instalando a dependência....155
2. Impressão em pdf....160
Capítulo V – Jsoup....167

1. Introdução....167
2. Instalando a dependência....169
3. Consulta a página da Cerne Tec....175
4. Consulta a página da Oracle....176
5. Consulta a página em php 1....178
6. Consulta a página em php 2....184
7. Consulta a página em php 3....193
Capítulo VI – HttpRequest....202
1. REST....202
2. JSON....204
3. XML....205
4. HttpRequest....206
5. Exemplo REST 1....208
6. Exemplo REST 2....210
Capítulo VII – Spark....213
1. Introdução....213
2. REST....214
3. JSON....216
4. XML....217
5. Instalando a dependência....218
6. Exemplo 1: Respondendo requisições GET por HTTP....223
7. Exemplo 2: Alterando a porta de comunicação....225
8. Exemplo 3: Adicionando novas rotas....226
9. Exemplo 4: Contador de acessos....230
10. Exemplo 5: Retornando dados no formato JSON....231
11. Exemplo 6: Retornando dados no formato XML....233
12. Exemplo 7: Controlador Rest....236
13. Exemplo 8: Calculadora com método GET....239
14. Exemplo 9: Calculadora com método POST....245
Capítulo VIII – API....253
1. Introdução....253
2. REST....254
3. JSON....255

4. Código fonte....256
Capítulo IX – Plotando gráficos....262
1. Instalando a dependência....262
2. Função de 1°....267
3. Função de 2°....274
4. Função de 3°....279
5. Função seno....283
6. Função cosseno....285
7. Função tangente....287
8. Função exponencial....289
9. Função Gaussiana....291
9. Função sigmoide....294
Capítulo X – Lombok....299
1. Introdução....299
2. Instalando a dependência....300
3. Exemplo....306
Capítulo XI – Spring Boot....312
1. Introdução....312
2. REST....314
3. JSON....315
4. XML....316
5. Spring Initializr....317
6. Exemplo 1: Controlador Rest....321
7. Exemplo 2: Controlador Rest....325
8. Exemplo 3: Retornando dados no formato JSON....327
9. Exemplo 4: Retornando dados no formato XML....329
10. Exemplo 5: Controlador Rest....331
11. Exemplo 6: Controlador Rest....334
12. Exemplo 7: Controlador Rest....338
Capítulo XII – Teste unitário com JUnit....347
1. Teste unitário....347
2. Teste de integração....347
3. Teste de ponta a ponta (E2E)....348

4. Teste manual....348
5. Teste automatizado....349
6. Teste de caixa branca....350
7. Teste de caixa preta....350
8. JUnit....351
9. Teste em uma classe Calculadora....352
10. assertEquals....368
11. assertNotEquals....370
12. assertArrayEquals....372
13. assertTrue....374
14. assertFalse....375
15. assertNotNull....377
16. assertNull....379
17. assertSame....381
18. assertNotSame....381
19. fail....384
20. assertAll....389
Capítulo XIII – Acesso a arquivo externo....387
1. Exemplo de escrita....387
2. Exemplo de leitura....389
Capítulo XIV – Projetos no Java....391
1. Cálculo de circunferência....391
2. Cálculo de raízes de equação do 2º....392
3. Determinante de matriz 3x3....393
4. Triângulo retângulo....395
5. Conversão para radianos....397
6. Cálculo de cosseno e seno....398
7. Dado eletrônico....399
8. Conversor de base....400
9. Média de uma turma....401
10. Juros compostos....403
11. Sequência de Fibonacci....404
12. Tipos de Triângulo....405

13. Dados do censo 2022....407
14. Cadastro de pessoas....403
15. Números primos....414
16. Impressão de sequência de números primos....415
17.Soma de Gauss....416
18.Número de Fermat....418
Capítulo XV – Regex....420
1. Expressões regulares....420
2. Exemplo 1: Verificando o caractere de início....422
3. Exemplo 2: Validando uma string....425
4. Exemplo 3: Validando uma string....427
5. Exemplo 4: Verificando a ocorrência numérica....429
6. Exemplo 5: Verificando a ocorrência numérica de N caracteres....430
7. Exemplo 6: Formatação de CEP....431
8. Exemplo 7: Validando placas no formato MERCOSUL....432
9. Exemplo 8: Validando telefones com DDD....433
10. Exemplo 9: Verificando o início e fim de uma string....435
11. Exemplo 10: Verificando a formatação de e-mail....436
12. Exemplo 11: Capturando sequência numérica na string....437
13. Exemplo 12: Substituindo espaços em branco por hífen....438
Capítulo XVI – Perceptron....442
1. Introdução....442
2. Implementando o perceptron....450

Introdução

A proposta desta literatura é abordar diversos temas fundamentais para estudantes e profissionais que trabalham com Java. Temas relevantes como testes unitários, Spark, Regex, Spring Boot, Lombok, Classes, Interface Gráfica, Perceptron, POO dentre outros são abordados ao longo da literatura.

