



Programação Orientada à Objetos utilizando Java

Leonardo Oliveira, CaP/ETS/ 03/05/2022

04

Comandos Condicionais, Conversões de Tipos, Laços de Repetições e Tratamento de Exceções.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Comando Condicionais

```
Novo.java x
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         double media=7.1;
6         int faltas=20;
7         boolean postura=true;
8         String situacao;
9         if(media>=5.0 && faltas<25 && postura==true)
10        {
11            situacao="aprovado";
12        }
13        else if (media<5.0 && faltas<25 && postura)
14        {
15            situacao="Recuperação";
16        }
17        else if (media>5.0 && faltas>=25&&postura)
18        {
19            situacao="sem férias";
20        }
21        else if (media>5.0 && faltas<25 &&postura==false)
22        {
23            situacao="chamar pra conversar";
24        }
25        else
26        {
27            situacao="reprovado";
28        }
29        System.out.println(situacao);
30    }
31 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Conversões de Tipos

```
1  package bosch;
2
3  ▶ public class Novo {
4  ▶   public static void main(String[] args) {
5      Byte b = 100;
6      Short s = 1000;
7      Integer i = 10000;
8      Long l = 100000L;
9
10     System.out.println(b.byteValue());
11     System.out.println(s.toString());
12     System.out.println(i*3);
13     System.out.println(l/3);
14
15   }
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Conversões de Tipos

```
1 package bosch;
2
3 ▶ public class Novo {
4 ▶   public static void main(String[] args) {
5     Float f =123.10F;
6     System.out.println(f);
7
8     Double d = 1234.5678;
9     System.out.println(d);
10
11     Boolean bo = Boolean.parseBoolean(s: "true");
12     System.out.println(bo);
13     System.out.println(bo.toString().toUpperCase());
14
15   }
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Conversões de Tipos

```
1 package bosch;
2
3 ▶ public class Novo {
4 ▶   public static void main(String[] args) {
5     double a=1;
6     System.out.println(a);
7
8     float b = (float) 1.122499999999;
9     System.out.println(b);
10
11     int c=127;
12     byte d = (byte) (c);
13     System.out.println(d);
14
15     double e =1.999999999999;
16     int f = (int) e;
17     System.out.println(f);
18   }
19 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Conversões de Tipos

```
1  package bosch;
2
3  ▶ public class Novo {
4  ▶  public static void main(String[] args) {
5      Integer num1=10000;
6      System.out.println(num1.toString().length());
7
8      int num2 = 1000000;
9      System.out.println(Integer.toString(num2).length());
10
11     System.out.println((""+num2).length());
12 }
13 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Conversões de Tipos

```
1  package bosch;
2
3  ▶ public class Novo {
4  ▶     public static void main(String[] args) {
5         String numero1= "12";
6         String numero2="3.14";
7
8         int x = Integer.parseInt(numero1);
9         double y = Double.parseDouble(numero2);
10        double soma =x+y;
11        System.out.println(x);
12        System.out.println(y);
13        System.out.println(soma);
14    }
15 }
```


PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Laco.java x
1  package bosch;
2
3  ▶ public class Laco {
4  ▶  public static void main(String[] args) {
5      int contador=0;
6      while (contador<=20)
7      {
8          System.out.printf("O contador está em: %d\n",contador);
9          contador++;
10     }
11 }
12 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1  package bosch;
2
3  ▶ public class Novo {
4  ▶     public static void main(String[] args) throws InterruptedException {
5      for (int i = 0; i < 10 ; i++) {
6          System.out.println(i);
7          Thread.sleep( millis: 1000);
8      }
9  }
10 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1      package bosch;
2
3      ▶ public class Novo {
4      ▶     public static void main(String[] args) {
5          int x=2;
6          for (;x<10;)
7          {
8              System.out.println(x);
9              x++;
10         }
11     }
12 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1  package bosch;
2
3  ▶ public class Novo {
4  ▶     public static void main(String[] args) {
5         for(;true;)
6         {
7             System.out.println("Hello");
8         }
9     }
10 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1      package bosch;
2
3      import java.util.Scanner;
4
5      public class Novo {
6      public static void main(String[] args) {
7          Scanner entrada = new Scanner(System.in);
8          String valor="";
9          while (!valor.equalsIgnoreCase( anotherString: "sair"))
10         {
11             System.out.print("Digite algo: ");
12             valor=entrada.nextLine();
13         }
14         entrada.close();
15     }
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1      package bosch;
2
3      import java.util.Scanner;
4
5      public class Novo {
6      public static void main(String[] args) {
7          Scanner entrada = new Scanner(System.in);
8          String valor = "";
9          do{
10             System.out.print("Diga-me algo: ");
11             valor= entrada.nextLine();
12         }while (!valor.equalsIgnoreCase( anotherString: "algo"));
13         entrada.close();
14     }
15 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1  package bosch;
2
3  import java.util.Scanner;
4
5  public class Novo {
6      public static void main(String[] args) {
7          Scanner entrada = new Scanner(System.in);
8          int contador=0, acumulador=0;
9          while (contador<5){
10             contador++;
11             System.out.print("Dgite um número: ");
12             acumulador+=entrada.nextInt();
13             entrada.nextLine();
14             System.out.printf("Contador está em %d e " +
15                 "o acumulador está em %d\n",contador,acumulador);
16         }
17     }
18 }
```