Para isso, diversos exemplos práticos são abordados apresentando tópicos importantes para se compreender a programação em Java previamente.

A interface de programação usada foi o IntelliJ IDEA versão Community.

Capítulo I

Programação em Java

1. Introdução

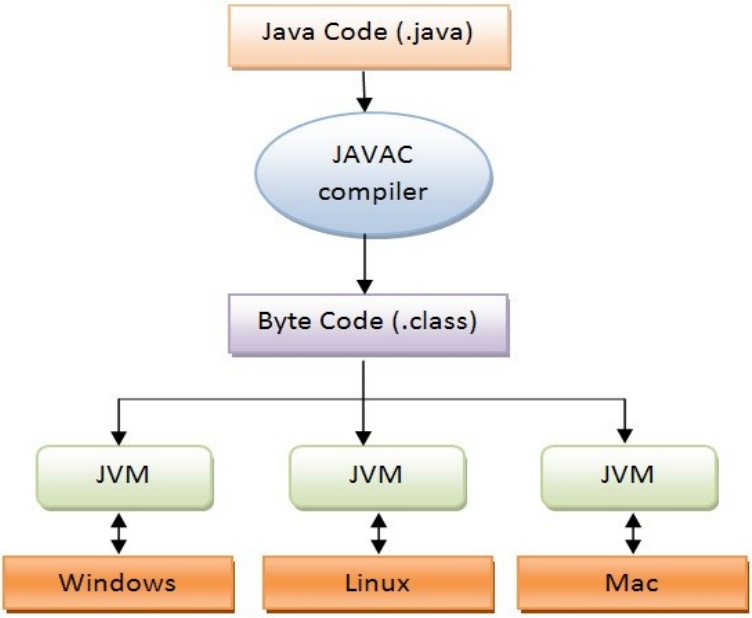
O Java é uma linguagem de programação de propósito geral que foi desenvolvida pela Sun Microsystems na década de 1990. Sua criação é atribuída a uma equipe liderada por James Gosling. A linguagem foi concebida com o objetivo de ser portátil, eficiente e orientada a objetos.

A Sun Microsystems lançou oficialmente o Java em 1995, promovendo-o como uma tecnologia capaz de oferecer "write once, run anywhere" (escreva uma vez, execute em qualquer lugar). Isso significa que os programas Java podem ser executados em qualquer dispositivo que tenha uma máquina virtual Java (JVM) instalada, proporcionando portabilidade entre diferentes plataformas.

Uma das características marcantes do Java é sua capacidade de lidar com a complexidade do desenvolvimento de software, através de conceitos como coleta de lixo (garbage collection) e gerenciamento

automático de memória. Além disso, a linguagem ganhou popularidade na criação de aplicativos web, servidores de aplicações, dispositivos móveis e outros ambientes de desenvolvimento.

A próxima figura apresenta uma figura que mostra que o código fonte escrito em Java na extensão de mesmo que após compilação gerá um código intermediário chamado bytecode, onde este é executado por algum sistema operacional através da JVM.



Em 2010, a Oracle Corporation adquiriu a Sun Microsystems, assumindo o controle do desenvolvimento do Java. Desde então, o Java continua a evoluir, com atualizações regulares e novas versões que incorporam melhorias de desempenho, recursos de linguagem e suporte a novas tecnologias.

Hoje, o Java é uma das linguagens de programação mais utilizadas no mundo, com uma vasta comunidade de desenvolvedores e uma presença significativa em uma ampla variedade de aplicações, desde sistemas embarcados até grandes sistemas empresariais.

2. Ambiente IntelliJ IDEA

A proposta deste capítulo é desenvolver diversas experiências na programação em Java. Estes exemplos são importantes, pois permitem um maior entendimento da linguagem Java.

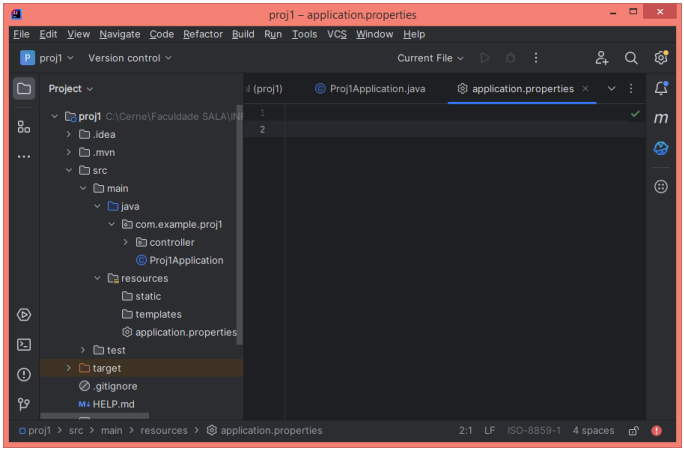
O primeiro passo consiste em baixar a JDK (*Java Development Kit*) de acordo com a versão do seu sistema operacional.

https://www.oracle.com/br/java/technologies/downloads/

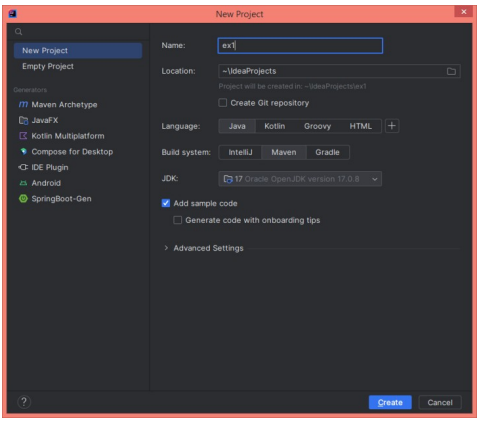
O ambiente de desenvolvimento (IDE) utilizado foi o IntelliJ IDEA, no qual a versão *Community* (gratuita) este pode ser baixado através do link abaixo.

https://www.jetbrains.com/idea/download/?section=windows

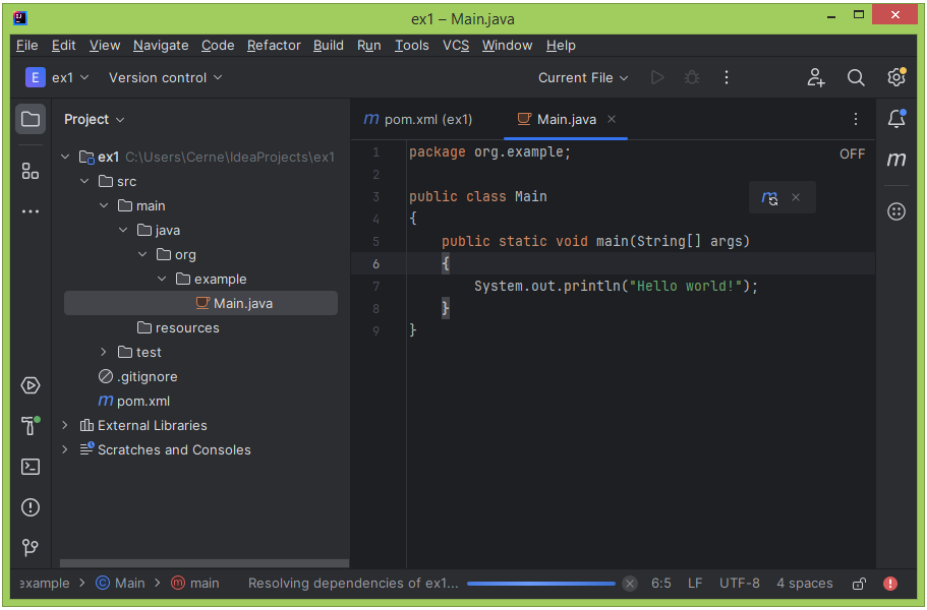
Após o download e instalação, inicialize o IDE. A tela inicial será apresentada na figura abaixo, considerando que a versão utilizada nesta literatura foi a 2023.3.1.