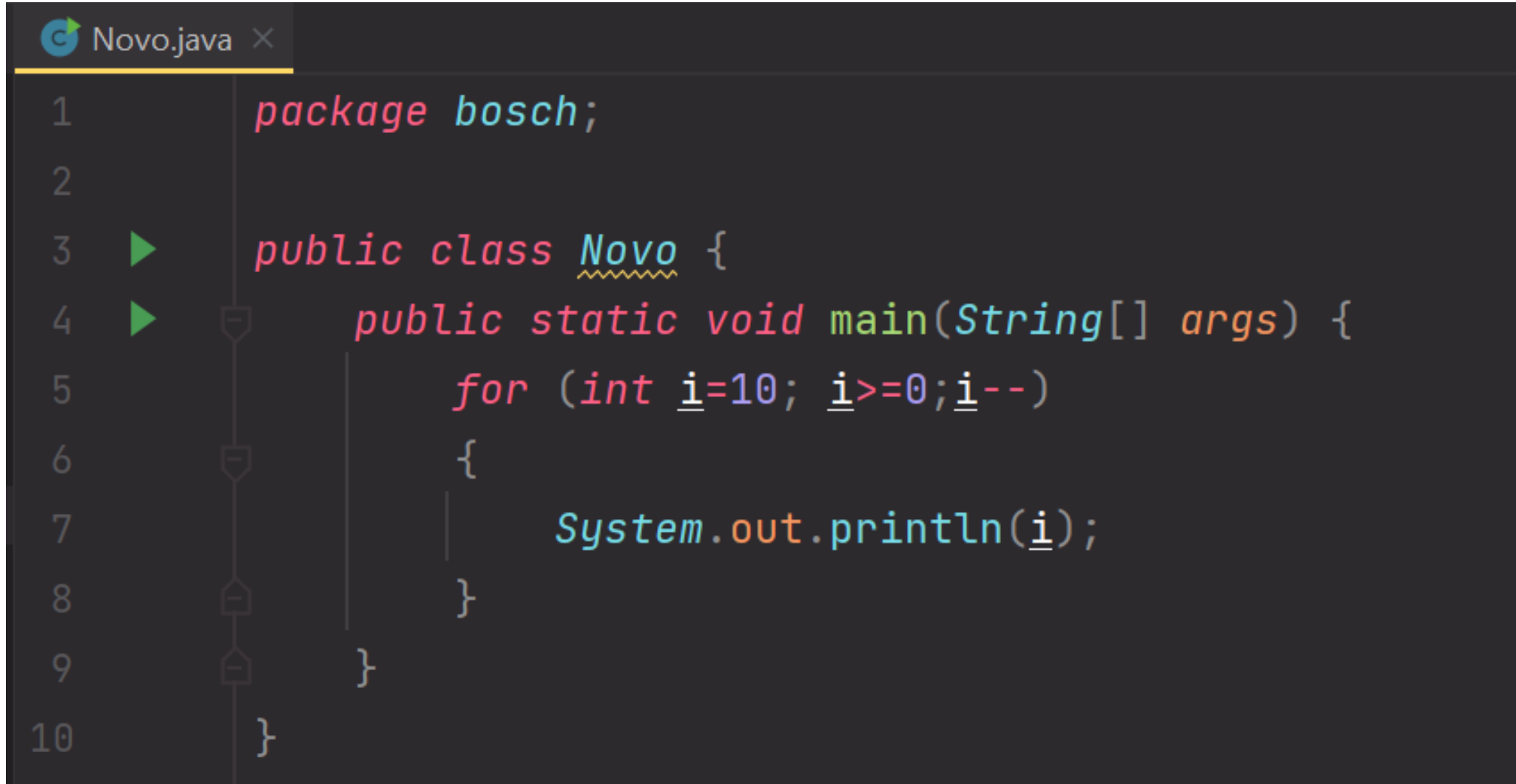
PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1      package bosch;
2
3      public class Novo {
4      public static void main(String[] args) throws InterruptedException {
5          int contador=0, acumulador=0;
6          for(int i=0; i<5; i++){
7              contador=i;
8              acumulador+=i*i;
9              System.out.printf("Contador está em %d e " +
10                  "o acumulador está em %d\n", contador, acumulador);
11              Thread.sleep( millis: 2000);
12          }
13      }
14  }
```


PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição