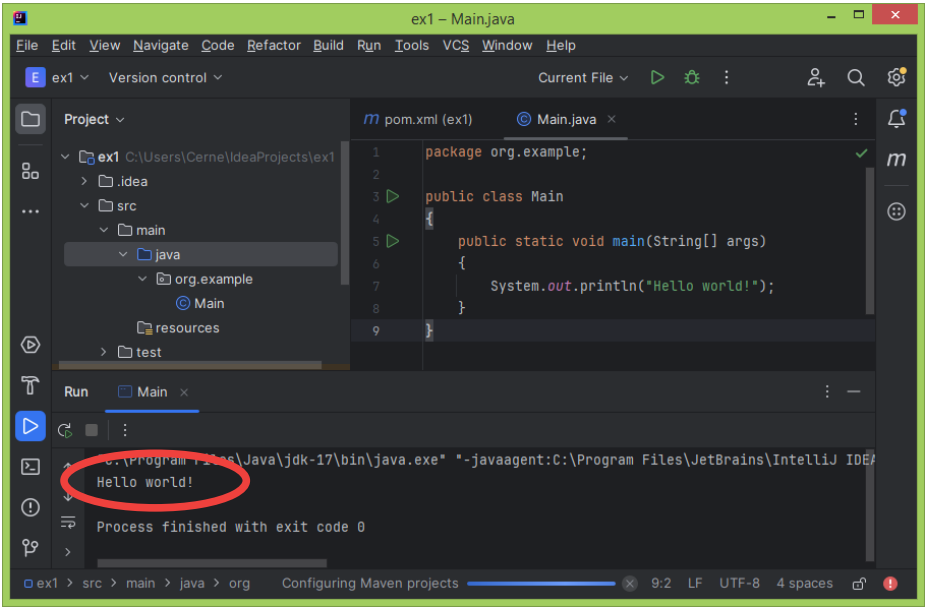
Para criar um projeto, vá ao menu *File-> Project*.



Em *name* escolha o nome do projeto. Em *location,* escolha a pasta no qual o projeto será salvo. Em *language*, escolha Java. Em *build system* escolha *Maven.* Clique no botão Create. A tela ficará conforme a imagem abaixo com o novo projeto criado.



O projeto é criado com um código inicial que imprime um valor no console do ambiente. Para rodar o programa, basta clicar no botão  ou ir ao menu *Build→ Build Project*. Acompanhe o resultado na tela de console.



3. Apresentando mensagens

Digite o código Java abaixo na área de edição.

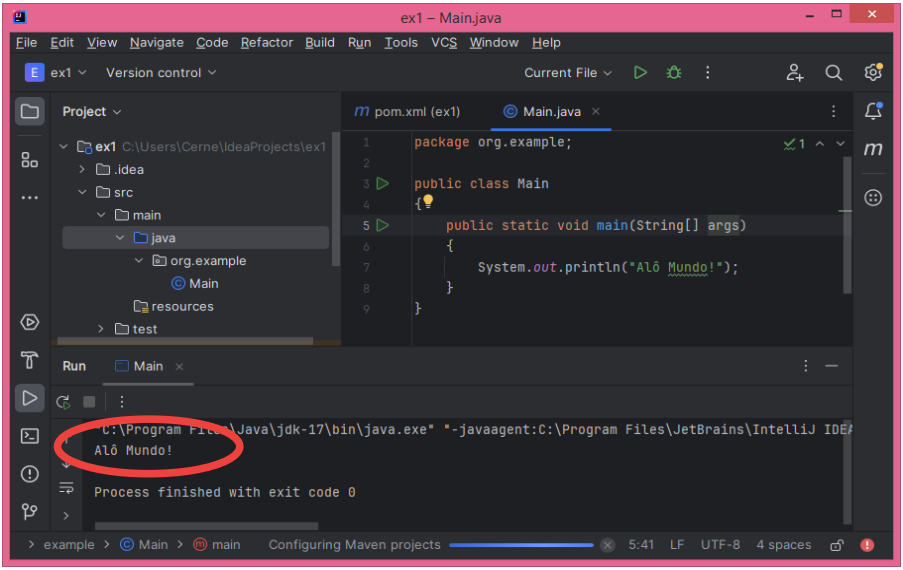
```
package org.example;

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

O programa inicia com a declaração public class Main, onde nesta linha está sendo definida a classe pública que chama-se Main de acordo com o nome do projeto e arquivo criado. A declaração public static void main(String args[]) define a função principal do programa Java, que é a função main.

A linha System.*out*.println("Alô Mundo!") permite imprimir um valor na tela, no caso a frase “Alô Mundo!”. As chaves marcam o início e fim de cada bloco no Java.

Após escreve este programa, execute-o. Veja abaixo o resultado na janela de Console:



Em todos os exemplos propostos estes passos para editar e rodar a aplicação vão se repetir.

4. Comentários

Algo muito importante em qualquer programa escrito em Java é poder comentar o que determinadas funções ou comandos fazem. No exemplo anterior é possível escrever o seguinte comentário:

Imprime uma frase na tela de Vídeo

Observe como ficaria o programa escrito desta forma:

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        System.out.println("Alô mundo!");
       Imprime uma frase na tela de Video
    }
}
```

Caso você tente compilar este código ocorrerá com um erro, pois o Java para entender que determinado trecho de código é um comentário, precisa ser informado através do operador "//". Observe agora como ficaria o exemplo:

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        System.out.println("Alô mundo!");
       //Imprime uma frase na tela de Video
    }
}
```

O comentário de linha no Java é o símbolo barra barra “ // ” além deste comentário é possível ter o comentário de múltiplas linhas, observe o exemplo:

```
package org.example;

public class Main
{
   /*Este
    * aqui
    * é um exemplo
    * de comentário de
    * múltiplas linhas
    */

    public static void main(String[] args)
    {
        System.out.println("Alô mundo!");
       //Imprime uma frase na tela de Video
    }
}
```

Observe que neste caso o comentário inicia com barra estrela “ /* ” e finaliza com estrela barra “ */ ” .

5. Capturando dados no Java

Será demonstrado como ler a idade de uma pessoa. Inicialmente é impresso uma mensagem solicitando o dado, para que em seguida seja feita a leitura da mesma.

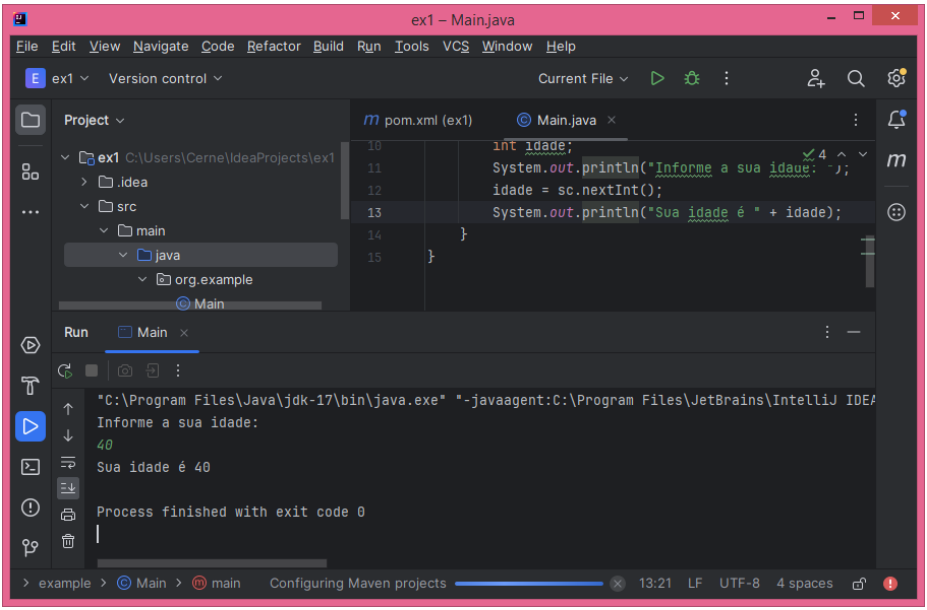
```
package org.example;

import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int idade;
        System.out.println("Informe a sua idade: ");
        idade = sc.nextInt();
        System.out.println("Sua idade é " + idade);
    }
}
```

Inicialmente o programa importa a classe Scanner do Java. Esta classe permite o acesso à entrada de dados do teclado do PC. No bloco main há a declaração do objeto sc do tipo Scanner(System.in) e de outra variável chamada idade. Em seguida, é impressa a mensagem *"Informe a sua idade: "* para que a variável idade recebe o conteúdo

digitado no teclado através do método sc.nextint() e faça a impressão no console em seguida.



6. Calculadora I

O objetivo é solicitar dois números ao usuário e em seguida, apresentar a sua soma numérica. O exemplo fica como apresentado abaixo:

```
package org.example;

import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int n1,n2,n3;
        System.out.println("Primeiro número: ");
        n1 = sc.nextInt();
        System.out.println("Segundo número: ");
        n2 = sc.nextInt();
        n3=n1+n2;
        System.out.println("A soma é " + n3);
    }
}
```

Observe que o exemplo possui muita similaridade com o anterior. As principais diferenças são que a declaração da variável n1, n2 e n3 onde n1 recebe o primeiro valor digitado pelo usuário e n2 o segundo. Em seguida, n3 recebe a soma de n1 mais n2 e finalmente o resultado é impresso apresentando a soma dos dois valores.

7. Calculadora II

A proposta é desenvolver uma calculadora um pouco mais avançada usando a soma, subtração, divisão e multiplicação.