```
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         for (int i=10; i>=0; i--)
6         {
7             System.out.println(i);
8         }
9     }
10 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1      package bosch;
2
3      public class Novo {
4      public static void main(String[] args) {
5          for (int i=1; i<=10;i++)
6          {
7              for (int j = 0; j <=10 ; j++) {
8                  System.out.printf("%d X %d = %d\n",i,j,i*j);
9              };
10             System.out.println();
11         }
12     }
13 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1      package bosch;
2
3      ▶ public class Novo {
4      ▶     public static void main(String[] args) {
5          for (int i=1; i<=10;i++)
6          {
7              if (i==5)
8              {
9                  break;
10             }
11             System.out.println(i);
12         }
13     }
14 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x
1      package bosch;
2
3      public class Novo {
4      public static void main(String[] args) {
5          for (int i=1; i<=10;i++)
6          {
7
8              if(i%2==0)
9              {
10                 continue;
11             }
12             System.out.println(i);
13         }
14     }
15 }
16 }
```

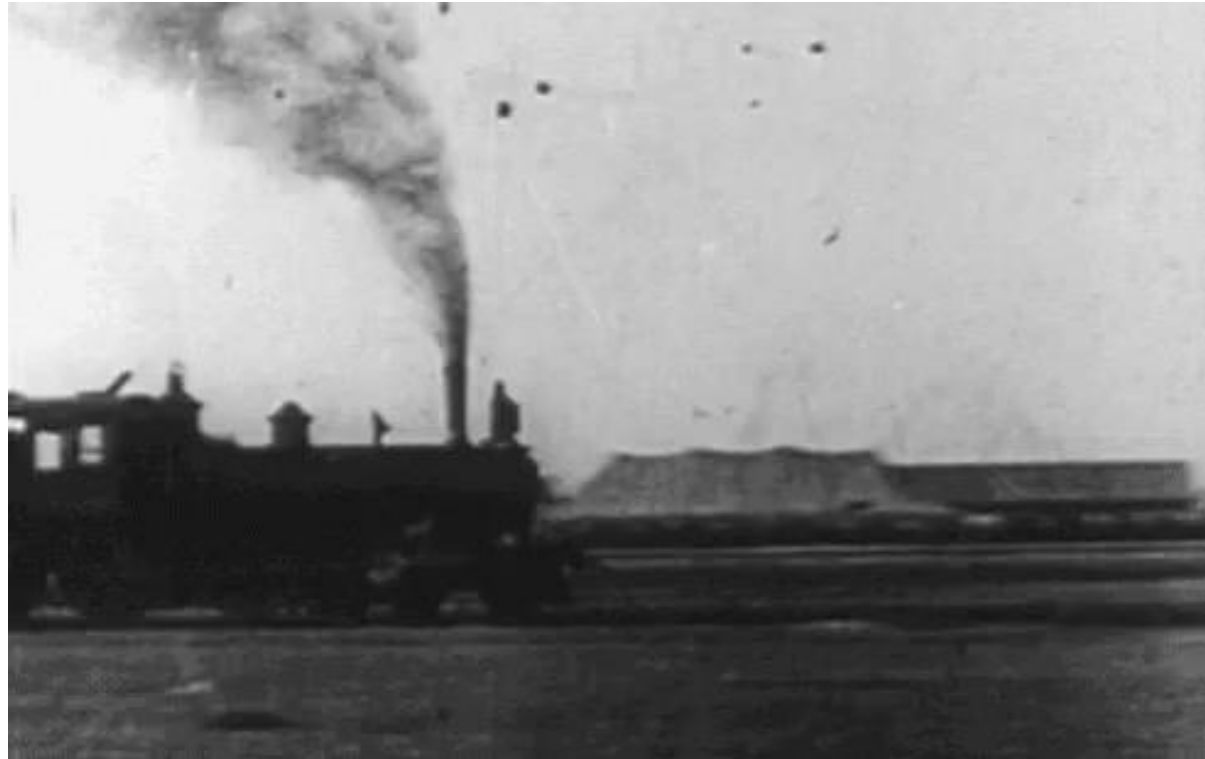
PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x Main.java x
1 package bosch;
2
3 import java.util.InputMismatchException;
4 import java.util.Scanner;
5
6 public class Main {
7
8     public static void main(String[] args) {
9         Scanner ler = new Scanner(System.in);
10        int numero;
11        while (true){
12
13            try {
14                System.out.print("Digite um numero: ");
15                numero = ler.nextInt();
16                if (numero<20)
17                {
18                    continue;
19                }
20                break;
21            } catch (InputMismatchException e) {
22                System.out.println("Ops... você digitou caracteres. Precisamos que digite apenas números.");
23            }
24            ler.nextLine();
25        }
26        System.out.println(numero);
27    }
28 }
```