```
package org.example;

import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int n1,n2,n3;

        System.out.println("Primeiro número: ");
        n1 = sc.nextInt();

        System.out.println("Segundo número: ");
        n2 = sc.nextInt();

        n3=n1+n2;
        System.out.println("A soma é " + n3);

        n3=n1-n2;
        System.out.println("A subtração é " +n3);

        n3=n1*n2;
        System.out.println("A multiplic. é " + n3);

        n3=n1/n2;
        System.out.println("A divisão é " + n3);
    }
}
```

Note que a maior diferença deste exemplo para o anterior é o fato de realizar as outras três operações matemáticas adicionais carregando no caso o resultado na variável n3 e imprimindo-a.

8. Tipos de dados

Na tabela abaixo se encontram os diversos tipos de dados existentes na linguagem Java:

Tipo Primitivo	Tamanho	Valor Mínimo	Valor Máximo	Tipo que Encapsula
boolean	-	-	-	Boolean
char	16 bits	Unicode 0	Unicode $2^{16}-1$	Character
byte	8 bits	-128	+127	Byte
short	16 bits	-2^{15}	$+2^{15}-1$	Short
int	32 bits	-2^{31}	$+2^{31}-1$	Integer
long	64 bits	-2^{63}	$+2^{63}-1$	Long
float	32 bits	IEEE754	IEEE754	Float
double	64 bits	IEEE754	IEEE754	Double
void	-	-	-	Void

Através das variáveis são guardados os valores para o posterior tratamento em no programa. Existem diversos tipos de dados e cada um deles é apropriado de acordo com a necessidade do programa. Por exemplo, a variável do tipo byte somente consegue armazenar valores na faixa de -128 a 127 e variável short armazena da faixa de -32768 a 32767.

Outro exemplo é supondo que precise declarar uma variável para armazenar a idade de uma pessoa. Neste caso, a variável do tipo byte

se encaixa perfeitamente, pois dificilmente são encontradas pessoas com idade superior a 127 anos. Se precisar de uma variável para saber quantos litros há em uma caixa de água que suporte no máximo 5000 litros, uma variável do tipo short neste caso poderia se mais adequada pois suporta valores até 32767.

Acompanhe alguns exemplos de código fonte que declaram, inicializam e imprimem o conteúdo de uma variável no console.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        byte x;
        x=33;
        System.out.println(x);
    }
}
```

```
33
```

O próximo exemplo declara e inicializa na mesma linha uma variável. O resultado da impressão continua o mesmo.

```
package org.example;

public class Main
{
```

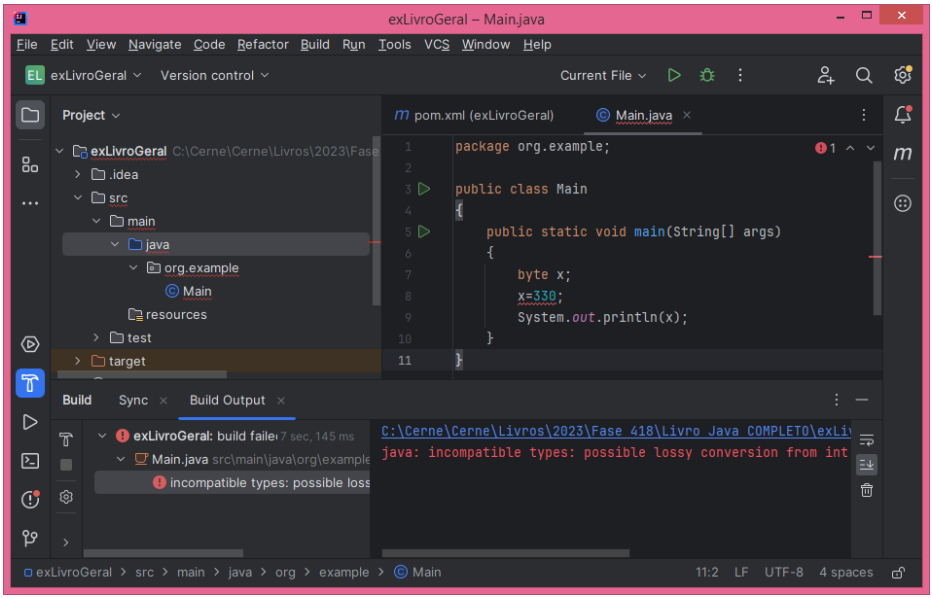
```
    public static void main(String[] args)
    {
        byte x=33;
        System.out.println(x);
    }
}
```



O próximo exemplo declara e inicializa com um valor que o permitido a variável. Neste caso o programa não inicializa, gerando um erro de compilação.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        byte x=330;
        System.out.println(x);
    }
}
```



Os próximos exemplos declaram e inicializam as variáveis de outros tipos de dados.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        short x=3300;
        System.out.println(x);
    }
}
```

```
3300
```

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        short x=100000;
        System.out.println(x);
    }
}
```

```
100000
```

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        long x=999999999;
        System.out.println(x);
    }
}
```

```
999999999
```

```
package org.example;

public class Main
{
```

```
    public static void main(String[] args)
    {
        char x='a';
        System.out.println(x);
    }
}
```

```
a
```

Neste ponto é importante apresentar a tabela ASCII, que faz a conversão de dados do formato numérico para caractere e vice-versa. Há outra tabela chamada unicode que no qual a ASCII está contida e possui além dos caracteres do alfabeto Latino outros conjuntos de alfabetos.

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

No exemplo abaixo a variável x foi inicializada com 58 em hexadecimal que conforme a tabela acima, é o caractere ‘X’.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        char x='\u0058';
        System.out.println(x);
    }
}
```

```
X
```

O próximo exemplo faz o *typecast*, que é a conversão de um dado numérico para caractere.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        int number1 = 66;
        char chars1 = (char)number1;

        int number2 = 76;
        char chars2 = (char)number2;

        int number3 = 79;
        char chars3 = (char)number3;

        int number4 = 71;
```

```
        char chars4 = (char)number4;

        System.out.println(chars1);
        System.out.println(chars2);
        System.out.println(chars3);
        System.out.println(chars4);
    }
}
```

```
B
L
O
G
```

O exemplo a seguir declara uma variável do tipo float, onde neste caso deve estar presente um f ao final do valor numérico.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        float num1=3.14592f;
        System.out.println(num1);
    }
}
```

```
3.14592
```

A seguir a mesma declaração, porém usando a variável do tipo double.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        double num1=3.14592;
        System.out.println(num1);
    }
}
```

```
3.14592
```

A seguir um exemplo com variável booleana.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        boolean y=true;
        System.out.println(y);
    }
}
```

```
true
```

O próximo experimento faz diversas operações aritméticas com uma variável inteira.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        int x=10;

        x+=4;
        System.out.println("Soma= " + x);

        x-=6;
        System.out.println("Subtração= " + x);

        x*=6;
        System.out.println("Produto= " + x);

        x/=3;
        System.out.println("Divisão= " + x);

        x%=5;
        System.out.println("Resto da divisão= " + x);
    }
}
```

```
Soma= 14
Subtração= 8
Produto= 48
Divisão= 16
Resto da divisão= 1
```

O exemplo abaixo inicia a variável x e y e logo em seguida faz o teste de comparação e diferença booleano entre os seus valores.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        int x=10,y=20;
        boolean z;

        z = x==y;
        System.out.println("x==y? " + z);

        z = x!=y;
        System.out.println("x!=y? " + z);
    }
}
```

```
x==y? false
x!=y? true
```

O objetivo do próximo exemplo é testar os operadores relacionais usando como referência as variáveis x e y.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        int x=10,y=20;
        boolean z;

        z = x>y;
        System.out.println("x>y? " + z);
```

```
        z = x>=y;
        System.out.println("x>=y? " + z);

        z = x<y;
        System.out.println("x<y? " + z);

        z = x<=y;
        System.out.println("x<=y? " + z);
    }
}
```

```
x>y? false
x>=y? false
x<y? true
x<=y? true
```

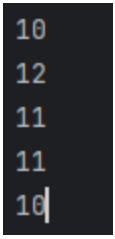
O exemplo abaixo testa os operadores de incremento e decremento de variáveis. Observe que a execução do código é da esquerda para direita, onde em alguns casos ocorre a impressão primeiro e logo em seguida a operação e vice-versa.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        int x=10;
        System.out.println(x++);
        System.out.println(++x);
```

```
        System.out.println(--x);
        System.out.println(x--);
        System.out.println(x);
    }

}
```



O próximo experimento trabalha com outra base numérica diferente da tradicional na base 10. Observe que a variável é inicializada em binário e logo em seguida o seu conteúdo é impresso na base 2, 8 e 16.