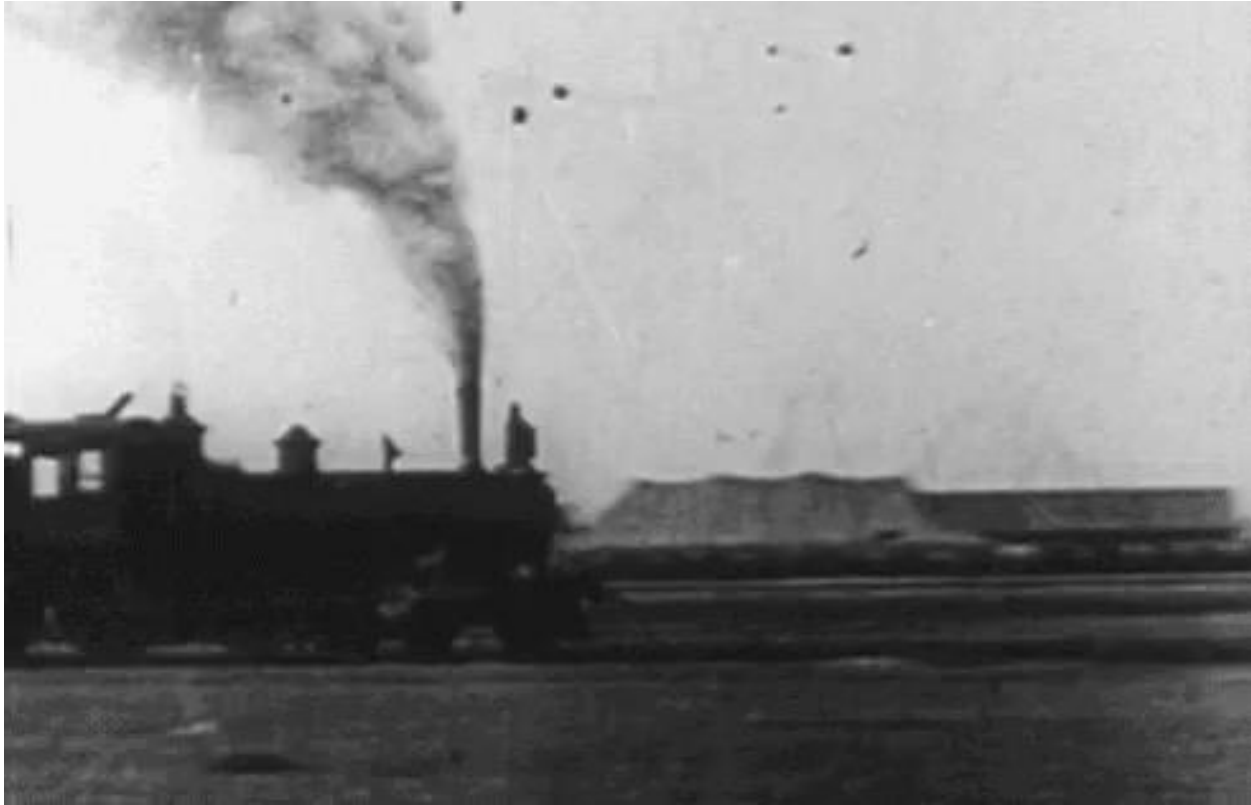
LAB 01 – Colisão de Trens

Suponha que dois trens partam ao mesmo tempo de cidades diferentes, de maneira que em algum momento eles irão colidir. Você não pode fazer nada para impedir a tragédia, a única coisa que você pode fazer é determinar o instante e o local que colidem.



JAVA

LAB 01 – Colisão de Trens



- Considere que as extremidades da ferrovia vão do KM 0 até o KM 10.000
- Considere que a velocidade do trem A sempre será positiva e a velocidade do trem B sempre será negativa.
- Considere que o módulo da velocidade do trem será de no máximo 300 km/h
- Seu programa terá 4 variáveis de entrada: posição do trem A, posição do trem B, velocidade do trem A e velocidade do trem B.
- Seu programa deve exibir após quantos segundos ocorreu a colisão e em que KM ocorreu a colisão.

LAB 01 – Colisão de Trens – Requisitos Básicos



- Seu programa deve exibir o print no seguinte formato:

```
System.out.printf("A colisão de trens acontecerá no KM %f e ocorrerá"+  
    " após %f segundos no horario de %s",posicao,tempo,horario);
```

- Faça com que nas entradas sejam aceito somente números, se necessário crie uma função para isso.
- Ao encerra o programa faça com que apareça “FIM DO PROGRAMA” na tela

LAB 01 – Colisão de Trens – Requisitos de Desafio



- Faça com que seu programa limite a posição entre o KM 0 e KM 10.000 e exiba uma mensagem de erro caso seja uma posição inválida
- Faça com que seu programa limite o módulo da velocidade dos trens a 300 km/h e exiba uma mensagem de erro caso valor seja inválido