```
package org.example;

public class Main
{
    public static void main(String[] args)
    {
        byte x=0b00001011;
        System.out.println(x);
        System.out.println(Integer.toString(x,2));
        System.out.println(Integer.toString(x,8));
        System.out.println(Integer.toString(x,16));
    }
}
```

```
11
1011
13
b
```

O exemplo a seguir captura um valor do console e atribui a uma variável chamada nome do tipo String e logo em seguida imprime uma saudação ao usuário.

```
package org.example;
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        String nome;
        Scanner inp = new Scanner(System.in);
        System.out.println("Qual o seu nome? ");
        nome = inp.next();
        System.out.println("Seja bem-vindo " + nome + " ao Java!");
    }
}
```

```
Qual o seu nome?
Vitor
Seja bem-vindo Vitor ao Java!
```

9. Operadores da linguagem

Conheça uma série de operadores da linguagem Java que serão usados no decorrer da literatura:

Operação	Descrição
=, +=, -=, *=, /=, %=	Operadores de atribuição
==, !=	Operadores de igualdade e diferença
<, <=, >, >=	Operadores de desigualdade
+, -	Operadores unários
+, -, *, /, %	Adição, subtração, multiplicação, divisão e módulo
+=, -=, *=, /=, %=	Operadores de atribuição com adição, subtração, multiplicação, divisão e módulo
++, --	Incremento e decremento
<<, >>, >>>	Operadores de deslocamento de bits
<<=, >>=, >>>=	Operadores de atribuição com deslocamento de bits
~	Operador lógico de negação
&, |, ^	Operadores lógicos E, OU e OU-exclusivo
&=, |=, ^=	Operadores de atribuição com operação lógica E, OU e OU-exclusivo

1) No exemplo proposto solicita-se dois valores ao usuário e fornece como resultado *true (verdadeiro)* se forem iguais e *false (falso)* sendo diferentes.

```
package org.example;
```

```
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int n1,n2;
        System.out.println("Primeiro número: ");
        n1 = sc.nextInt();
        System.out.println("Segundo número: ");
        n2 = sc.nextInt();
        System.out.println("O resultado é "+(n1==n2));
    }
}
```

2) Solicita-se que seja digitado dois números e retorna o resultado da divisão assim como o resto da divisão. Por exemplo, 10 dividido por 3 em inteiros dá o resultado 3 com resto 1.

```
package org.example;

import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int n1,n2;
        System.out.println("Primeiro número: ");
        n1 = sc.nextInt();
```

```
        System.out.println("Segundo número: ");
        n2 = sc.nextInt();
        System.out.println("A divisão é " + (n1/n2));
        System.out.println("O resto é " + (n1%n2));
    }
}
```

3) Utilize os operadores de incremento e decremento para incrementar (somar 1) ou decrementar (subtrair 1) o valor de uma variável.

```
package org.example;

import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int n1,n2;
        System.out.println("Primeiro número: ");
        n1 = sc.nextInt();
        n2=n1;
        System.out.println("O incremento é " + (++n1));
        System.out.println("O decremento é " + (--n2));
    }
}
```

4) Este exemplo explora os operadores lógicos AND e OR. O objetivo é que de acordo com os dois valores digitados na entrada seja possível

realizar tais operações e em seguida apresentar o resultado de acordo com os números passados.

```
package org.example;

import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int n1,n2;
        System.out.println("Primeiro número: ");
        n1 = sc.nextInt();
        System.out.println("Segundo número: ");
        n2 = sc.nextInt();
        System.out.println("AND: " + (n1 & n2));
        System.out.println("OR: " + (n1 | n2));
    }
}
```

10. Estrutura de repetição for

As estruturas de repetição são um meio interessante quando for preciso repetir um número de vezes o bloco de código ou aguardar até que uma determinada condição seja atendida no programa. Esta estrutura tem a sintaxe apresentada abaixo:

```
for(inicialização;expressão;passo)
```

```
{
}
```

Observe que o loop possui um valor de inicialização e é executado até que a condição expressa em *expressão* seja verdadeira. A variável que foi inicializada também é incrementada de acordo com o valor definido em *passo.* Observe abaixo um exemplo que imprime os valores de 1 a 10 no Java usando a estrutura *for:*

```
package org.example;

import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        int contagem;
        for(contagem=1;contagem<=10;contagem++)
        {
            System.out.println(contagem);
        }
    }
}
```

Primeiro foi declarada uma variável chamada contagem do tipo inteiro e em seguida entrou em um loop de 10 vezes, iniciando a