- Faça com que a velocidade do trem A seja sempre positiva e a velocidade do trem B seja sempre negativa.
- Faça com que seu programa pergunte ao usuário se deseja executar novamente.
- Considerando que ambos os trens partem de suas respectivas cidades às 17 h, determine o horário da colisão no formato 00:00:00.
- Existe uma situação específica em que os trens não irão colidir, determine qual é esta situação e faça com que seu programa mostre uma mensagem falando que os trens não irão colidir.

LAB 01 – Colisão de Trens - Fórmulas

$$S(t) = S0 + V.t$$

Equação horaria do espaço

$$Sa(t) = S0a + Va.t$$

Equação do trem A

$$Sb(t) = S0b - Vb.t$$

Equação do trem B

$$t = \frac{S0a - S0b}{Vb - Va}$$

Equação do tempo

- S = Posição final
- S0 = Posição inicial
- V = Velocidade
- t = Tempo
- a = Refere-se ao trem A
- b = Refere-se ao trem B

LAB 01 – Colisão de Trens – Entradas e Saídas

ENTRADAS	SAÍDAS
S0a=0, S0b=200, Va=20, Vb=-30	KM 80, 14400 segundos, 21:00:00
S0a=20, S0b=100, Va=100, Vb=-100	KM 60, 1440 segundos, 17:24:00
S0a=300, S0b=400, Va=20, Vb=0	KM 400, 18000 segundos, 22:00:00
S0a=800, S0b=3500, Va=200, Vb=-250	KM 2000, 21600, 23:00:00
S0a=H	Você digitou um caractere inválido. Por favor, digite novamente:
S0a=15000	O número deve ser maior ou igual a zero e menor que 10000.
S0a=20, S0b=60, Va=-240	A velocidade do trem A sempre será positiva. Por favor, digite novamente:
S0a=400, S0b=600, Va=280, Vb=200	A velocidade do trem B sempre será negativa. Por favor, digite novamente:
S0a=40, S0b=400, Va=200, Vb=c8	Você digitou um caractere inválido. Por favor, digite novamente: